Towards Unifying Semantic Constraints and Security Constraints in Distributed Information Systems

Dissertation

von

Barbara Sprick

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Universität Dortmund
am Fachbereich Informatik

Dortmund 2003

Tag der mündlichen Prüfung: 6. Oktober 2003
Dekan: Prof. Dr. Bernhard Steffen
1. Gutachter: Prof. Dr. Joachim Biskup
2. Gutachter: Prof. Dr. Ernst-Erich Doberkat

Acknowledgement

My work would not have been possible without the help, the guidance and the patience of a lot of people.

First and foremost, I wish to thank my supervisor Joachim Biskup for his guidance, his encouragement and his patience. His extensive way of giving comments on and making suggestions for my work has always significantly and positively influenced both form and content of the work.

I especially owe a lot of thanks to my colleague Ralf Menzel. Ralf was always willing to discuss ideas as well as technical details with me. The discussions with him were invaluable for my understanding of the interaction of knowledge and action. Many times when I got stuck in a proof he helped me with his straight forward thinking and with his ability to arrange my ideas. It was a great pleasure to work with him.

Very important for the development of the tableau system were the fruitful discussions with Hubert Wagner. When I had no idea how to tackle the interaction of knowledge and belief in the tableau system, he helped me with his broad knowledge about logics in general and about tableau systems. The discussions with him were always motivating and invaluable for my work.

I thank Professor Doberkat for agreeing to be the second referee of this work and for his comments on the text.

I would also like to thank my other colleagues, Christian Altenschmidt, Ulrich Flegel, Jürgen Freitag, Yücel Karabulut, Thomas Leineweber, Frank Müller, Jörg Parthe and Sandra Wortmann for the good cooperation in the group and for all the support in my fights with the computers. Especially Frank, with his mathematical background, was of great help for details of the numerous proofs in my work.

I am most grateful for all the support I got from Sudhir Agarwal, for his moral support, for listening to all my problems, to each day's success and failures and for his enthusiasm for discussing ideas.

My flat mates Sabine Kühle and Birgitta Wolf showed a lot of consideration for me and patiently listened to all my reports about failures. They always cheered me up in long evenings with good food and wine.

Naturally, my work would not have been possible without my parents. I owe them a lot of thanks for their untiring moral support.

Abstract

Modern information systems must respect certain restrictions in order to guarantee the proper and desired functionality. Semantic constraints help to prevent inconsistencies in the stored data resulting from faulty updates. Security constraints are to maintain integrity, secrecy and availability over updates and over queries. This thesis designs a unifying framework for the specification of semantic constraints and security constraints in information systems in order to study interactions between them.

We consider an information system as a distributed, reactive system in which each actor and each object acts autonomously and concurrently. Actors gain knowledge by performing read operations on objects and they may update the content of an object by performing update operations. To execute read or update operations, actors need execute rights that can be granted or revoked by other actors.

This view of an information system is captured in a computational model. In this model, we consider each component of the information system, actors as well as objects, uniformly as a sequential agent that performs operations autonomously and jointly with other sequential agents. Each agent is affiliated with a set of local propositions and a set of local operations as well as with relations that capture the agent's knowledge and belief.

An agent's knowledge is determined completely by its local state. Change in knowledge of an agent is due to operations performed by the agent. Interaction between knowledge and operations is captured by the requirement that the enabling and the effect of an operation is completely determined by the knowledge of the acting agents. Knowledge of agents can be changed only by operations in which they participate.

We define a temporal and epistemic specification language with temporal and epistemic operators. The logic provides for each agent local next and until operators as temporal operators and local knowledge and belief operators as epistemic operators.

We develop a modal tableau based proof system for a subset of the logic and show its soundness. Completeness can be shown only for a smaller, but still reasonable subset of the logic, decidability remains an open question. The main difficulty of the tableau system arises from the interaction requirement between knowledge and action.

In a detailed example we demonstrate how the framework can be used for specifying semantic constraints and security constraints in information systems.

Contents

Chapter 1

Introduction

An information system models a part of the "real world" while respecting certain constraints in order to guarantee the proper and desired functionality. These constraints can be imposed by the real world as well as by the application context and can be declared on the stored data in the information system as well as on the behavior of the system and its users.

Semantic constraints help to prevent the information system to have data that is incompatible to the real world. For example,

- a 25 years old employee cannot have a work history of 30 years,
- each employee has a social security number,
- an employee's salary may not decrease,
- every person can either be male or female.

Since data incompatible to the real world may result from faulty updates, semantic constraints are to be checked during the execution of update operations. They restrict the possible (sequences of) updates in each state of the information system.

Security constraints help to maintain integrity, secrecy and availability over (sequences of) updates as well as over queries. For example,

- an employee may not increase her own salary,
- an administration employee in a hospital may not alter laboratory data of a patient,
- secret information may not be disclosed to an unclassified user,
- whenever a user queries the system, she must eventually get the requested information,

While semantic constraints are concerning the values of data in the information system, security constraints are more concerning the circumstances under which the data may be modified or disclosed.

Some years ago, information systems were mostly monolithic: We could assume that data was stored centrally in one place that there was one schema for the database which included all semantic constraints imposed on the data and thus that there was a central and global control over the system.

Over time, the data stored in information systems has been dramatically increasing, and the potentials of combining information systems over a network have been growing. So, distributed systems with heterogeneous distributed components became more and more important.

We can now consider an information system as a highly distributed system with heterogeneous distributed components. In such systems, it seems unrealistic to assume anything like a global schema or a centralized mechanism for updates with enforcement of semantic constraints and for query evaluation. Instead, we have to assume that each individual component has only a restricted view on the whole system. Accordingly, security is considered as being concerned with the different interests and views of the various components. Here, it seems to be rather undesirable to assume anything like a global security policy or centralized supervising mechanisms. Instead, whenever possible, security interests should be enforced by the components autonomously. In the remainder of this thesis, whenever we consider semantic constraints and security constraints, we will have to keep in mind that these constraints are employed in a distributed environment. In particular, even if an ideal observer thinks in terms of a global state, which, however, is questionable because of the problem of synchronized time, in practice any component will have only partial knowledge about that fictitious state.

Our goal is to provide a unifying framework for both kinds of constraints, semantic constraints as well as security constraints declared for distributed information systems in order to study interactions between these constraints on the conceptual level at design time in a manageable way supported by an algorithmic tool.

In the following we review the basic features of the two kinds of constraints that we have in mind to provide a unifying framework for both of them.

1.1 Various Types of Constraints for Information Systems

Let us first investigate the various types of constraints in information systems in order to justify the required features of our framework.

1.1.1 Semantic Constraints

Semantic constraints reflect properties of an outside “mini-world” that is modeled by the information system. There are several possibilities to classify these properties: We can distinguish between

- properties that can be found in the real world (like *An employee of 25 years age cannot have a work history of 30 years*, *A person cannot have two biological mothers* or *A person has exactly one birth date*) and
- properties that are be imposed by the application requirements (like *Every employee must have a social security number*, *A person may get her driving license only after she is 18 years old* or *Every participant of the lab course must be a student*).

Another way to classify semantic constraints is to distinguish between

- static constraints, that are properties relative to one instance of the information system (like the above constraints *Every employee must have a social security number*, *Every participant of the practical course must be a student* or *A person may sign an invoice only if she holds the secret signature key*) and
- dynamic constraints, that are properties regarding a sequence of instances over time (like *The salary of an employee may not decrease*, *Once a bank account exists, it exists forever* or *A prescription can be used only once*).

Semantic constraints are invariants that need to be maintained by the information system whenever the current instance of the information system is changed by some update operation. Further, semantic constrains are intended to support the users of an information system when they interpret instances of a system in terms of a real world.

1.1.2 Security Constraints

Security constraints reflect obligations and restrictions of human individuals in the outside mini-world modeled by the information system. Their main purpose is to maintain *secrecy, integrity,* and *availability.*

Maintaining *secrecy* means preventing the improper disclosure of data. Users may access information directly by querying the information system or indirectly through logical conclusions of their knowledge or belief. Thus, we can distinguish between two types of confidentiality. The first type is “authorization confidentiality”, roughly meaning queries should be successfully invoked only by authorized users. The second type of confidentiality roughly means that the users' knowledge and belief should comply with the task specific restrictions, they should not be able to infer information they are not entitled to know.

Maintaining *availability* means avoidance of denial of service. Availability has two aspects: Firstly, whenever information is requested by a user, the system should eventually provide this information or, in other words, user knowledge should eventually comply with the task specific richness. Secondly, whenever a user submits an (update) operation, this operation should eventually be executed.

Maintaining *integrity* means preventing the improper modification of data. In the context of security requirements this can be seen as "authorization integrity": updates should be successfully invoked only by authorized users, e.g. in a company, employees may not increase their own salary.

Security constraints are invariants that are to be maintained by the system whenever some operation sequences are executed on behalf of an actor. Here, update operations (e.g. for maintaining integrity) as well as read operations (e.g. for maintaining secrecy) are important. Further, security constraints are intended to support system users when they employ the system as a communication tool, e.g. by controlling the flow of information.

Often we cannot strictly distinguish among integrity in the context of security, integrity as semantic constraints, and integrity required by availability. However, in all the three contexts, we sometimes have obligations concerning sequences of operations: a user should eventually invoke a required operation, a collection of users should eventually invoke required sequences of operations or stored data should eventually reach required states. These obligations can be seen as *weak* obligations: it is not required, that they always are always satisfied, or that they are satisfied immediately after some checkpoint. It is only required that they will eventually be satisfied.

Similarly, constraints in the context of availability can be seen as *weak* constraints: again, it is only required that the system has to execute requested operations eventually and not necessarily immediately.

1.2 The Problem

There are subtle interactions of semantic constraints and security constraints in information systems. For example, the semantic constraints may demand that if some data $\mathfrak{a}$ is modified, then data $\mathfrak{b}$ needs to be modified accordingly, while the security constraints allow the user to modify only data $\mathfrak{a}$ but not data $\mathfrak{b}$. Another example could be that the knowledge about semantic constraints reveals information to a user that should be kept secret from her.

To guarantee the desired functionality of an information system, the specification of the system and of the constraints should cover the requirements imposed by the real world and the application context. It is essential to investigate the interactions between the various types of constraints to ensure the correct design. The question that arise are:

- Are the imposed constraints consistent?
- Does the system description ensure the desired constraints?

Interactions of semantic constraints and security constraints can be investigated only if they are uniformly expressed in the same framework.

Our aim is to develop such a framework and to have an (at least semi-) automatic proof tool to answer the above questions.

1.3 Analysis of the Problem

An information system models a part of the real world, often called "mini-world". The outside mini-world imposes certain restrictions on the data stored in its objects. Semantic constraints of an information system reflect these restrictions of the modeled mini-world.

In section 1.1.1 we distinguished between two types of semantic constraints, namely static semantic constraints and dynamic semantic constraints. Static constraints restrict the possible *states* of the information system, whereas dynamic constraints restrict possible *updates* and *sequences of updates*. Thus, **our framework must be able to deal with states and (sequences of) updates.**

The outside mini-world imposes certain obligations and restrictions on its actors. Security constraints of an information system reflect these obligations and restrictions of the modeled outside mini-world.

In section 1.1.2 we identified three main types of security constraints, namely *secrecy*, *integrity*, and *availability*. Secrecy constraints restrict the flow of information by keeping track of the actors' knowledge and by restricting read access to data. Thus, **our framework must have a notion of actors' knowledge and of execute rights for read operations.**

Integrity constraints restrict the possible updates and sequences of updates by restricting the actors' write access to objects. Thus, **our framework must have a notion of execute rights for write operations.**

Availability constraints impose obligations on the system. In a weak sense, these obligations can be modeled using temporal notions: If a user submits an operation, the system should *eventually* execute it. Thus, **our framework must have a notion of time.**

In the context of semantic constraints as well as in the context of security constraints we sometimes talk about *sequences of operations*. Such sequences can also be modeled using temporal notions.

Further, the framework must be able to formalize **actors**, e.g. users, administrators etc. These actors perform *operations*, thus **we need a notion of operations**.

Our framework must also be capable of formalizing **stored data**, e.g. by formalizing **objects** whose current state represents the stored data. On these objects, **operations** are executable. Also, the current state of the objects may change over time by the execution of operations, which means that *the state of objects over time has to be formalized*.

As observed above, operations can be of various types:

- **queries or read operations**, which change the knowledge of users,
- **update or write operations**, which change the state of stored data, and
- **authorization operations** (like grant or revoke) by which actors can grant or revoke other actors authorizations of operations.

If our framework is capable to express all the constraints we wish to formalize, then it would be of great help to have an automated reasoning tool that decides for a set of constraints expressed in this framework whether it is consistent or inconsistent. So, we are looking for a framework that on the one hand is expressive enough to encode all the desired constraints and on the other hand is restricted enough to be decidable or at least semi-decidable.

1.4 Our Approach

In this work we develop a logical framework that is capable to uniformly formalize semantic constraints and security constraints of information systems.

The framework consists of a computational model, a logical calculus the semantics of which is defined over the computational model and a tableau based proof system.

The computational model represents each actor and each object of an information system uniformly as agents. Each agent can perform operations and has her own operation alphabet. The data stored in objects is represented by propositions local to the respective agent. The factual knowledge of an actor is represented by propositions local to the respective agent.

We assume that each agent works autonomously of other agents and has her own local time line. Agents synchronize with other agents by jointly performing synchronization operations. Each agent has a local view on the system and a local state.

Two states of the system are equivalent for agent i if they do not differ in the local state of agent i, however, they can differ in the local state of some agent j other than i. This means, if the system reaches a state c' from a state c through an operation in which i has not participated, the states c and c' are equivalent for i.

To capture an agent's knowledge in the model we define an indistinguishability relation for knowledge for each agent i that describes for each state c, which states are indistinguishable from c for agent i. Similarly, to capture an agent's belief in the model, we define an indistinguishability relation for belief for each agent i.

Apart from the computational model, our framework contains a temporal and epistemic logic the semantics of which is defined on the described computational model. For each agent, the language provides local next operators labelled by operations, a local until operator, a local belief and a local knowledge operator.

The temporal part of the logical calculus is a lightweight version of that presented in [Nie97]. The key idea of the temporal part of the logic is that formulae "look" at the configurations from a local point of view of either a single agent or of a group of agents.

The epistemic part of the logical calculus is closely related to the work of Ramanujam presented in [Ram96]. We will discuss this in detail later in chapter 2. The key idea of the epistemic part of the logic is that knowledge of agents changes *due* to actions: if an agent does not participate in an action, her knowledge remains unaffected and also, the enabling of an action remains is independent of the knowledge of agents that do not participate in the action.

The third component of our unifying framework is a sound and complete tableau based proof system for a subset of this logic that does not contain belief operators. In the computational model, change of knowledge of an agent is only due to performance of operations by the agent. The enabling and the effect of an operation is dependent only on the knowledge of participating agents. This seems to be a very natural constraint. However, exactly this constraint makes the tableau system quite involved. Our tableau system will explicitly represent the temporal relations and epistemic indistinguishability relations in the tableau to have a kind of 'global view' on the whole system under construction.

1.5 Outline of the Thesis

In chapter 2 we try to get an overview over the various approaches of modal logics available so far for the specification of the various types of constraints and discuss in detail the approaches relevant for our framework.

In chapter 3 we describe our view of an information system. We define a computational model and show how this model represents our view of an information system.

In chapter 4 we formally define the syntax and semantics of the specification language and investigate some properties of the language.

Chapters 5 and 6 present the tableau based proof system for the logic and prove its soundness. The developed tableau system is not complete if we consider the full logic. However, we show that the tableau system is complete for a reasonable subset of the defined logic.

In chapter 7 we present a detailed example in which we show, how the developed framework can be used to uniformly specify the various types of constraints in an information system.

Finally, in chapter 8 we discuss open problems which we did not investigate in this thesis.

A part of this work, namely our view of an information system captured in the computational model in chapter 3 and the logical language defined in chapter 4 was published as [BS03]. At some points in this work, we use text passages from the mentioned publication, if we find it appropriate and do not see the need of a reformulation.

Chapter 2

Related Work

In the past a lot of work concerning formalization of database constraints has been done.

In the logical approach to information systems, it is assumed that the system administrator expresses the semantic constraints in a declarative style at design time. He specifies formulae in an appropriately restricted formal language of a logic calculus. The logic calculus is presumed to have a syntax for formulae and model-theoretic semantics, i.e., there exists a definition of *a formula is satisfied by some structure (or some interpretation)* and based on that a definition of *a set of formulae implies another formula*. In the context of information system, instances of an information system are considered as structures in the sense of the logic calculus.

In a policy oriented approach to security, a security administrator expresses the security constraints in a declarative style. He specifies some formulae in an appropriate formal language. It is assumed that besides a syntax for formulae, a semantics is provided too, i.e. a definition of *a formula is satisfied by a history* that comprises, possibly among others, a sequence of actual executions of operations, and based on that, of *a set of formulae implies another formula*.

In this section we analyze various approaches for the logical formalization of information systems found in the literature. While semantical constraints have been widely investigated, still a lot of work is required with regard to security constraints and in particular with regard to the combination of both types.

In the introduction we have identified aspects that should be met by the framework: The framework should be suited to model *states* of the information system, its *temporal behaviour* and its *epistemic behaviour* in a *distributed environment*, in which various components act autonomously and concurrently. In most papers only few aspects of security constraints were investigated, other aspects were faded out.

2.1 Temporal Aspects in Distributed Information Systems

Temporal logics have been successfully used for modeling temporal integrity constraints in information systems (see e.g. [CT98]): Updates and integrity constraints can be formulated in an abstract, representation-independent way. Chomicki and Toman give an overview of the development of temporal logics in information systems.

Information systems can be seen as open, reactive and usually distributed systems that administrate persistent data. Each component (object, user, administrator, etc.) carries out operations that change the state of the information system. It is especially interesting for our aim to look for temporal logics that allow to model distributed behaviour. A lot of work has been done in the field of modeling temporal behaviour in reactive systems and multi-agent systems (see for example [MP92], [Lam94], [LPRT93], [Wei99], [HS98], [Woo00]).

In the field of multi-agent systems a very common approach are BDI-models (*belief-desire-intention*-models), [Wei99], [HS98], [Woo00]. Agents are seen as entities that are capable of observing their environment and of reasoning about it and of independently and autonomously performing actions based upon decisions made by these agents dependent on their observations of the environment. In this perspective, each agent is assumed to have its own belief, desires and intentions. In [Woo00], Wooldridge introduces a very powerful first-order BDI-logic called *LORA* (Logic for Reasoning Agents). *LORA* consists of several components: a first-order component, a belief/desire/intention component and a temporal component. The temporal component is based on CTL^*, a well known temporal logic with an interleaving semantics.

Reactive systems can be seen a bit wider than multi-agent-systems. In reactive systems, not all components necessarily need to be agents in the sense described above.

A very common model for reactive systems are so-called Mazurkiewicz traces, refer [Maz95], [MOP89]. Mazurkiewicz traces are a semantic concept for modeling concurrency. Unlike other semantic concepts, e.g. Kripke structures, the main characteristic of such trace systems is to explicitly differentiate between concurrency and nondeterministic choice. Another main feature of trace systems is the abstraction of interleaving; modeling of true concurrency is possible. In a transition system, $\mathsf{a} \| \mathsf{b}$ (a parallel b) is modeled same as $\mathsf{a}, \mathsf{b} \Box \mathsf{b}, \mathsf{a}$ (nondeterministic choice between a followed by b and b followed by a). A trace system on the other hand differentiates between these two operations. Although this model is not more expressive than semantic concepts with interleaving, it can avoid the known (notorious) state explosion problem.

Temporal logics for distributed systems based on Mazurkiewicz-trace systems have been developed by Penczek, Thiagarajan, Lodaya, Ramanujam, Niebert and others [Thi94, Pen93, Nie97]. Thiagarajan develops TPTL (Trace based Propositional Temporal Logic) in [Thi93] to express nonsequential behaviour directly without using interleaving. Nonsequential behaviour occurs in reactive systems with multiple autonomous sequential agents. A more expressive logic was introduced by Niebert in [Nie95, Nie97]. In addition to the usual logical connectors, TPTL contains a local next and until, similar to the ones known from PTL (for a detailed description see [Thi93]). νTrPTL contains fixpoint operators instead of until, thus νTrPTL is strongly more expressive than TPTL.

TPTL as well as νTrPTL base upon traces seen as nonsequential runs of a distributed system. Such a system consists of an arbitrary but fixed number k of sequential agents which synchronize by performing actions jointly. Each agent i is assigned a non-empty local alphabet Σ_i of actions , $\tilde{\Sigma} = (\Sigma_1, \Sigma_2, \ldots, \Sigma_k)$, ($k$ = number of agents of the system) is called a *distributed alphabet*. Agent i must take part in each action $a \in \Sigma_i$. Thus synchronization between individual agents is modeled by the joint execution of actions. If two actions a and b are not contained in the same alphabet, they can be executed independently. Infinite traces can be seen as Σ-labelled partial orders (fulfilling certain characteristics), where $\Sigma = \bigcup_{i \in \{1,\ldots,k\}} \Sigma_i$.

[Thi93, NS97] prove the decidability of the satisfiability problem for TPTL as well as for νTrPTL.

Inspired by this development of logics for distributed systems, Ehrich, Caleiro, Sernadas and Denker presented in [ECSD98] two object-oriented logics capable of expressing communication among objects. In these logics, all objects of a database system are seen as sequential agents and are thus components of the distributed system.

2.2 Epistemic Aspects in Distributed Systems

There are several approaches in which a database is seen as a collection of knowledge (see e.g. [Rei90] or [CD96, CD97]). If we follow this view, we can uniformly see database objects, users, administrators etc. as reasoning agents in a distributed environment. Over the last decade, modelling of knowledge has been a field of great research interest, especially for multi-agent systems. [FHMV95] gives an excellent overview of the state of the art in this topic.

Traditionally, modal logics of knowledge are interpreted over global states of the distributed system. As motivated in the introduction, we cannot assume such a global state: Every agent only has a local view on the system and when the system changes due to an action by a group of agents, only agents of the acting group typically know the effect of that action. The knowledge of the non-participating agents remains unchanged, and this fact is known by all agents.

Ramanujam analyses in [Ram96, Ram94, Ram99] the meaning of knowledge in distributed systems. A similar discussion carried out by Van der Hoek and Meyer in [vdHM92] addresses the following questions:

1. What exactly is a knowledge state?
2. Is the knowledge that is logically implied by the available information actually computable?
3. Given a knowledge state, can we decide, which other states are reachable?
4. Can the actions of one agent have influence on the knowledge of another?

Ramanujam develops an action based temporal and epistemic logic ([Ram94]), in which knowledge changes caused by actions of agents can be expressed. This logic is a natural extension of PDL (Propositional Dynamic Logic) with actions enriched with the modal operator K as it is defined by Hintikka in the logic *S5*.

2.3 Deontic Aspects

We considered different types of constraints in the introduction. Security constraints in a database system always deal with obligations and authorizations. The tendency to formalize such constraints through deontic logic has been mainly followed by Cuppens, Demolombe, Carmo and Jones. In this method, a database is considered as a normative system and the corresponding security and semantic constraints are seen as deontic constraints: *It ought to be, that ..., It is necessary, that ..., It is permitted, that ...*. The meaning of a normative system in this context is, that a set of agents (software systems, humans etc.) interact according to some rules. It is not explicitly mentioned, that the misconduct of the agents is impossible. Normative systems rather suggest, how to handle the misconduct of an agent.

In these approaches, it is often distinguished between hard and soft deontic constraints (see [CJ94]). Hard deontic constraints are constraints, that may never be violated, such as *Every member of the department possesses an identification number*. Soft deontic constraints, e.g. *Books can only be issued to the members of the department*, must eventually be fulfilled, however, they might be violated for a certain period of time. Similar to this view is the distinction between ideal, sub-ideal and prohibited states of such a system. This method of classification of states was first introduced by [JP85]. Ideal states in databases are those situations, in which all constraints are fulfilled. In sub-ideal states, some weak constraints may be violated, but the system must still be transformable in an ideal state, i.e. eventually fulfill all constraints. Prohibited states correspond to situations which are not allowed to occur. Logics suggested in this context (e.g. in [CJ94]) mainly focus on the deontic aspects. Temporal features as we would need them for our framework are only very basic (mainly simple transitions between states), strong temporal components like *until* or *eventually* to capture larger periods of time are not provided.

In our view, authorizations for performing operations can be modeled by assigning particular rights to the components of the information system. Then we can formalize a temporal constraint, that roughly expresses the following: All operations that are not explicitly allowed for a component, cannot happen. We could also formalize a dual temporal constraint: All operations that are not explicitly forbidden, may eventually happen. These rights are not static in the system but may be changed by other components.

The distinction between soft deontic constraints (those, that must eventually be fulfilled) and hard deontic constraints (those, that must always be fulfilled) can also be done in a temporal way: Hard deontic constraints *always* have to be fulfilled, whereas soft deontic constraints may be violated at some states but must *eventually* be fulfilled.

As a consequence, in our approach deontic aspects are completely reduced to temporal aspects and the added feature of explicit rights, or explicit forbiddances, respectively.

Dignum et al have a somewhat different interpretation of deontic aspects in [DMWK96]. This paper deals with the question, which obligations for individual users of a database systems are born from the corresponding past of each user. The logic developed in this paper consists of temporal operators (directed into past) and deontic operators, dynamic (*ought-to-do*), as well as static (*ought-to-be*). Further, the logic contains one more modal operator, which formalizes the intention of the corresponding agent. This operator can be seen as a kind of *Next*-operator, which only allows actions, which are ideal in sense of intended actions, as next actions.

2.4 Combination of Deontic and Epistemic Aspects

We look back to the epistemic aspects of security constraints. There are a lot of works, mainly by Reiter [Rei90], Cuppens and Demolombe [CD96, CD97], but also by MacEwen [GMP92, MCK96] in which a database is seen as a collection of knowledge: the database *knows* or *believes in* a set of facts about the real world, users of the database get to know the parts of these facts by querying the database. In association with inference control, the question arises, which knowledge is allowed to be accumulated by a user? What is he unauthorized to know? In ([CD97]), Cuppens and Demolombe define a modal logic with epistemic and deontic operators and based on Kripke structures. They show, that this logic is axiomatisable. It can be expressed in this logic, which contents of a database a user knows ($\mathsf{KB}_\mathtt{i}$), or may know ($\mathsf{PKB}_\mathtt{i}$) as well as which contents a user in a particular role may or may not know ($\mathsf{PKB}_\mathtt{r}, \mathsf{FKB}_\mathtt{r}$).

The knowledge, the prohibitions or permissions to know are always related to the facts in the database. Knowledge about the knowledge of other users or the knowledge about the actions performed in the database is however not formulatable. Changes in the database, which lead to changes of knowledge of the users, were also disregarded. It is assumed, that the contents of the database are fixed. In our context we must assume, that users can gain knowledge not only about the data stored in the information system but also about the behaviour of the system itself as well as of the knowledge (or belief) of other users.

Chapter 3

Our View of an Information System

In this chapter, we capture our view of an information system in a computational model for the temporal and epistemic logic defined in the next section. The model consists of three aspects: static aspects, dynamic aspects concerning control and dynamic aspects concerning knowledge and belief. In the first two aspects we roughly follow the definitions in [Nie97] of a partial order model for a reactive system. We then extend the model by an epistemic component. The epistemic definitions are more or less standard: Situations (worlds) are related by indistinguishability relations. To capture interaction between knowledge and time, we define the indistinguishability relations dependent of the temporal definitions.

3.1 Static Aspects

We view a distributed information system as consisting of several components, e.g. data objects, users, administrators, and security managers. If the information system is viewed as a piece of software, then data objects lie inside the system whereas users, administrators, etc. appear outside the system. If we provide a representative user-agent inside the system for each outside actor, and a repository-agent for each data object, then we can model all the components of an information system uniformly as sequential agents. We call the finite set of all k agents of an information system Ag. That is,

$$Ag := \{1, \dots, k\}.$$

The current state of each object, i.e. its data content, is represented by a set of propositions local to the corresponding repository-agents. Similarly, the current state of a user, i.e. what data she has read or which rights have been granted to her, is represented by a set of propositions local to the corresponding user-agent.

Each agent i of our system is thus associated with a finite set of local propositions $\mathcal{P}_i$. Let

$$\tilde{\mathcal{P}} := (\mathcal{P}_1, \dots, \mathcal{P}_k)$$

be the distributed set of propositions, such that $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ for $i \neq j$. Further, we denote with $\mathcal{P} = \bigcup_{i \in Ag} \mathcal{P}_i$ the set of all propositions.

All agents can perform *operations* (read, insert, delete, grant rights, revoke rights, etc.) partly autonomously, partly together with other agents as joint operations.

Seen as a whole system, all these components work concurrently: E.g. while actor a_1 inserts some value x in object o, another actor a_2 could at the same time (concurrently) grant some execute right r to a third actor a_3. However, each single component performs its operations sequentially.

Thus, we equip each agent i with her own finite, non-empty operation alphabet $\mathcal{O}_i$. Let

$$\tilde{\mathcal{O}} = (\mathcal{O}_1, \dots, \mathcal{O}_k)$$

be a distributed set of operations, and $\mathcal{O} = \bigcup_{i \in Ag} \mathcal{O}_i$.

With $ag(op) := \{i \in Ag | op \in \mathcal{O}_i\}$ we refer to the set of agents which are involved in the execution of a joint operation op. An operation $op \in \mathcal{O}_i \cap \mathcal{O}_j$ is called a *synchronization* operation between agents i and j. Two operations op_1 and op_2 are called *independent* iff $ag(op_1) \cap ag(op_2) = \emptyset$.

The informal meaning of these operations will be represented by the changes of the interpretation (see definition 3.3.4 below) of local propositions.

Summarizing the discussion above, we see the static declarations of a distributed information system as a concurrent system of sequential agents, where each agent is equipped with a set of operations and a set of local propositions.

Definition 3.1.1 (static declarations of an information system) *Let the static declarations of an information system be defined as a tuple* $(Ag, \tilde{\mathcal{O}}, \tilde{\mathcal{P}})$*, consisting of a set of agents, their distributed set of operations and their distributed set of local propositions.*

3.2 Dynamic Aspects Concerning Control

Given the static declarations of an information system, we now describe the control flow of its possible dynamic behaviours. Below, this control flow is formalized as runs. Each occurrence of an operation within a behaviour is denoted by an event. Thus a possible behaviour is captured by a set of events which should satisfy some requirements in order to represent a reasonable control flow.

- It should be finite or at most denumerably infinite.

- It should be partially ordered according to the relative occurrence of events in time.
- It should distinguish some specific events as the explicit beginning of a behaviour. More precisely, each single event should have only a finite set of predecessors according to the partial order.
- At each event, exactly one operation occurs.
- For each agent i, the set of events this agent is involved in is even totally ordered according to the relative sequence in time.

Every occurrence of an operation in the control flow of a possible behaviour of an information system is an event. The set of all events of a run is partially ordered. And since we have an explicit beginning of the run of such an information system, the downward closure of each subset of the set of events of a run is finite.

Definition 3.2.1 (downward closure) *Let E be a set and $\leq \subseteq E \times E$ be a partial order on E. For $M \subseteq E$,*

$$\downarrow M := \{e \in E \mid \exists e' \in M : e \leq e'\}$$

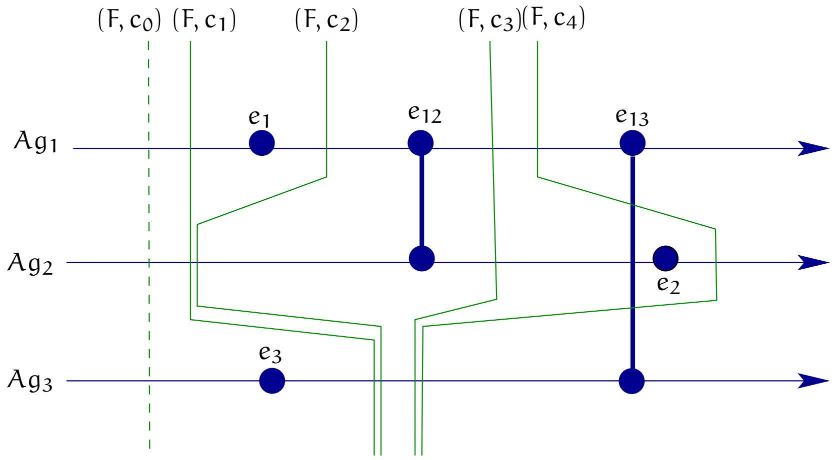
denotes the downward closure *of M. For $e' \in E$ we write $\downarrow e'$ instead of $\downarrow \{e'\}$.*

Now we can formally define the control flow of a possible behaviour of an information system as a run.

Definition 3.2.2 (run) *A* run $F_{(Ag,\tilde{\mathcal{O}},\tilde{\mathcal{P}})} = (E, \leq, \lambda)$ *of an information system with the static declarations $(Ag, \tilde{\mathcal{O}}, \tilde{\mathcal{P}})$ is a partially ordered, labelled set of events E, s.t. the following holds:*

- *E is finite or denumerably infinite.*
- *$\leq$ is a partial order on E.*
- *For all $e \in E$ the downward closure $\downarrow e$ is finite.*
- *$\lambda : E \longrightarrow \mathcal{O}$ is a labelling function yielding the operation $\lambda(e)$ occurred at event e.*
- *For all $i \in Ag$, the reduction of $\leq$ on $E_i := \{e \in E \mid \lambda(e) \in \mathcal{O}_i\}$, i.e. $\leq \cap (E_i \times E_i)$, is a total order.*

We define $\mathcal{F}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$ *as the set of all possible runs* $\mathsf{F}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$ *over the same static declarations* $(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})$*. We write* F *instead of* $\mathsf{F}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$ *and* $\mathcal{F}$ *instead of* $\mathcal{F}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$ *where this does not lead to misunderstandings.*



Examples of configurations:

$c_0 = \emptyset$, $\quad c_1 = \{e_3\}$, $\quad c_2 = \{e_1, e_3\}$, $\quad c_3 = \{e_1, e_3, e_{12}\}$,
$c_4 = \{e_1, e_3, e_{12}, e_2\}$

Ag_3's view of configuration c_4: $\downarrow^{ag_3} c_4 = c_1 = \{e_3\}$
$c_1 \equiv_{\mathsf{Ag}_3} c_2 \equiv_{\mathsf{Ag}_3} c_3 \equiv_{\mathsf{Ag}_3} c_4$

Figure 3.1: Example of a run and local configurations

See figure 3.1 for an illustration of a possible run: The example run is executed by three agents $\mathsf{Ag}_1, \mathsf{Ag}_2$ and Ag_3. Each agent is represented by her own (horizontal) time line. The agents perform operations sequentially: Ag_1 performs the sequence of operations $\mathsf{op}_1, \mathsf{op}_{12}, \mathsf{op}_{13}$. Agent Ag_2 performs the sequence of operations $\mathsf{op}_{12}, \mathsf{op}_2$. And agent Ag_3 performs the sequence of operations $\mathsf{op}_3, \mathsf{op}_{13}$. Each event, i.e. occurrence of an operation, is represented by a vertical bar with circles for the participating agents. Though the events for each agent are totally ordered, the set of all events of the system are only partially ordered: for example, operations op_1 and op_3 are performed concurrently, no order is given between them.

A configuration c of a run F is a downward closed set of events, it contains all events that are performed by agents up to a particular point in time. Each agent only has a local view on the system. The local view on a configuration c of a run F for an agent i is itself a configuration and consists of the downward closure of all events that agent i has performed in configuration c. It is thus the least configuration which coincides with c on all i-events.

Formally, we define a configuration as follows:

Definition 3.2.3 ((local) configuration; i-view) *Let $F = (E, \leq, \lambda)$ be a run. A* configuration c *of F is a finite, downward closed set of events (i.e. $c \subseteq E$ and $\downarrow c = c$). Let $\mathcal{C}_F$ denote the set of all configurations of run F.*

The localization of a configuration $\downarrow^i c := \downarrow (c \cap E_i)$ *is called the* i-view *of configuration c, i.e. the* i-view *is the least configuration that coincides with c on all i-events.*

Two configurations c, c' of F are said to be i-equivalent *($c \equiv_i c'$), iff $\downarrow^i c = \downarrow^i c'$.*

They are said to be A-equivalent *($c \equiv_A c'$), iff they are i-equivalent for each $i \in A$.*

On configurations of F we define a successor relation so that $c \xrightarrow{op} c'$ iff $c' = c \cup \{e\}$ for some $e \in E$ with $\lambda(e) = op$.

Two configurations of the same run that differ from each other only through events in which agent i does not participate are equivalent from the point of view of agent i. We call such configurations i-equivalent.

If a configurations c becomes a configuration d by execution of an operation op and a configuration c' becomes configuration d' by execution of an operation op, and if $c \equiv_i c'$ for at least one $i \in ag(op)$, then operation op is a synchronization operation for all agents $j \in ag(op)$ and thus, the resulting configurations d and d' are j-equivalent ($d \equiv_j d'$) for all $j \in ag(op)$.

Lemma 3.2.4 *Let $c, c', d, d' \in \mathcal{C}_F$ be configurations of the same run F, and let op be an operation. It holds that*
if $c \equiv_i c'$ for any $i \in ag(op)$ and $c \xrightarrow{op} d$ and $c' \xrightarrow{op} d'$
then $d \equiv_j d'$ for all agents $j \in ag(op)$.

Proof:

Let F be a run and let $c, c', d, d' \in \mathcal{C}_F$ be configurations of this run. Let op be an operation. $c \equiv_i c'$ for any agent $i \in ag(op)$ and $c \xrightarrow{op} d$ and $c' \xrightarrow{op} d'$.

$$
\begin{array}{ll}
& c \equiv_i c' \text{ for any agent } i \in ag(op) \text{ and } c \xrightarrow{op} d \text{ and } c' \xrightarrow{op} d'. \\
\Longrightarrow & \downarrow(c \cap E_i) = \downarrow(c' \cap E_i) \text{ and there exist events } e, e' \in E \text{ with} \\
& c \cup \{e\} = d \text{ with } \lambda(e) = op \text{ and} \\
& c' \cup \{e'\} = d' \text{ with } \lambda(e') = op \\
\Longrightarrow & (\text{* by definition 3.2.2 item (5) *}) \\
& \downarrow(c \cap E_i) = \downarrow(c' \cap E_i) \text{ and there exists an event } e \in E_i \text{ with} \\
& c \cup \{e\} = d \text{ and } c' \cup \{e\} = d' \text{ with } \lambda(e) = op \\
\Longrightarrow & (\text{* by definition 3.2.3 } d \text{ and } d' \text{ are downward closed *}) \\
& c \equiv_j c' \text{ for all } j \in ag(op) \\
\Longrightarrow & (\text{* } e \in E_j \text{ for all } j \in Ag \text{ *}) \\
& d \equiv_j d' \text{ for all agents } j \in ag(op)
\end{array}
$$

□

Consider again figure 3.1. A *configuration* is indicated by a (more or less vertical) line that crosses each agent's time line at a box. The configuration contains all events to the left of the line. The initial configuration $c_0 = \emptyset$ contains no events. In the figure it is denoted by a dashed line. The configurations c_1, c_2, c_3 and c_4 are equivalent from agent Ag_3's point of view, which is illustrated by the solid lines through agent Ag_3's second box. In each of these configurations, agent Ag_3 has performed only event e_3, the configurations differ only in events performed by other agents. Configuration c_1 represents the view of agent Ag_3 on all the configurations c_1, c_2, c_3 and c_4.

Sometimes, we need to talk about configurations of different runs. To make sure, which configuration of which run we mean, we have to name both the configuration and the run to which it belongs.

Definition 3.2.5 (situation) *We call* (F, c) *a* situation*, when* $F = (E, \leq, \lambda)$ *is a run and* $c \in \mathcal{C}_F$ *is a configuration of this run.*

On situations we define a successor relation so that $(F, c) \xrightarrow{op} (F, c')$ *iff* $c \xrightarrow{op} c'$ *for* $c, c' \in \mathcal{C}_F$.

We define $\downarrow^i (F, c)$ *as the localization of a situation* (F, c) *iff* $\downarrow^i (F, c) = (F, c')$ *and* $c' = \downarrow^i c$.

Two situations (F, c) *and* (F, c') *are called* i*-equivalent* $((F, c) \equiv_i (F, c'))$ *iff* $c \equiv_i c'$.

For a set of runs $\mathcal{A}$ *we define* $|\mathcal{A}| := \{(F, c) \mid F \in \mathcal{A}, c \in \mathcal{C}_F\}$ *to be the set of all situations of all runs in* $\mathcal{A}$.

Note, that our definition of situation mainly describes progress in time and leaves propositions still uninterpreted.

3.3 Dynamic Aspects Concerning Knowledge and Belief

In order to capture the agent's knowledge and belief, we need a more general notion of "indistinguishability" than is defined above.

An agent does not know the actual current configuration of the actual run of the system. At all times, an agent i only sees a partial system, e.g. it can notice the behaviour of other agents only via synchronization operations, other operations of other agents are independent of and invisible for agent i. Apart from that, the agent is not even aware of the actual run, she for example cannot distinguish between the actual configuration of the actual run and a configuration of a different run, in which she has acquired the same knowledge. Though we assume that an agent knows, which operations other agents may in principle perform, i.e. though the agent is aware about other agents' operation alphabet, she does not know which operations the other agents actually perform. Further, we do not require that the agents have perfect recall, which means, we do not assume that each agent always remembers all the operations she has already performed and always remembers all the knowledge, she has already acquired. We allow that an agent i considers two situations as indistinguishable, though they have a different history local to agent i. The agent does not necessarily remember the *way* how she acquired some knowledge, she only knows, *that* she has acquired the knowledge.

As already indicated above, since each agent only has a partial view on the whole system, she does not know all facts that are true. As in the epistemic logic $\mathsf{S5}$ we require, however, that if an agent knows a fact, then this fact must be true. This property is often called the *Knowledge Axiom* or the *Truth Axiom*.

Further, as in the epistemic logic $\mathsf{S5}$ we require that agents can do introspection regarding their knowledge. An agent should know, what she knows (this is typically called *Positive Introspection*) and she should know, what she does not know (typically called *Negative Introspection*).

Later, in chapter 4.2 we define that *an agent knows a fact* if this fact is true in all situations indistinguishable for this agent. (In terms of standard modal logics such situations are called *indistinguishable*, the corresponding relation is called *indistinguishability relation.*)

We now define an indistinguishability relation R_i^k for knowledge for each agent $i \in Ag$. In the definition of the indistinguishability relation we follow the standard approach of the modal logic $\mathsf{S5}$ ($\mathsf{KT45}$). As for example shown in [FHMV95], each of the above requirements is directly reflected by properties of the indistinguishability relation: Reflexivity of the indistinguishability relation reflects the truth axiom, transitivity of the indistinguishability relation reflects the positive introspection axiom and the Euclidean property reflects the negative introspection axiom. Hence, we require each relation R_i^K to be an equivalence relation:

- reflexive: the agent knows only true facts (truth axiom).

  The relation R_i^K is reflexive, iff for every situation (F, c) with $F \in \mathcal{A}$ and $c \in \mathcal{C}_F$, it holds that $((F, c), (F, c)) \in R_i^K$.

- transitive: the agent knows, what she knows (positive introspection).

  The relation R_i^K is transitive, iff for all situations $(F_1, c_1), (F_2, c_2), (F_3, c_3)$ with $F_1, F_2, F_3 \in \mathcal{A}$ and $c_1 \in F_1, c_2 \in F_2, c_3 \in F_3$ it holds that if $((F_1, c_1), (F_2, c_2)) \in R_i^K$ and $(F_2, c_2), (F_3, c_3)) \in R_i^K$ then $((F_1, c_1), (F_3, c_3)) \in R_i^K$.

- Euclidean: the agent knows, what she does not know (negative introspection).

  The relation R_i^K is Euclidean, iff for all situations $(F_1, c_1), (F_2, c_2), (F_3, c_3)$ with $F_1, F_2, F_3 \in \mathcal{A}$ and $c_1 \in F_1, c_2 \in F_2, c_3 \in F_3$ it holds that if $((F_1, c_1), (F_2, c_2)) \in R_i^K$ and $((F_1, c_1), (F_3, c_3)) \in R_i^K$ then $((F_2, c_2), (F_3, c_3)) \in R_i^K$.

The standard requirement for an equivalence relation is that it is reflexive, transitive and symmetric. However, a relation is reflexive, transitive and Euclidean if and only if it is reflexive, transitive and symmetric:

Proposition 3.3.1 *Let S be a set. A relation $R \subseteq S \times S$ is reflexive, transitive and Euclidean if and only if it is reflexive, transitive and symmetric.*

Proof:

$\Longrightarrow$: Let R be Euclidean, reflexive and transitive.
Consider $x, y \in S$ and suppose, $R(x, y)$.
Because the relation R is reflexive, we have $R(x, x)$. Then, because R is Euclidean, we also have $R(y, x)$, i.e. R is symmetric.

$\Longleftarrow$: Let R be symmetric, transitive and reflexive.
Consider $x, y, z \in S$ and suppose, $R(x, y)$ and $R(x, z)$.
Because the relation R is symmetric, we have $R(y, x)$. Then, because R is transitive, we also have $R(y, z)$, i.e. R is Euclidean.

$\square$

Whenever agents perform an operation, we assume that at least one of the acting agents is aware of the operation. We thus require that within a run two different configurations must be distinguishable for at least one agent.

Further we want that in a run an agent i considers all configurations as possible that are i-equivalent to the actual configuration. The difference between two i-equivalent configurations can only be a set of operations executed by an agent other than i and thus invisible to agent i. Note, that this implies that agent i must consider all operations of other agents as possible, which means, agent i must know that these operations belong to the other agents' operation alphabet.

Finally, we assume that changes of knowledge are *due* to operations. We require that the "enabling" of an operation as well as the effect of that operation that is, the changes in the truth values of local propositions (see definition 3.3.4 below), are only dependent on the knowledge of the acting agents and not on any outside factor. Additionally, the execution of an operation by a group of agents has no effect on the knowledge of other agents that are not participating in this operation. This requirement is met by item (4) of the following definition.

Definition 3.3.2 (indistinguishability relations (for knowledge)) *Let $\mathcal{A} \subseteq \mathcal{F}$ be a set of runs.*

For each agent $i \in Ag$ we define her indistinguishability relation for knowledge

$$R_i^K \subseteq \{((F, c), (F', c')) \mid F, F' \in \mathcal{A}, c \in \mathcal{C}_F, c' \in \mathcal{C}_{F'}\}$$

such that the following properties hold:

1. *R_i^K is an equivalence relation.*
2. *If $((F, c), (F, c')) \in R_i^K$ for all $i \in Ag$ then $c = c'$.*
3. *If for two situations $(F, c), (F, c') \in |\mathcal{A}|$ we have that $(F, c) \equiv_i (F, c')$ then $((F, c), (F, c')) \in R_i^K$.*
4. *If for two situations $(F, c_1), (F, c_2) \in |\mathcal{A}|$ holds, that $(F, c_1) \xrightarrow{op} (F, c_2)$ and there exists $(F', c_1') \in |\mathcal{A}|$ such that for every $i \in ag(op)$ it holds that $((F, c_1), (F', c_1')) \in R_i^K$ then there exists $(F', c_2') \in |\mathcal{A}|$ such that $(F', c_1') \xrightarrow{op} (F', c_2')$ and $((F, c_2), (F', c_2')) \in R_i^K$ for every $i \in ag(op)$.*

Sometimes it might be more appropriate not to talk about *knowledge* but about *belief* of an agent. For example, we might want to model that some agent i "trusts" some other agent j or that agent i "believes" that agent j follows some protocol faithfully. Often, an agent's behaviour is dependent of her belief about other agents. However, it might well be that the agent's belief is false. In the literature, *knowledge* is often seen as *true belief*: Although one might have false beliefs, one cannot know something that is false. However, as in the standard logic for belief, KD45, we require that an agents belief is at least not contradictory.

To model belief of an agent, we want to weaken the conditions of the indistinguishability relation in the sense that we no longer require the indistinguishability relation to be reflexive but serial, i.e. in each situation each agent must consider some situation as possible.

Following the standard approach of the modal logic KD45, the indistinguishability relation R_i^B for belief should be

- serial: the belief of the agent may not be contradictory.

  The relation R_i^B is serial, iff for all situations (F, c) there exists some situation (F', c') such that $((F, c), (F', c')) \in R_i^B$.

- transitive: the agent believes, what she believes (positive introspection).

  The relation R_i^B is transitive, iff for all situations $(F_1, c_1), (F_2, c_2), (F_3, c_3)$ it holds that if $((F_1, c_1), (F_2, c_2)) \in R_i^B$ and $(F_2, c_2), (F_3, c_3)) \in R_i^B$ then $((F_1, c_1), (F_3, c_3)) \in R_i^B$.

- Euclidean: the agent believes, what she does not believe (negative introspection).

  The relation R_i^B is Euclidean, iff for all situations $(F_1, c_1), (F_2, c_2), (F_3, c_3) \in |\mathcal{A}|$ it holds that if $((F_1, c_1), (F_2, c_2)) \in R_i^B$ and $((F_1, c_1), (F_3, c_3)) \in R_i^B$ then $((F_2, c_2), (F_3, c_3)) \in R_i^B$.

We require that an agent i has the same belief in situations (F, c) and (F, c') that are equivalent from her point of view $((F, c) \equiv_i (F, c'))$. This means that each situation that is R_i^B-indistinguishable from (F, c) is also R_i^B-indistinguishable from (F, c').

Further we require that each agent at least believes, what she knows. That is, we require that the indistinguishability relation for belief must be a subset of the indistinguishability relation for knowledge.

These observations lead to the following definition:

Definition 3.3.3 (indistinguishability relations (for belief)) *Let $\mathcal{A} \subseteq \mathcal{F}$ be a set of runs.*

For each agent $i \in Ag$ we define her indistinguishability relation for belief

$$R_i^B \subseteq \{((F, c), (F', c')) \mid F, F' \in \mathcal{A}, c \in \mathcal{C}_F, c' \in \mathcal{C}_{F'}\}$$

such that the following properties hold:

1.
   - R_i^B *is serial,*
   - R_i^B *is transitive,*
   - R_i^B *is Euclidean.*

2. *If* $(F, c) \equiv_i (F, c')$ *then for all* $(\bar{F}, \bar{c}) \in |\mathcal{A}|$ *we have that* $((F, c), (\bar{F}, \bar{c})) \in R_i^B$ *iff* $((F, c'), (\bar{F}, \bar{c})) \in R_i^B$.

3. *Each relation* R_i^B *is a subset of the corresponding indistinguishability relation for knowledge* R_i^K*:* $R_i^B \subseteq R_i^K$.

All agents are equipped with a set of local propositions. Agents that represent data objects in an information system can be seen as repository-agents: They are equipped with a set of local propositions the interpretation of which reflects their current content. User-agents are equipped with a set of local propositions the interpretation of which is something like "the information this user currently holds", e.g. as a printout on her desk.

More formally, treating the notion of *current* by situations, we consider an interpretation as mapping each pair (situation, local proposition) to a truth-value.

The content (or state) of repository-agents and the information of user-agents can be changed via operations, e.g. if a user inserts some fact into an object, then after this operation the local proposition that represents this fact for her must be true. If a user-agent reads some fact from the information system, then after the read operation, the local proposition that represents this fact for her must be true. If a user-agent grants another user-agent the right to do something, then after the grant operation, the local proposition that represents the right for the other user-agent must be true.

The informal meaning of operations will be represented by the changes of the interpretation of local propositions:

Definition 3.3.4 (interpretation) *An* interpretation of a set of runs $\mathcal{A} \subseteq \mathcal{F}$ *is a mapping* $\mathcal{I} : \{(F, c) \mid F \in \mathcal{A}, c \in \mathcal{C}_F\} \times \mathcal{P} \longrightarrow \{\top, \bot\}$*, such that if* $p \in \mathcal{P}_i$ *and* $((F, c), (F', c')) \in R_i^K$ *then* $\mathcal{I}((F, c), p) = \mathcal{I}((F', c'), p)$.

In definition 3.3.4 we have the requirement that in two situations that are indistinguishable for an agent i the interpretation of all propositions local to this agent must be the same. In terms of our information system framework this means that each user-agent is aware of all information she holds (on her desk) and all repository-agents are aware of their content.

3.4 Summarized View

In section 3.1 we have defined the static part of an information system to consist of a set of agents, their distributed set of propositions and their distributed set of operations. In section 3.2 we defined partially ordered runs as an underlying structure for dynamic aspects. In a run, each agent performs a sequence of operations. Some of the operations she performs alone, others she performs together with other agents, so that the overall structure for control flow is a partial order. Finally, in section 3.3 we introduced a notion of indistinguishable situations. These indistinguishable situations together with interpretations are the basis for internal states of agents, their knowledge and their belief.

This summarized view can be represented like in the example of figure 3.2. Figure 3.2 is an extension of figure 3.1. Both figures contain the same temporal model. Additionally, figure 3.2 represents a part of the indistinguishability relation of agent Ag_3. Since Ag_3 does not only consider the actual run F as possible but also other runs, the figure shows runs F' and F'' next to the actual run.

An agent's current state is represented by a box on the agent's time-line. The truth-value of a proposition local to an agent can change when this agent performs an operation: for example, before agent Ag_2 performs event e_2, the value of proposition p_2 is false. After event e_2 has happened, the value of proposition p_2 is true. Since agent Ag_3 does not participate in event e_2, the local propositions of agent Ag_3 do not change by event e_2.

An agent i *knows* a fact if the fact is true in all situations agent i considers to be possible. Let for example the local view of agent Ag_3 on the actual situation be (F, c_1). Agent Ag_3 considers all Ag_3-equivalent configurations as possible. Further, she considers some configurations of runs F', F'' as possible as well.

We now combine all these aspects to define a model of an information system.

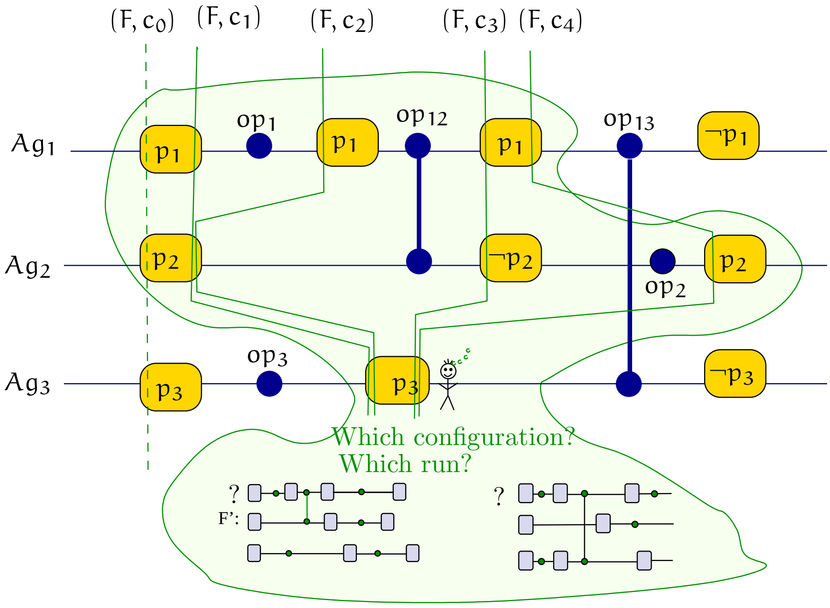


Figure 3.2: Illustration of an example model

Definition 3.4.1 (model) *A* model $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{R}^B, \mathcal{I})$ *of an information system is a tuple consisting of*

- *a set $\mathcal{A}$ of runs over a fixed static part of an information system $(\mathsf{Ag}, \tilde{\mathcal{O}}, \tilde{\mathcal{P}})$,*
- *a family $\mathcal{R}^K$ consisting of an indistinguishability relation for knowledge R_i^K for each agent $i \in \mathsf{Ag}$,*
- *a family $\mathcal{R}^B$ consisting of an indistinguishability relation for belief R_i^B for each agent $i \in \mathsf{Ag}$, and*
- *an interpretation $\mathcal{I}$ of $\mathcal{A}$.*

The set of all situations of a model $\mathcal{M}$ is denoted as $|\mathcal{M}|$.

Chapter 4

A Temporal and Epistemic Logic $\mathcal{L}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$

In this chapter we introduce an action based temporal and epistemic logic for a distributed system. Basically, the logic consists of three types of operators:

- the usual logical connectives $\neg$ and $\vee$
- a local next operator $\langle \mathtt{op} \rangle_i$ and a local until operator $\mathcal{U}_i$ for each agent as temporal operators and
- a local knowledge operator K_i as well as a belief operator B_i for each agent as epistemic operators.

The temporal part of the language follows the definitions of Niebert ([Nie97]) in a simplified way. The epistemic part of the language is closely related to the logic introduced by Ramanujam in [Ram96].

4.1 Syntax of the logic $\mathcal{L}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$

The key idea of our logic for the specification constraints of information systems is that the formulae "look" at the configurations from a local point of view. Some formulae look at the configuration of a run from the point of view of a single agent (e.g. in the case of local propositions or epistemic operators), others may involve a joint look from several agents (e.g. after a joint action of these agents). This idea is reflected in the syntax by a *family* of sets of formulae Φ_A where $A \subseteq \mathsf{Ag}$ is a set of agents. Formulae $\phi \in \Phi_A$ are called of *type* A: $\mathtt{type}(\phi) = A$. Note, that every formula has exactly one type, i.e. for two sets of agents $A, B \subseteq \mathsf{Ag}$ with $A \neq B$ we have $\Phi_A \cap \Phi_B = \emptyset$.
As we will see in the following definition, these types are used for the syntactic restriction of the construction of temporal formulae.

Definition 4.1.1 (Syntax of typed formulae of $\mathcal{L}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$) *Given a fixed static part of an information system $(\mathsf{Ag}, \tilde{\mathcal{O}}, \tilde{\mathcal{P}})$ as defined in section 3.*

We define a family of formulae, such that for each set $A \subseteq \mathsf{Ag}$ *the set* Φ_A *is the least set satisfying the following properties:*

$$
\begin{aligned}
&\textbf{propositions} \\
&\qquad \bot, \top \in \Phi_\emptyset \\
&\qquad \textit{for } p \in \mathcal{P}_i \textit{ we have } p \in \Phi_{\{i\}} \\
&\textbf{logical connectives} \\
&\qquad \phi \in \Phi_A \textit{ implies } \neg\phi \in \Phi_A \\
&\qquad \phi \in \Phi_A, \psi \in \Phi_B \textit{ implies } \phi \vee \psi \in \Phi_{A \cup B} \\
&\textbf{temporal operators} \\
&\qquad i \in \mathsf{ag}(\mathsf{op}), A \subseteq \mathsf{ag}(\mathsf{op}), \phi \in \Phi_A \textit{ implies } \langle \mathsf{op} \rangle_i \phi \in \Phi_{\{i\}} \\
&\qquad \phi, \psi \in \Phi_A, A \subseteq \{i\} \textit{ implies } \phi\, \mathcal{U}_i \psi \in \Phi_{\{i\}} \\
&\textbf{epistemic operators} \\
&\qquad K_i \phi \in \Phi_{\{i\}} \\
&\qquad B_i \phi \in \Phi_{\{i\}}
\end{aligned}
$$

We define the language $\mathcal{L}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$ *as the union of all sets* Φ_A *with* $A \subseteq \mathsf{Ag}$*:*

$$\mathcal{L}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})} := \bigcup_{A \subseteq \mathsf{Ag}} \Phi_A$$

In the following, we will write $\mathcal{L}$ *instead of* $\mathcal{L}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$ *where this does not lead to misunderstandings.*

Note: In the following we will write ϕ_i instead of $\phi_{\{i\}}$ where appropriate.

Definition 4.1.2 (type of a formula) *Let* $\phi \in \mathcal{L}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$ *be a formula. We define the* type *of* ϕ *as follows:*

$$\mathsf{type}(\phi) := A \textit{ for } \phi \in \Phi_A$$

Note that for $A, B \subseteq \mathsf{Ag}$ with $A \neq B$ we have that $\Phi_A \cap \Phi_B = \emptyset$, i.e.,every formula has a *unique type*.

The intuitive meaning of a temporal formula $\langle \mathsf{op} \rangle_i \phi$ is that from agent i's point of view the next operation is op and after the execution of op, ϕ holds. There is a syntactic restriction on the construction of such temporal formulae: within a formula of form $\langle op \rangle_i \phi$ (next-modality) the sub formula ϕ may only refer to agents that participate in operation op. Also, agent i has to participate in operation op, i.e. changing of the point of view of a temporal next-formula is only possible via a joint operation of the "old" (before the execution of the operation) and the "new" point of view (after execution of the operation). Consider for example a set of agents $Ag = \{ag_1, ag_2\}$, with $\mathcal{P}_{ag_1} = \{p_1\}$, $\mathcal{P}_{ag_2} = \{p_2\}$ and with $\mathcal{O}_{ag_1} = \{o_1, o_{12}\}, \mathcal{O}_{ag_2} = \{o_{12}\}$. Then the formula $\langle o_{12} \rangle_{ag_1} p_2$ is well-formed whereas the formula $\langle o_1 \rangle_{ag_1} p_2$ is not. In the first formula the view is changed from agent ag_1 to agent ag_2 via a common operation o_{12}, whereas in the latter formula the view is changed from agent ag_1 to agent ag_2 via operation o_1. However, o_1 is only local to agent ag_1 and after the execution of o_1 the local configuration of ag_2 could result from several configurations differing for agent ag_1.

Without this restriction, these temporal formulae could lead to "uncontrolled jumps into the past". (Further explanations are given in [Nie97]). A temporal formula of kind $\phi \mathcal{U}_i \psi$ intuitively means that either ψ holds already in the *current* configuration or the sub formula ϕ holds in the *current* configuration and in all *following* configurations *until* at some point of time ψ holds. (In this context, *until* can be seen as a *strong* until: At some point in the future ψ will hold.) The intuitive meaning of this until operator is quite standard, however note, that in our logic we have imposed the syntactic restriction, that both the formulae ϕ and ψ have to be local to agent i.

The epistemic operators are quite standard again. The intuitive meaning of $K_i \phi$ is that *agent* i *knows* ϕ, i.e. in all situations, that agent i considers to be possible ϕ holds, in particular we have that ϕ holds in the current configuration. The intuitive meaning of $B_i \phi$ is similar, i.e. again ϕ holds in all situations agent i considers to be possible with respect to belief. However, when talking about belief, agent i need not consider the actual configuration to be possible.

4.2 Semantics of the logic $\mathcal{L}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$

We will now give the formal semantics of the logic introduced above.

Definition 4.2.1 (semantics) *Given a model $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{R}^B, \mathcal{I})$, a run $F \in \mathcal{A}$ and a configuration $c \in \mathcal{C}_F$ of run F. The semantics of a formula ϕ is then inductively* *defined by:*

$$
\begin{array}{rl}
(F,c) \models_{\mathcal{M}} \top & \\
(F,c) \not\models_{\mathcal{M}} \bot & \\
(F,c) \models_{\mathcal{M}} p & \text{:iff } \mathcal{I}((F,c),p) = \mathsf{true} \\
(F,c) \models_{\mathcal{M}} \neg\phi & \text{:iff } \textit{not } (F,c) \models_{\mathcal{M}} \phi \\
(F,c) \models_{\mathcal{M}} \phi \vee \psi & \text{:iff } (F,c) \models_{\mathcal{M}} \phi \textit{ or } (F,c) \models_{\mathcal{M}} \psi \\
(F,c) \models_{\mathcal{M}} \langle op\rangle_i\phi & \text{:iff } \textit{there exists } c', r \in \mathcal{C}_F, \textit{ such that} \\
 & \quad c \equiv_i c' \textit{ and } c' \xrightarrow{op} r \textit{ and } (F,r) \models_{\mathcal{M}} \phi \\
(F,c) \models_{\mathcal{M}} \phi\, \mathcal{U}_i \psi & \text{:iff } \textit{there exists } c' \in \mathcal{C}_F \textit{ such that} \\
 & \quad \downarrow^i c \subseteq \downarrow^i c' \textit{ and } (F, \downarrow^i c') \models_{\mathcal{M}} \psi \textit{ and for all} \\
 & \quad c'' \in \mathcal{C}_F \textit{ with } \downarrow^i c \subseteq \downarrow^i c'' \subset \downarrow^i c' \\
 & \quad \textit{it holds that } (F, \downarrow^i c'') \models_{\mathcal{M}} \phi \\
(F,c) \models_{\mathcal{M}} K_i\phi & \text{:iff } \textit{for all } (F',c') \textit{ with } ((F,c),(F',c')) \in R_i^K \\
 & \quad \textit{it holds that } (F',c') \models_{\mathcal{M}} \phi \\
(F,c) \models_{\mathcal{M}} B_i\phi & \text{:iff } \textit{for all } (F',c') \textit{ with } ((F,c),(F',c')) \in R_i^B \\
 & \quad \textit{it holds that } (F',c') \models_{\mathcal{M}} \phi
\end{array}
$$

Definition 4.2.2 (satisfiability, validity) *A formula* $\phi \in \mathcal{L}$ *is* **satisfiable** *iff there exists a model* $\mathcal{M}$ *and a situation* $(F,c) \in |\mathcal{M}|$ *such that* $(F,c) \models_{\mathcal{M}} \phi$*.*

A set of formulae $\Phi \subset \mathcal{L}$ *is* **satisfiable***, iff there exists a model* $\mathcal{M}$ *and a situation* $(F,c) \in |\mathcal{M}|$ *such that* $(F,c) \models_{\mathcal{M}} \phi$ *for all* $\phi \in \Phi$*.*

A formula $\phi \in \mathcal{L}$ *is* **valid** *iff* $\neg\phi$ *is not satisfiable.*

Note, that the semantics of logical connectives and of local propositions is dependent on a single configuration of a single run.

We have already stated the intuitive semantics of the temporal formulae. The semantics of temporal formulae involves several configurations, but is, however, dependent only on a single run.

Further note, that formulae of form $\langle op\rangle_i\phi$ are local to a single agent. $\langle op\rangle_i\phi$ does *not* mean, that the next operation of the run is op, but the next operation of agent i. It might well be, that some other agent which is involved in operation op performs operations other than op first.

The semantics of the knowledge/belief operators are classical: An agent i *knows* in a situation whatever is true in all situations the agent considers possible, including the actual situation. An agent i *believes* that a formula ϕ holds, $B_i\phi$, if ϕ is true in all situations which i cannot distinguish from the actual situation. According to definition 3.3.2, the relations R_i^K are equivalence relations, which means, that the actual situation is always considered as possible. So, each situation that satisfies $K_i\phi$ also satisfies ϕ. Moreover, in each situation each agent knows all valid formulae and all consequences of her knowledge. This property is called "logical omniscience".

According to definition 3.3.3, the indistinguishability relation for belief does not need to be reflexive, which means, an agent need not consider the actual situation as possible, and thus in the case of belief we cannot generally conclude ϕ from $B_i\phi$.

In the rest of this work, it will be convenient to use the following derived operators as "short cuts" to make formulae more easily understandable:

- $[op]_i\phi \equiv \neg\langle op\rangle_i\neg\phi$
  The operator $[\cdot]_i$ is the dual operator to $\langle\cdot\rangle_i$. Intuitively, $[op]_i\phi$ means that *if* op is the next operation from agent i's point of view, then after the execution of op the formula ϕ holds.
- $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$
- $\Diamond_i\phi \equiv \top\, \mathcal{U}_i\phi$
  Intuitively, $\Diamond_i\phi$ means that eventually from agent i's point of view ϕ holds.
- $\Box_i\phi \equiv \neg\Diamond_i\neg\phi$
  The box operator $\Box_i\phi$ is the dual to the diamond operator above. It means, that from agent i's point of view, ϕ holds always in the future.
- $\phi \Rightarrow \psi \equiv \neg\phi \vee \psi$
- $\phi\mathcal{W}_i\psi \equiv \phi\mathcal{U}_i\psi \vee \Box_i\phi$
  The operator $\mathcal{W}$ can be seen as a "weak"-until operator: While the until-operator $\mathcal{U}$ must eventually be "resolved", i.e. ψ must eventually become true, this is, however, not necessary in the case of the "weak"-until operator $\mathcal{W}$.

For better readability we define the following descending order of precedence:

1.	not ($\neg$)
2.	strong next ($\langle a\rangle_i$), weak next ($[a]_i$)
3.	strong until ($\mathcal{U}_i$), weak until ($\mathcal{W}_i$)
4.	knows (K_i), believes (B_i), always ($\Box_i$), eventually ($\Diamond_i$)
5.	and ($\wedge$), or ($\vee$), implies ($\Rightarrow$)

4.3 Properties of the logic $\mathcal{L}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$

In this section we show useful properties of the logic.

Some quite interesting properties of logics are whether they have the finite model property and whether they are compact: A logic has the *finite model property* if every non theorem can be falsified in some situation in a finite model. Compactness means, that every infinite set of formulae is satisfiable, iff all its finite subsets are satisfiable.

Unfortunately, the logic $\mathcal{L}$ doesn't have any of these properties. It can be easily seen, that $\mathcal{L}$ is not compact: Consider for example the set of formulae

$$\Phi := \{\top\ \mathcal{U}_i \phi\} \cup \{\neg\phi, [a]_i\neg\phi, [a]_i[a]_i\neg\phi, [a]_i[a]_i[a]_i\neg\phi, \ldots\}, \text{ where } \mathcal{O}_i := \{a\}. \tag{4.1}$$

Obviously, any every finite subset of Φ is satisfiable but the infinite set Φ is not, since the until formula demands that ϕ becomes true after a finite number of steps and the infinite set says, that ϕ is neither true now, nor after one step, nor after two steps, nor after three, etc. Consequently, the logic is *not compact*.

Next, we show that the logic does not have the finite model property: Consider for example the formula

$$\phi_1 := \mathsf{K}_i\langle a\rangle_j\top \text{ for } i \notin ag(a) \tag{4.2}$$

Agent i does not participate in operation a, and thus she cannot distinguish between the situations before and after agent j has performed operation a. According to the semantics of the knowledge operator, a situation (F, c) satisfies the formula ϕ_1 iff the sub formula $\langle a\rangle_j\top$ is satisfied in all situations (F', c') that are R_i^K-indistinguishable from situation (F, c) for agent i, and in particular in all situations that are i-equivalent to (F, c). Since the indistinguishability relation for knowledge is reflexive, the situation (F, c) must satisfy $\langle a\rangle_i\top$ and thus there must exist a situation (F, ca), such that $(F, c) \xrightarrow{a} (F, ca)$. Since $i \notin ag(a)$, we have that $(F, c) \equiv_i (F, ca)$. Again, the situation (F, ca) must satisfy $\langle a\rangle_i\phi$ and thus there must exist a situation (F, caa) such that $(F, ca) \xrightarrow{a} (F, caa)$. We can apply the same argument infinitely often. In order to satisfy formula ϕ_1, agent j must perform an infinite sequence of solely a. Consequently, the formula ϕ_1 can be satisfied only in an infinite model (in which at least agent j performs infinitely many operations).

Consider now the formula

$$\phi_2 := \neg\phi_1. \tag{4.3}$$

ϕ_2 can be falsified only in an infinite model: every finite model falsifies ϕ_1 and thus satisfies $\neg\phi_1$. Only an infinite model can satisfy ϕ_1 and thus falsify $\neg\phi_1$.

Consequently, the logic does *not have the finite model property.*

One important feature of the logic is that it is trace-consistent. Formulae are always local to a set of agents. If two situations (F, c) and $(F, \bar{c})$ are equivalent for all agent $i \in A$ in a set of agents A, then a formula of type A is true in (F, c) if and only if it is true in $(F, \bar{c})$.

Definition 4.3.1 (trace-consistent) *Let* $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{R}^B, \mathcal{I})$ *be a model. We call the language* $\mathcal{L}$ trace-consistent *iff it has the following property:*

For each pair of situations $(F, c), (F, c') \in |\mathcal{M}|$ *and for each formula* $\phi \in \mathcal{L}$ *it holds that*
if $(F, c) \equiv_A (F, c')$ *and* $\mathtt{type}(\phi) = A$
then $(F, c) \models_{\mathcal{M}} \phi$ *if and only if* $(F, c') \models_{\mathcal{M}} \phi$.

Lemma 4.3.2 *Let* $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{R}^B, \mathcal{I})$ *be a model, let* $\phi \in \Phi_A$ *be a formula of type* A *and let* $(F, c), (F, \bar{c}) \in |\mathcal{M}|$ *be two situations such that* $(F, c) \equiv_i (F, \bar{c})$ *for all* $i \in A$.

Then $(F, c) \models_{\mathcal{M}} \phi$ *iff* $(F, \bar{c}) \models_{\mathcal{M}} \phi$

Proof:
We prove this lemma by induction over the structure of the formula.

Induction begin:
Suppose, $\phi = \top$, with $\top \in \Phi_\emptyset$.
According to definition 4.2.1, it holds that $s \models_{\mathcal{M}} \top$ for every $s \in |\mathcal{M}|$ and thus in particular $(F, c) \models_{\mathcal{M}} \top$ and $(F, \bar{c}) \models_{\mathcal{M}} \top$ for every $(F, c) \equiv_\emptyset (F, \bar{c})$.

Suppose, $\phi = \bot$, with $\bot \in \Phi_\emptyset$.
According to definition 4.2.1, it holds that $s \not\models_{\mathcal{M}} \bot$ for all $s \in |\mathcal{M}|$ and thus in particular $(F, c) \not\models_{\mathcal{M}} \bot$ and $(F, \bar{c}) \not\models_{\mathcal{M}} \bot$ for every $(F, c) \equiv_\emptyset (F, \bar{c})$.

Suppose, $\phi = P$ with $P \in \mathcal{P}_i$.

$$
\begin{array}{ll}
 & (F, c) \models_{\mathcal{M}} p \\
\iff & (\text{* by definition 4.2.1 *}) \\
 & \mathcal{I}((F, c), p) = \top \\
\iff & (\text{* by definitions 3.3.4 and 3.3.2(3) *}) \\
 & \mathcal{I}((F, \bar{c}), p) = \top \\
\iff & (\text{* by definition 4.2.1 *}) \\
 & (F, \bar{c}) \models_{\mathcal{M}} p
\end{array}
$$

Induction hypothesis: Suppose, for all formulae ϕ' of a size less than that of ϕ it holds that if $(F, c) \equiv_i (F, \bar{c})$ for all $i \in \mathtt{type}(\phi')$, then $(F, c) \models_{\mathcal{M}} \phi'$ iff $(F, \bar{c}) \models_{\mathcal{M}} \phi'$.

Induction step:

- Suppose, $\phi = \neg\phi_1$ and the induction hypothesis holds for ϕ_1. Further, suppose, $(F, c) \equiv_i (F, \bar{c})$ for all $i \in \mathtt{type}(\phi)$.
  $(F, c) \models_{\mathcal{M}} \neg\phi_1$
  $\iff$ $(F, c) \not\models_{\mathcal{M}} \phi_1$
  $\iff$ (* by the induction hypothesis and because $\mathtt{type}(\phi) = \mathtt{type}(\phi_1)$ *)
  $(F, \bar{c}) \not\models_{\mathcal{M}} \phi_1$
  $\iff$ $(F, \bar{c}) \models_{\mathcal{M}} \neg\phi_1$.

- Suppose, $\phi = \phi_1 \vee \phi_2$ and $(F, c) \equiv_i (F, \bar{c})$ for all $i \in \mathtt{type}(\phi)$.
  $(F, c) \models_{\mathcal{M}} \phi_1 \vee \phi_2$
  $\iff$ (* by definition 4.2.1 *)
  $(F, c) \models_{\mathcal{M}} \phi_1$ or $(F, c) \models_{\mathcal{M}} \phi_2$
  $\iff$ (* $\mathtt{type}(\phi_1) \subseteq \mathtt{type}(\phi)$ and $\mathtt{type}(\phi_2) \subseteq \mathtt{type}(\phi)$ *)
  $(F, \bar{c}) \models_{\mathcal{M}} \phi_1$ or $(F, \bar{c}) \models_{\mathcal{M}} \phi_2$
  $\iff$ (* by definition 4.2.1 *)
  $(F, \bar{c}) \models \phi_1 \vee \phi_2$

- Suppose, $\phi = K_i\phi_1$ and $(F, c) \equiv_i (F, \bar{c})$.
  $(F, c) \models_{\mathcal{M}} K_i\phi_1$
  $\iff$ (* by definition 4.2.1 *)
  for all $(F', c') \in |\mathcal{M}|$ with $((F, c), (F', c')) \in R_i^K$ it holds that $(F', c') \models_{\mathcal{M}} \phi_1$
  $\iff$ (* by definition 3.3.2(3) and 3.3.2(1) (transitive) *)
  for all $(F', c') \in |\mathcal{M}|$ with $((F, \bar{c}), (F', c')) \in R_i^K$ it holds that $(F', c') \models_{\mathcal{M}} \phi_1$
  $\iff$ (* by definition 4.2.1 *)
  $(F, \bar{c}) \models K_i\phi_1$

- Suppose, $\phi = B_i\phi_1$ and $(F, c) \equiv_i (F, \bar{c})$.
  $(F, c) \models_{\mathcal{M}} B_i\phi_1$
  $\iff$ (* by definition 4.2.1 *)
  for all $(F', c') \in |\mathcal{M}|$ with $((F, c), (F', c')) \in R_i^B$ it holds that $(F', c') \models_{\mathcal{M}} \phi_1$
  $\iff$ (* by definition 3.3.3(2) *)
  for all $(F', c') \in |\mathcal{M}|$ with $((F, \bar{c}), (F', c')) \in R_i^B$ it holds that $(F', c') \models_{\mathcal{M}} \phi_1$
  $\iff$ (* by definition 4.2.1 *)
  $(F, \bar{c}) \models B_i\phi_1$

- Suppose, $\phi = \langle a \rangle_i \phi_1$ and $(F, c) \equiv_i (F, \bar{c})$.

$$\begin{array}{ll}
 & (F,c) \models_{\mathcal{M}} \langle a \rangle_i \phi_1 \\
\Longleftrightarrow & \text{there exist situations } (F,c'),(F,r) \in |\mathcal{M}| \text{ such that } (F,c') \equiv_i (F,c) \text{ and} \\
 & (F,c') \xrightarrow{a} (F,r) \text{ and } (F,r) \models_{\mathcal{M}} \phi_1 \\
\Longleftrightarrow & (*\ (F,c) \equiv_i (F,\bar{c})\ *) \\
 & \text{there exist situations } (F,c'),(F,r) \in |\mathcal{M}| \text{ such that } (F,c') \equiv_i (F,\bar{c}) \text{ and} \\
 & (F,c') \xrightarrow{a} (F,r) \text{ and } (F,r) \models_{\mathcal{M}} \phi_1 \\
\Longleftrightarrow & (F,\bar{c}) \models \langle a \rangle_i \phi_1
\end{array}$$

- Suppose, $\phi = \phi_1\, \mathcal{U}_i \phi_2$ and $(F,c) \equiv_i (F,\bar{c})$.

$$\begin{array}{ll}
 & (F,c) \models_{\mathcal{M}} \phi_1\, \mathcal{U}_i \phi_2 \\
\Longleftrightarrow & \text{there exists } c' \in \mathcal{C}_F \text{ such that } \downarrow^i c \subseteq \downarrow^i c' \text{ and } (F, \downarrow^i c') \models \phi_2 \text{ and for} \\
 & \text{all } c'' \in \mathcal{C}_F \text{ with } \downarrow^i c \subseteq \downarrow^i c'' \subset \downarrow^i c' \text{ it holds that } (F, \downarrow^i c'') \models_{\mathcal{M}} \phi_1 \\
\Longleftrightarrow & (*\text{ because } \downarrow^i c = \downarrow^i \bar{c}\ *) \\
 & \text{there exists } c' \in \mathcal{C}_F \text{ such that } \downarrow^i \bar{c} \subseteq \downarrow^i c' \text{ and } (F, \downarrow^i c') \models \phi_2 \text{ and for} \\
 & \text{all } c'' \in \mathcal{C}_F \text{ with } \downarrow^i \bar{c} \subseteq \downarrow^i c'' \subset \downarrow^i c' \text{ it holds that } (F, \downarrow^i c'') \models_{\mathcal{M}} \phi_1 \\
\Longleftrightarrow & (F,\bar{c}) \models_{\mathcal{M}} \phi_1\, \mathcal{U}_i \phi_2
\end{array}$$

$\square$

Remark 1: The logic $\mathcal{L}$ is a linear time logic, i.e. the events belonging to one agent are totally ordered. This means that though there are several possible next operations for the whole system in each situation, there is at most one for each agent. Logically, this means that for all situations (F,c) the following holds:

$(F,c) \models_{\mathcal{M}} \langle op \rangle_i \phi$ implies $(F,c) \models_{\mathcal{M}} [op]_i \phi$

Proof:

$$\begin{array}{ll}
 & (F,c) \models_{\mathcal{M}} \langle op \rangle_i \phi \\
\Longrightarrow & \text{there exists } c', r \in \mathcal{C}_F, \text{ such that } c \equiv_i c' \text{ and } c' \xrightarrow{op} r \text{ and } (F,r) \models_{\mathcal{M}} \phi \\
\Longrightarrow & (*\text{Definition 3.2.2 ensures that the events of each agent are totally ordered}*) \\
 & \text{for all } c', r \in \mathcal{C}_F, \text{ such that } c \equiv_i c' \text{ we have } c' \xrightarrow{op} r \text{ and } (F,r) \models_{\mathcal{M}} \phi \\
\Longrightarrow & \text{not exists } c', r \in \mathcal{C}_F, \text{ such that } c \equiv_i c' \text{ and } c' \xrightarrow{op} r \text{ and } (F,r) \models_{\mathcal{M}} \neg\phi \\
\Longrightarrow & (F,c) \models_{\mathcal{M}} \neg\langle op \rangle_i \neg\phi \\
\Longrightarrow & (F,c) \models_{\mathcal{M}} [op]_i \phi
\end{array}$$

Remark 2: The following equivalence will be helpful for defining a tableau based proof system in the next chapter:

$(F, c) \models_{\mathcal{M}} \phi\ \mathcal{U}_i\psi$

iff

$(F, c) \models_{\mathcal{M}} \psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i(\phi\ \mathcal{U}_i\psi))$ for $\langle\mathcal{O}\rangle_i\alpha := \bigvee_{o\in\mathcal{O}_i}\langle o\rangle_i\alpha$

Proof:

$(F, c) \models_{\mathcal{M}} \phi\ \mathcal{U}_i\psi$

$\iff$ (∗by definition 4.2.1∗)

there exists $c' \in \mathcal{C}_F$ such that $\downarrow^i c \subseteq \downarrow^i c'$ and $(F, \downarrow^i c') \models_{\mathcal{M}} \psi$ and

for all c'', with $\downarrow^i c \subseteq \downarrow^i c'' \subset \downarrow^i c'$ it holds that $(F, \downarrow^i c'') \models_{\mathcal{M}} \phi$

$\iff$ either$(F, \downarrow^i c) \models_{\mathcal{M}} \psi$ or

$(F, \downarrow^i c) \not\models_{\mathcal{M}} \psi$ and $(F, \downarrow^i c) \models_{\mathcal{M}} \phi$ and

there exists $c' \in \mathcal{C}_F$, such that $\downarrow^i c \subset \downarrow^i c'$ and $(F, \downarrow^i c') \models_{\mathcal{M}} \psi$ and

for all $c'' \in \mathcal{C}_F$ with $\downarrow^i c \subset \downarrow^i c'' \subset \downarrow^i c'$: $(F, \downarrow^i c'') \models_{\mathcal{M}} \phi$

$\iff$ either $(F, \downarrow^i c) \models_{\mathcal{M}} \psi$ or

$(F, \downarrow^i c) \models_{\mathcal{M}} \neg\psi \wedge \phi$ and

there exists an $o \in \mathcal{O}_i$, such that $c \xrightarrow{o} co$ and

there exists $c' \in \mathcal{C}_F$, such that $\downarrow^i co \subseteq \downarrow^i c'$ and $(F, \downarrow^i c') \models_{\mathcal{M}} \psi$ and

for all $c'' \in \mathcal{C}_F$ with $\downarrow^i co \subseteq \downarrow^i c'' \subset \downarrow^i c'$ it holds that $(F, \downarrow^i c'') \models_{\mathcal{M}} \phi$

$\iff$ either $(F, \downarrow^i c) \models_{\mathcal{M}} \psi$ or

$(F, \downarrow^i c) \models_{\mathcal{M}} \neg\psi \wedge \phi$ and $(F, \downarrow^i c) \models_{\mathcal{M}} (\bigvee\limits_{o\in\mathcal{O}_i} \langle o\rangle_i(\phi\ \mathcal{U}_i\psi))$

$\iff$ either $(F, \downarrow^i c) \models_{\mathcal{M}} \psi$ or $(F, \downarrow^i c) \models_{\mathcal{M}} \neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i\phi\ \mathcal{U}_i\psi$

$\iff$ $(F, \downarrow^i c) \models_{\mathcal{M}} \psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i\phi\ \mathcal{U}_i\psi)$

$\iff$ (∗ by lemma 4.3.2 and the fact that by definition 4.1.1

$\mathtt{type}(\psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i\phi\ \mathcal{U}_i\psi)) = \{i\}$ ∗)

$(F, c) \models_{\mathcal{M}} \psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i\phi\ \mathcal{U}_i\psi)$

$\square$

Remark 3: A similar equivalence holds for the weak until operator:

$(F, c) \models_{\mathcal{M}} \phi\mathcal{W}_i\psi$

iff

$(F, c) \models_{\mathcal{M}} \psi \vee (\neg\psi \wedge \phi \wedge [\mathcal{O}]_i(\phi\mathcal{W}_i\psi))$ for $[\mathcal{O}]_i\alpha := \bigwedge_{o\in\mathcal{O}_i}[o]_i\alpha$.

Proof:

$$
\begin{aligned}
& (F,c) \models_{\mathcal{M}} \phi\ \mathcal{W}_i \psi \\
\iff & (\ast \text{ by equivalences defined on page 39 } \ast) \\
& (F,c) \models_{\mathcal{M}} (\phi\ \mathcal{U}_i \psi) \vee \neg(\top\ \mathcal{U}_i(\neg\phi)) \\
\iff & (\ast \text{ by remark 2 on page 44 } \ast) \\
& (F,c) \models_{\mathcal{M}} (\psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i \phi\ \mathcal{U}_i\psi)) \vee \neg(\neg\phi \vee (\phi \wedge \top \wedge \langle\mathcal{O}\rangle_i \top\ \mathcal{U}_i(\neg\phi))) \\
\iff & (F,c) \models_{\mathcal{M}} (\psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i \phi\ \mathcal{U}_i\psi)) \vee (\phi \wedge (\neg\phi \vee \bot \vee \neg(\langle\mathcal{O}\rangle_i \top\ \mathcal{U}_i(\neg\phi)))) \\
\iff & (\ast \text{ by definition of } \langle\mathcal{O}\rangle_i\alpha \text{ and } [\mathcal{O}]_i\alpha\ \ast) \\
& (F,c) \models_{\mathcal{M}} (\psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i \phi\ \mathcal{U}_i\psi)) \vee (\phi \wedge (\neg\phi \vee \bot \vee [\mathcal{O}]_i \neg(\top\ \mathcal{U}_i(\neg\phi)))) \\
\iff & (\ast \text{ by equivalences defined on pages 39 f}: \top\ \mathcal{U}_i\alpha \equiv \Diamond_i\alpha \ast) \\
& (F,c) \models_{\mathcal{M}} (\psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i \phi\ \mathcal{U}_i\psi)) \vee (\phi \wedge (\neg\phi \vee \bot \vee [\mathcal{O}]_i \neg\Diamond_i\neg\phi)) \\
\iff & (\ast \text{ by equivalences defined on pages 39 f}: \neg\Diamond_i\neg\alpha \equiv \Box_i\alpha \ast) \\
& (F,c) \models_{\mathcal{M}} (\psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i \phi\ \mathcal{U}_i\psi)) \vee (\phi \wedge (\neg\phi \vee \bot \vee [\mathcal{O}]_i \Box_i\phi)) \\
\iff & (F,c) \models_{\mathcal{M}} (\psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i \phi\ \mathcal{U}_i\psi)) \vee (\phi \wedge \neg\phi) \vee (\phi \wedge \bot) \vee (\phi \wedge [\mathcal{O}]_i \Box_i\phi) \\
\iff & (F,c) \models_{\mathcal{M}} (\psi \vee (\neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i \phi\ \mathcal{U}_i\psi)) \vee (\phi \wedge [\mathcal{O}]_i \Box_i\phi) \\
\iff & (F,c) \models_{\mathcal{M}} \psi \vee (\neg\psi \wedge \phi \wedge (\langle\mathcal{O}\rangle_i \phi\ \mathcal{U}_i\psi \vee [\mathcal{O}]_i \Box_i\phi)) \\
\iff & (\ast \text{ by definition 4.2.1 and definitions of } \langle\mathcal{O}\rangle_i\alpha \text{ and } [\mathcal{O}]_i\alpha\ \ast) \\
& (F,c) \models_{\mathcal{M}} \psi \vee (\neg\psi \wedge \phi \wedge [\mathcal{O}]_i(\phi\ \mathcal{U}_i\psi) \vee (\Box_i\phi)) \\
\iff & (\ast \text{ by equivalences defined on page 39 } \ast) \\
& (F,c) \models_{\mathcal{M}} \psi \vee (\neg\psi \wedge \phi \wedge [\mathcal{O}]_i(\phi\ \mathcal{W}_i\psi)
\end{aligned}
$$

□

Remark 4
The "strong" and the "weak" until operators can be seen as duals: For all situations (F,c) it holds that

$$(F,c) \models_{\mathcal{M}} \neg(\phi\ \mathcal{U}_i\psi) \text{ iff } (F,c) \models_{\mathcal{M}} \neg\psi\ \mathcal{W}_i(\neg\phi \wedge \neg\psi) \tag{4.4}$$

$$(F,c) \models_{\mathcal{M}} \neg(\phi\ \mathcal{W}_i\psi) \text{ iff } (F,c) \models_{\mathcal{M}} \neg\psi\ \mathcal{U}_i(\neg\phi \wedge \neg\psi) \tag{4.5}$$

Proof:

1. Equation 4.4:

$(\mathsf{F}, \mathsf{c}) \models_{\mathcal{M}} \neg(\phi\ \mathcal{U}_i \psi)$
$\iff$ (* by definition 4.2.1 *)
$\iff$ there does *not* exist a configuration $c' \in \mathcal{C}_\mathsf{F}$ with $\downarrow^i c \subseteq \downarrow^i c'$ such that $(\mathsf{F}, \downarrow^i c') \models_{\mathcal{M}} \psi$ and for all $c'' \in \mathcal{C}_\mathsf{F}$ with $\downarrow^i c \subseteq \downarrow^i c'' \subset \downarrow^i c'$ it holds that $(\mathsf{F}, \downarrow^i c'') \models_{\mathcal{M}} \phi$
$\iff$ either for all configurations $c' \in \mathcal{C}_\mathsf{F}$ with $\downarrow^i c \subseteq \downarrow^i c'$ it holds that $(\mathsf{F}, \downarrow^i c') \models_{\mathcal{M}} \neg\psi$
or there exists a configuration $c'' \in \mathcal{C}_\mathsf{F}$ with $\downarrow^i c \subseteq \downarrow^i c''$ such that $(\mathsf{F}, \downarrow^i c'') \models_{\mathcal{M}} \neg\phi \wedge \neg\psi$ and for all configurations $\downarrow^i \bar{c}$ with $\downarrow^i c \subset \downarrow^i \bar{c} \subseteq \downarrow^i c''$ it holds that $(\mathsf{F}, \downarrow^i \bar{c}) \models_{\mathcal{M}} \neg\psi$
$\iff$ (* by definition 4.2.1 *)
$(\mathsf{F}, \downarrow^i c) \models_{\mathcal{M}} \neg(\top\ \mathcal{U}_i \psi)$
or $(\mathsf{F}, \downarrow^i c) \models_{\mathcal{M}} (\neg\psi\ \mathcal{U}_i(\neg\phi \wedge \neg\psi))$
$\iff$ (* by definitions on page 39 and definition 4.2.1 *)
$(\mathsf{F}, \downarrow^i c) \models_{\mathcal{M}} (\neg\psi\ \mathcal{U}_i(\neg\phi \wedge \neg\psi)) \vee \Box_i \neg\psi$
$\iff$ (* by definitions on page 39 *)
$(\mathsf{F}, \downarrow^i c) \models_{\mathcal{M}} \neg\psi\ \mathcal{W}_i(\neg\phi \wedge \neg\psi)$
$\iff$ (* by lemma 4.3.2 *)
$(\mathsf{F}, \mathsf{c}) \models_{\mathcal{M}} \neg\psi\ \mathcal{W}_i(\neg\phi \wedge \neg\psi)$

2. Equation 4.5:

$(\mathsf{F}, \mathsf{c}) \models_{\mathcal{M}} \neg(\phi\ \mathcal{W}_i \psi)$
$\iff$ (* by definitions on page 39 *)
$(\mathsf{F}, \mathsf{c}) \models_{\mathcal{M}} \neg((\phi\ \mathcal{U}_i \psi) \vee \neg(\top\ \mathcal{U}_i \neg\phi))$
$\iff$ $(\mathsf{F}, \mathsf{c}) \models_{\mathcal{M}} \neg(\phi\ \mathcal{U}_i \psi) \wedge (\top\ \mathcal{U}_i \neg\phi))$
$\iff$ (* by definition 4.2.1 *)
there does *not* exist a configuration $c' \in \mathcal{C}_\mathsf{F}$ with $\downarrow^i c \subseteq \downarrow^i c'$ such that $(\mathsf{F}, \downarrow^i c') \models_{\mathcal{M}} \psi$ and for all configurations $c'' \in \mathcal{C}_\mathsf{F}$ with $\downarrow^i c \subseteq \downarrow^i c'' \subset \downarrow^i c'$ it holds that $(\mathsf{F}, \downarrow^i c'') \models_{\mathcal{M}} \phi$
and there does exist a configuration $c' \in \mathcal{C}_\mathsf{F}$ with $\downarrow^i c \subseteq \downarrow^i c'$ such that $(\mathsf{F}, \downarrow^i c') \models_{\mathcal{M}} \neg\phi$
$\iff$ there exists a (least) configuration $c' \in \mathcal{C}_\mathsf{F}$ with $\downarrow^i c \subseteq \downarrow^i c'$ such that $(\mathsf{F}, \downarrow^i c') \models_{\mathcal{M}} \neg\phi \wedge \neg\psi$ and for all configurations $c'' \in \mathcal{C}_\mathsf{F}$ with $\downarrow^i c \subseteq \downarrow^i c'' \subset \downarrow^i c'$ it holds that $(\mathsf{F}, \downarrow^i c'') \models_{\mathcal{M}} \neg\psi$
$\iff$ (* by definition 4.2.1 *)
$(\mathsf{F}, \downarrow^i c) \models_{\mathcal{M}} \neg\psi\ \mathcal{U}_i(\neg\phi \wedge \neg\psi)$
$\iff$ (* by lemma 4.3.2 *)
$(\mathsf{F}, \mathsf{c}) \models_{\mathcal{M}} \neg\psi\ \mathcal{U}_i(\neg\phi \wedge \neg\psi)$

$\Box$

In our framework, the knowledge of an agent is determined by her local state. The agent knows whatever is true in all situations that she considers to be possible. We do not require, that agents always keep track of the information they had in the past, but allow agents to forget information they had already acquired. This means, we do not assume that the agents in our framework have perfect recall. Agents with perfect recall acquire new information by performing operations while keeping track of their “old” information. Since we allow agents to “forget” about their past behavior, the following formulae do not hold in general:

$$\mathsf{K}_i[a]_i\phi \quad \Rightarrow [a]_i\mathsf{K}_i\phi \tag{4.6}$$

$$\mathsf{K}_i\Box_i\phi \quad \Rightarrow \Box_i\mathsf{K}_i\phi \tag{4.7}$$

The following very simple model $\mathcal{M}$ contains a configuration, that does not satisfy the formula $\mathsf{K}_i[a]_i\phi \Rightarrow [a]_i\mathsf{K}_i\phi$.
Let $\mathsf{Ag} = \{i\}, \mathcal{O}_i = \{a\}$ and $\mathcal{P}_i = \{p\}$ be the static declaration of a model $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{R}^B, \mathcal{I})$ with

- $A = \{F_1, F_2\}$ with $F_1 = (\{e\}, \{(e,e)\}, \lambda_1)$ and $F_2 = (\emptyset, \emptyset, \lambda_2)$ and $\lambda_1(f) = a$ for all $f \in \{e\}$ and $\lambda_2(f) = a$ for all $f \in \emptyset$,
- $\mathcal{R}^K = (R_i^K)$ and $R_i^K = \{((F_1,\emptyset),(F_1,\emptyset)), ((F_1,\{e\}),(F_1,\{e\})), ((F_1,\{e\}),(F_2,\emptyset)), ((F_2,\emptyset),(F_2,\emptyset)), ((F_2,\emptyset),(F_1,\{e\}))\}$,
- $\mathcal{R}^B = \mathcal{R}^K$,
- $\mathcal{I}((F_1,\emptyset),p) = \mathcal{I}((F_2,\emptyset),p) = \bot$ and $\mathcal{I}((F_1,\{e\}),p) = \top$

We show, that the initial configuration $(F_1,\emptyset)$ of run F_1 in this model satisfies the formula $\mathsf{K}_i[a]_ip \wedge \langle a\rangle_i\neg\mathsf{K}_ip$ and thus does not satisfy the formula $\mathsf{K}_i[a]_i\phi \Rightarrow [a]_i\mathsf{K}_i\phi$ (which is equivalent to $\neg(\mathsf{K}_i[a]_ip \wedge \langle a\rangle_i\neg\mathsf{K}_ip)$):

Proof:

$(F_1,\emptyset) \models_{\mathcal{M}} \mathsf{K}_i[a]_ip \wedge \langle a\rangle_i\neg\mathsf{K}_ip$

$\Longleftrightarrow$ (* by definition 4.2.1 *)

$(F_1,\emptyset) \models_{\mathcal{M}} \mathsf{K}_i[a]_ip$ and $(F_1,\emptyset) \models_{\mathcal{M}} \langle a\rangle_i\neg\mathsf{K}_ip$

$\Longleftrightarrow$ (* by definition 4.2.1 *)

$(F_1,\emptyset) \models_{\mathcal{M}} [a]_ip$ and $(F_1,\{e\}) \models_{\mathcal{M}} \neg\mathsf{K}_ip$

$\Longleftrightarrow$ (* by definition 4.2.1 and definition of $\mathcal{M}$ *)

$(F_1,\{e\}) \models_{\mathcal{M}} p$ and there exists a situation $s \in |\mathcal{M}|$ with $((F_1,\{e\}),s) \in R_i^K$ and $s \models_{\mathcal{M}} \neg p$

$\Longleftrightarrow$ (* by definition 4.2.1 *)

$$\begin{array}{ll} & \mathcal{I}((\mathsf{F}_1,\{e\}),p) = \top \text{ and } ((\mathsf{F}_1,\{e\}) \models_{\mathcal{M}} \neg p \text{ or } (\mathsf{F}_2,\emptyset) \models_{\mathcal{M}} \neg p) \\ \iff & (\text{* by definition 4.2.1 and definition of } \mathcal{M} \text{ *}) \\ & \mathcal{I}((\mathsf{F}_1,\{e\}),p) = \top \text{ and } \mathcal{I}((\mathsf{F}_2,\emptyset),p) = \bot \end{array}$$

Consequently, the situation $(\mathsf{F}_1,\emptyset)$ does not satisfy the formula $\mathsf{K}_i[a]_i\phi \Rightarrow [a]_i\mathsf{K}_i\phi$.

□

Chapter 5

A Tableau Proof System for $\mathcal{L}_{(\mathsf{Ag},\tilde{\mathcal{O}},\tilde{\mathcal{P}})}$

In the previous chapters we have developed a class of formal models $\mathcal{M}$ of information systems and a logic $\mathcal{L}$, the semantics of which is based on these models. The models $\mathcal{M}$ together with the language $\mathcal{L}$ can be seen as a unifying logical framework for the specification of semantic constraints and security constraints for information systems. The intention behind specifying these two types of constraints in a unified framework is to find conflicts between them: given a set of formulae $\Gamma \subset \mathcal{L}$ as a specification of the desired constraints. Is this set of formulae satisfiable? Does there exist a model and a situation of this model that satisfies each formula $\gamma \in \Gamma$?

The set of formulae Γ that specifies the desired constraints might be very big so that it will be very hard to prove its satisfiability (or non-satisfiability) by hand. We wish to have a tool for checking satisfiability (or non-satisfiability) of Γ automatically. A well known method in automated deduction are semantic tableaux. In this chapter, we will develop a tableau method for a subset of our logic $\mathcal{L}$.

The models $\mathcal{M}$ contain three different relations for time, for knowledge and for belief. These relations are not independent of each other: Each agent believes everything that she knows. Further, knowledge of an agent can change only due to operations this agent performs. These dependencies between operations and change of knowledge as well as between knowledge and believe of agents make the structure of models quite involved. As we will see later, the structure of the models must in some sense be represented in the tableau method. This makes the definition of the tableau rules quite complicated. For feasibility reasons, we therefor decided to restrict the logic to the fragment, that contains only modal knowledge operators instead of knowledge and belief operators. We then do not need to take the relations $\mathsf{R}^{\mathsf{B}}_{\mathsf{i}}$ between situations of the model into account and restrict the model to $\mathcal{M} = (\mathcal{A}, \mathcal{R}^{\mathsf{K}}, \mathcal{I})$ only.

5.1 Introduction to Tableaux

The method of semantic tableaux can be seen as a refutation procedure that decomposes a given set of formulae into a network of sets. It is a well established tool in automated deduction to prove, whether a formula is a logical consequence of a set of formulae.

Modal tableaux and modal logics became increasingly important with the introduction of “possible-worlds-semantics” by Kripke in 1959. One can essentially distinguish between two types of modal tableaux: Explicit systems and implicit systems. In explicit systems the reachability relations between situation (or possible worlds in Kripke Semantics) are explicitly represented by some means and we are allowed to directly reason about the known properties of the relations. In implicit systems, there is no explicit representation of the reachability relations and we cannot directly reason about their properties. Instead, the properties of the reachability relations must be built in the logical formulae.

As Rajeev Gore states in [Gor99], there is a subtle, but important difference between implicit and explicit tableaux: Implicit systems are local in the sense, that in each tableau node, it works only on a set of formulae local to one world (situation) with no explicit references to the particular properties of the reachability relations. Explicit tableaux are more global in the sense, that one can “see” the reachability relation and hence keep a picture of the whole model under construction. Our logic is a multi modal logic with strong dependencies between the temporal component and the epistemic component of a model. In section 5.3 we will discuss the specific difficulties that arise through the strong interdependencies between the various reachability relations and argue, why we employ a version of explicit tableaux.

Let us now give a brief introduction in the syntax of tableaux, following [Gor99].

We use Γ for a set of formulae and γ, ϕ, ψ as formulae. Each tableau node is labelled with a finite set of formulae. We write Γ, ϕ instead of $\Gamma \cup \{\phi\}$. Then, obviously, $\Gamma, \phi, \phi = \Gamma, \phi$ and also $\Gamma, \phi, \psi = \Gamma, \psi, \phi$, i.e. the number of copies of a formula as well as the order in which formulae occur in the set is immaterial as far as the notation is concerned.

A *tableau rule* consists of a finite set of formula schemas called the *numerator* $\mathcal{N}$ and a (finite) list of finite sets of formulae called denominators $\mathcal{D}_1, \ldots \mathcal{D}_n$:

$$\frac{\mathcal{N}}{\mathcal{D}_1 \mid \ldots \mid \mathcal{D}_n} \tag{5.1}$$

A tableau rule is to be read as *If the numerator $\mathcal{N}$ is satisfiable, then so is one of the denominators $\mathcal{D}_1, \ldots \mathcal{D}_n$.* The numerator of each rule contains one or more distinguished formulae called the *principal formulae.* Each denominator may contain distinguished formulae called *side formulae.*

Below at the right, you find a tableau rule with

1. a numerator $\Gamma, \phi \vee \psi$ with a principal formula $\phi \vee \psi$
2. two denominators Γ, ϕ and Γ, ψ with side-formulae ϕ and ψ respectively.

$$\frac{\Gamma, \phi \vee \psi}{\Gamma, \phi \quad | \quad \Gamma, \psi}$$

A tableau system is then a finite collection of tableau rules. A tableau for a set of formulae Γ is an (infinite) tree of finite grade, whose nodes are labelled with finite sets of formulae. A tableau rule with a numerator $\mathcal{N}$ is applicable to a node, if the formula set at this node is an instance of $\mathcal{N}$.

There are two steps for extending a tableau tree:

1. Choose a leaf node that is labelled with a formula set Γ and choose a tableau rule that is applicable to Γ.
2. If the rule has n denominators, then create n successor nodes, with successor node i carrying an appropriate instantiation of the ith denominator.

5.2 Preliminaries

The logic $\mathcal{L}$ does not provide a dual for every operator. However, we have defined the duals of the logical and modal operators as derived operators on page 39 and have shown duality of strong and weak until in remark 4 on page 45. For simplification of the presentation we use the following equivalences for negative formulae to push the negation symbols inside. For most operators, the equivalence is obvious, for the operators $\mathcal{U}$ we have shown the equivalence in remark (4) on page 45 We allow negation only in front of atomic propositions and in front of epistemic formulae. As we will see, applications of the rules (**TR 7**), (**TR 8**) and (**TR 9**) put a negation symbol in front of a formula. In this case we assume that the negation symbol is immediately pushed inside according to the following equivalences. Similarly, at the beginning we assume, that all negation symbols occur only in front of knowledge operators or in front of atomic propositions $P \in \mathcal{P}$ (except for $\top$ and $\bot$).

$\neg\top$	$\bot$
$\neg\bot$	$\top$
$\neg\neg P$	P
$\neg\neg K_i\phi$	$K_i\phi$
$\neg(\phi \wedge \psi)$	$\neg\phi \vee \neg\psi$
$\neg\langle op\rangle_i\phi$	$[op]_i\neg\phi$
$\neg[op]_i\phi$	$\langle op\rangle_i\neg\phi$
$\neg(\phi\mathcal{U}_i\psi)$	$\neg\psi\mathcal{W}_i(\neg\phi \wedge \neg\psi)$
$\neg(\phi\mathcal{W}_i\psi)$	$\neg\psi\mathcal{U}_i(\neg\phi \wedge \neg\psi)$

5.3 Tableau Formulae

Our logic is a multimodal logic with temporal and epistemic modalities. In the following we will consider the temporal successor relation (see definition 3.2.3) as well as the epistemic indistinguishability relations for each agent (see definition 3.3.2) uniformly as accessibility relations on situations of a model $\mathcal{M}$. The temporal and the epistemic accessibility relations are not defined independently, there are interdependencies among the two types of accessibility relations.

The fourth condition of definition 3.3.2 requires that if in a situation (F, c) an operation o is possible, and if for all agents participating in operation o situation (F, c) is indistinguishable from a situations (F', c'), then operation o must be possible next in situation (F', c') as well and the resulting situations (F, co) and $(F', c'o)$ respectively must be indistinguishable as well for each of the participating agents. Figure 5.1 illustrates this requirement.

$$\begin{tikzcd}
(F, c) \arrow[r, "a"] \arrow[d, "R_i^K", "{\text{forall } i \in ag(a)}"'] & (F, ca) \arrow[d, dashed, "R_i^K", "{\text{forall } i \in ag(a)}"'] \\
(F', c') \arrow[r, dashed, "a"] & (F', c'a)
\end{tikzcd}$$

Figure 5.1: Interdependencies between temporal and epistemic relations

Representing this restriction in tableau rules seems to be quite involved. In [DG01] Davoren and Gore define a tableau system for a logic with a comparable requirement. They call it the lower diamond property as the lower part of the diamond follows from the upper part. The tableau system defined by Davoren and Gore is not complete in the sense, that there exist accepting tableaux for unsatisfiable formulae. They give a formula as a counterexample that is not satisfiable but the tableau system does accept the formula to be satisfiable. The problem is exactly the above described property; it has "non-local" effects: A kind of "global view" is required in order to detect and repair incomplete diamonds. According to [Gor99] we can distinguish between *static tableau rules*, in which the numerator of the rule corresponds to the same situation as all its denominators and *dynamic tableau rules* in which the denominator corresponds to the creation of a "successor" (either temporal or epistemic) situation. However, in both cases the formulae in one node correspond to only one situation. In case of the described interdependencies between temporal and epistemic relations a more global view on the model under construction is demanded.

A possible solution to this problem is to use explicit tableaux, which means, we explicitly represent the accessibility relations in the tableaux. Labelled tableaux are one kind of explicit tableaux. The main principle is to use tableau labels to bring some of the semantics into the syntax, as is suggested in [Fit83], [Fit96] and [Mas94]. Massacci uses structured labels as prefixes for formulae. Given two labels we are able to tell, whether they are related by the accessibility relation simply by looking at their structure. In our case, this is not so simple as we have to deal with a set of accessibility relations, not only one. Thus we slightly modify this technique and do not use structured labels but unstructured ones and relate them explicitly, so that we can reason about formulae as well as about the labels and the structure they suggest.

We introduce a tableau language $\mathfrak{L}$ which extends the logic $\mathcal{L}$ in the following way: We introduce a set of unstructured tableau labels each of which will later-on represent a situation of a model. All formulae $\phi \in \mathcal{L}$ will be preceded by such a tableau label which means, we can allocate formulae with different situations within one node of the tableau. Formulae preceded by a tableau label are called *labelled formulae*.

Further, we introduce a second type of formulae, so called *structure formulae*. They represent the various relations between various situations. This enables us to represent the structure of models directly inside the tableau and also to directly reason about the represented structure.

The third type of formulae are the labels themselves. They are included in the tableau language $\mathfrak{L}$ to ensure that a particular tableau rule (namely rule (**TR 14**)) will later on be applied in the correct way.

Recall, that the logic $\mathcal{L}$ does not have the finite model property: the formula $\neg K_i\langle a\rangle_j\phi$ with $i \notin ag(a)$ can be falsified only in an infinite model and $K_i\langle a\rangle_j\phi$ can be satisfied only in an infinite model. Similarly, the formula $\phi\ \mathcal{W}_i\bot$ can be satisfied only in an infinite model. This property causes an important problem for the tableau procedure: Each tableau path represents (a 'serialization' of each run of) a possible model for the set of formulae under consideration. Each situation of such a serialization is represented by a tableau label which means, we need to use infinitely many labels and may have to construct infinite tableau paths on which infinitely many labels occur. However, for each node in the tableau the set of formulae in this node is finite. Only the union of all nodes along an infinite path may contain an infinite set of formulae.

Another problem in the following tableau system is caused by formulae of type $\mathsf{K}_i\phi$, where $\phi \notin \Phi_{\{i\}}$. Formulae of this type cause the tableau system to be not complete. However, in chapter 6, section 6.2 we will show completeness for a reasonable subset of the logic $\mathcal{L}$, for which we require that for a formula $\mathsf{K}_i\phi$ it holds that $\phi \in \Phi_{\{i\}}$.

Definition 5.3.1 (tableau language $\mathfrak{L}$) *Let $\mathfrak{label}$ be a set of (unstructured) tableau labels containing a distinctive label $\mathfrak{l}_0$.*

We define the set of labelled formulae $\mathfrak{L}_\mathfrak{l}$ as follows:
If $\phi \in \mathcal{L}$ is a formula and if $\mathfrak{l} \in \mathfrak{label}$ is a tableau label, then

$$\mathfrak{l} \vdash \phi$$

is a **labelled formula**.

We define the set of structure formulae $\mathfrak{L}_\mathfrak{s}$ as follows:
If $\mathfrak{l}, \mathfrak{l}'$ are tableau labels and if $a \in \mathcal{O}$ is an operation and $i \in \mathsf{Ag}$ is an agent then

$$\mathfrak{l} \xrightarrow{a} \mathfrak{l}'$$

and

$$\mathfrak{l} R_i \mathfrak{l}'$$

are **structure formulae**.

The tableau language $\mathfrak{L}$ is defined as the union of the set of labelled formulae $\mathfrak{L}_\mathfrak{l}$, the set of structure formulae $\mathfrak{L}_\mathfrak{s}$ and set of tableau labels $\mathfrak{label}$:

$$\mathfrak{L} = \mathfrak{L}_\mathfrak{l} \cup \mathfrak{L}_\mathfrak{s} \cup \mathfrak{label}$$

For a set of tableau formulae $\Gamma_\mathcal{T} \subset \mathfrak{L}$ we denote the set of tableau labels occurring in $\Gamma_\mathcal{T} \cap (\mathfrak{L}_\mathfrak{l} \cup \mathfrak{L}_\mathfrak{s})$ as $\mathfrak{label}(\Gamma_\mathcal{T})$:

- $\mathfrak{l}_0 \in \mathfrak{label}(\Gamma_\mathcal{T})$,

- $\mathfrak{l} \vdash \gamma \in \Gamma_{\mathcal{T}}$ *implies that* $\mathfrak{l} \in \mathfrak{label}(\Gamma_{\mathcal{T}})$,
- $\mathfrak{l}\mathsf{R}_i\mathfrak{l}' \in \Gamma_{\mathcal{T}}$ *implies that* $\mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_{\mathcal{T}})$ *and*
- $\mathfrak{l} \xrightarrow{a} \mathfrak{l}' \in \Gamma_{\mathcal{T}}$ *implies that* $\mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_{\mathcal{T}})$.
- *For all other tableau labels* $\mathfrak{l} \in \mathfrak{label}$ *we have that* $\mathfrak{l} \notin \mathfrak{label}(\Gamma_{\mathcal{T}})$.

We define $\Gamma_{\mathcal{T}} \subset \mathfrak{L}$ *to be a* **tableau set** *iff*

$$\Gamma_{\mathcal{T}} \cap \mathfrak{label} = \mathfrak{label}(\Gamma_{\mathcal{T}}).$$

There is a close relation between the tableau language $\mathfrak{L}$ on the one hand and models and the language $\mathcal{L}$ on the other hand. The semantics of a formula $\phi \in \mathcal{L}$ is defined on situations of a model. We say, *a situation* (F, c) *of a model* $\mathcal{M}$ *satisfies a formula* $\phi \dots$ and if we have a set of formulae $\Gamma \subset \mathcal{L}$ we say, *a situation* (F, c) *of a model* $\mathcal{M}$ *satisfies a set of formulae* $\Gamma \dots$. All formulae of Γ must be satisfied in the *same* situation. In the language $\mathcal{L}$ we cannot formulate that different formulae γ and γ' contained in the same set of formulae Γ are related to different situations. This is, however, possible in the tableau language $\mathfrak{L}$. Two different labelled formulae $\mathfrak{l} \vdash \gamma$ and $\mathfrak{l}' \vdash \gamma'$ may refer to two different situations of a model. We define an *embedding* ℓ from the set of tableau labels $\mathfrak{label}(\Gamma_{\mathcal{T}})$ into the set of situations $|\mathcal{M}|$ of a model. The tableau formula $\mathfrak{l} \vdash \gamma$ then "means" that the formula $\gamma \in \mathcal{L}$ is satisfied by situation $\ell(\mathfrak{l})$ and the tableau formula $\mathfrak{l}' \vdash \gamma'$ "means" that the formula $\gamma' \in \mathcal{L}$ is satisfied by situation $\ell(\mathfrak{l}')$. Note, that these two tableau formulae may represent two different situations because the tableau labels $\mathfrak{l}$ and $\mathfrak{l}'$ may be embedded into different situations, i.e. $\ell(\mathfrak{l}) \neq \ell(\mathfrak{l}')$.

The language $\mathfrak{L}$ contains structure formulae, that refer to the structure of the model via the embedding of tableau labels into situations of a model. Strictly speaking, structure formulae refer to the indistinguishability relations for knowledge and to the temporal successor relation on situations.

Let us now capture this correspondence more formally:

Definition 5.3.2 ($\mathcal{L}$-embedding, $\mathcal{L}-$satisfiable) *Let* $\Gamma_{\mathcal{T}} \subset \mathfrak{L}$ *be a tableau set (i.e.* $\Gamma_{\mathcal{T}} \cap \mathfrak{label} = \mathfrak{label}(\Gamma_{\mathcal{T}})$ *) and let* $\mathcal{M} = (\mathcal{A}, \mathcal{R}^{\mathsf{K}}, \mathcal{I})$ *be a model.*

A mapping $\ell : \mathfrak{label}(\Gamma_{\mathcal{T}}) \longrightarrow |\mathcal{M}|$ *is an* **$\mathcal{L}$-embedding** *from the set of tableau labels* $\mathfrak{label}(\Gamma_{\mathcal{T}})$ *into the set of situations* $|\mathcal{M}|$, *iff for all tableau labels* $\mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_{\mathcal{T}})$, *for all agents* $\mathsf{i} \in \mathsf{Ag}$ *and for all operations* $\mathsf{o} \in \mathcal{O}$ *it holds that:*

- $\mathfrak{l}\ \mathsf{R}_i\ \mathfrak{l}' \in \Gamma_{\mathcal{T}}$ *implies* $(\ell(\mathfrak{l}), \ell(\mathfrak{l}')) \in \mathsf{R}^{\mathsf{K}}_i$

- $\mathfrak{l} \xrightarrow{\mathsf{o}} \mathfrak{l}' \in \Gamma_{\mathcal{T}}$ *implies that there exists a run* $\mathsf{F} \in \mathcal{A}$*, such that* $\ell(\mathfrak{l}) \in \mathcal{C}_{\mathsf{F}}$ *and* $\ell(\mathfrak{l}') \in \mathcal{C}_{\mathsf{F}}$ *and* $\ell(\mathfrak{l}) \xrightarrow{\mathsf{o}} \ell(\mathfrak{l}')$

$\Gamma_{\mathcal{T}}$ *is* $\mathcal{L}-$ **satisfiable with $\mathcal{L}$-embedding** ℓ *iff for all labelled formulae* $\mathfrak{l} \vdash \gamma \in \Gamma_{\mathcal{T}}$ *it holds that:*

$$\ell(\mathfrak{l}) \models_{\mathcal{M}} \gamma$$

$\Gamma_{\mathcal{T}}$ *is* $\mathcal{L}-$ **satisfiable** *iff there exists a model* $\mathcal{M} = (\mathcal{A}, \mathcal{R}^{\mathsf{K}}, \mathcal{I})$ *and there exists an* $\mathcal{L}$*-embedding* $\ell : \mathfrak{label} \longrightarrow |\mathcal{M}|$*, such that* $\Gamma_{\mathcal{T}}$ *is* $\mathcal{L}-$ *satisfiable with* $\mathcal{L}$*-embedding* ℓ*.*

If a set of tableau formulae $\Gamma_{\mathcal{T}} \subset \mathfrak{L}$ *is* $\mathcal{L}-$ *satisfiable with* $\mathcal{L}$*-embedding* ℓ*, then we call* ℓ *a* **satisfying $\mathcal{L}$-embedding** *for* $\Gamma_{\mathcal{T}}$*.*

Note: In reference to the close relation between tableau label and situations, we sometimes use a notation of label that is similar to the notation of situations, e.g. $(\mathfrak{F}, \mathfrak{c})$ instead of $\mathfrak{l}$ and $\mathfrak{l} \xrightarrow{\mathsf{a}} \mathfrak{la}$ or $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{\mathsf{a}} (\mathfrak{F}, \mathfrak{ca})$ instead of $\mathfrak{l} \xrightarrow{\mathsf{a}} \mathfrak{l}'$. Note, however, that tableau labels are only syntactic elements, they do not carry any semantics!

A set of formulae $\Gamma \subset \mathcal{L}$ is satisfiable iff there exists a model and a situation (F, c) in this model such that (F, c) satisfies each formula $\gamma \in \Gamma$. To prove satisfiability for a finite set of formula $\Gamma \subset \mathcal{L}$ we construct a tableau set from Γ to which we then apply the tableau rules. We take a label $\mathfrak{l} \in \mathfrak{label}$ and prefix each formula $\gamma \in \Gamma$ with this label. Further, we also add the label itself to the set to ensure, that the constructed set is a tableau set.

Definition 5.3.3 (construction of tableau formulae) *Let* $\Gamma \subset \mathcal{L}$ *be a finite set of formulae and let* $\mathfrak{l}_{\mathsf{o}} \in \mathfrak{label}$ *be the distinctive tableau label.*

The constructor τ *constructs a set of tableau formulae* $\Gamma_{\mathcal{T}}$ *out of the set of formulae* $\Gamma \subset \mathcal{L}$ *by*

1. *adding the tableau label* $\mathfrak{l}_{\mathsf{o}}$ *to the set of tableau formulae and*

2. *prefixing each formula* $\gamma \in \Gamma$ *by tableau label* $\mathfrak{l}_{\mathsf{o}}$*.*

$\tau : 2^{\mathcal{L}} \longrightarrow 2^{\mathfrak{L}}$ *where* $\tau(\Gamma) := \{\mathfrak{l}_{\mathsf{o}} \vdash \gamma | $ *for all* $\gamma \in \Gamma\} \cup \{\mathfrak{l}_{\mathsf{o}}\}$

Note, that the set of tableau formulae $\Gamma_{\mathcal{T}}$ constructed by constructor τ is a tableau set, i.e. $\Gamma_{\mathcal{T}} \cap \mathfrak{label} = \mathfrak{label}(\Gamma_{\mathcal{T}})$.

As defined above, a set of tableau formulae consists of structure formulae, labelled formulae and tableau labels. Often we are not only interested in a single structure formula but in the combination of structure formulae in the set. As defined in the definition of $\mathcal{L}$-embedding (definition 5.3.2), a structure formula of the type $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a)$ refers to a transition between situations $\ell((\mathfrak{F}, \mathfrak{c}))$ and $\ell((\mathfrak{F}, \mathfrak{c}a))$ for some appropriate $\mathcal{L}$-embedding ℓ into a model $\mathcal{M}$. We are interested in those subsets of the structure formulae in a set of tableau formulae that are related to interleavings of a run in a model. We call these subsets **chains**:

Definition 5.3.4 (chain) *Let $\Gamma_\mathcal{T}$ be a tableau set. We define a* **chain of** $\Gamma_\mathcal{T}$ *inductively as:*

$\chi = (\mathfrak{F}, \mathfrak{c})$ *is a* **chain** *of* $\Gamma_\mathcal{T}$ *iff* $(\mathfrak{F}, \mathfrak{c}) \in \mathfrak{label}(\Gamma_\mathcal{T})$.

$\chi = (\mathfrak{F}, \mathfrak{c}) \xrightarrow{o} (\mathfrak{F}, \mathfrak{c}')$ *is a* **chain** *of* $\Gamma_\mathcal{T}$ *iff* $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{o} (\mathfrak{F}, \mathfrak{c}') \in \Gamma_\mathcal{T}$.

If $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} \ldots \xrightarrow{o_{n-1}} (\mathfrak{F}, \mathfrak{c}_n)$ *is a chain of* $\Gamma_\mathcal{T}$
and $(\mathfrak{F}, \mathfrak{c}_n) \xrightarrow{o_n} (\mathfrak{F}, \mathfrak{c}_{n+1}) \in \Gamma_\mathcal{T}$
then $\chi' := \chi \xrightarrow{o_n} (\mathfrak{F}, \mathfrak{c}_{n+1})$ *is a chain of* $\Gamma_\mathcal{T}$.

We call a chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} \ldots \xrightarrow{o_{n-1}} (\mathfrak{F}, \mathfrak{c}_n)$ *of* $\Gamma_\mathcal{T}$ **maximal** *iff it does not exist a formula* $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{o} (\mathfrak{F}, \mathfrak{c}') \in \Gamma_\mathcal{T}$ *such that* $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_n)$ *or* $(\mathfrak{F}, \mathfrak{c}') = (\mathfrak{F}, \mathfrak{c}_o)$.

We call an infinite chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} \ldots$ *of* $\Gamma_\mathcal{T}$ **maximal** *iff it does not exist a formula* $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{o} (\mathfrak{F}, \mathfrak{c}') \in \Gamma_\mathcal{T}$ *such that* $(\mathfrak{F}, \mathfrak{c}') = (\mathfrak{F}, \mathfrak{c}_o)$.

Two sets of tableau formulae $\Gamma_\mathcal{T}^1$ *and* $\Gamma_\mathcal{T}^2$ *are called* **chain-equivalent** *(*$\Gamma_\mathcal{T}^1 \simeq \Gamma_\mathcal{T}^2$*) iff they contain the same set of maximal chains.*

In section 5.4 we define a tableau system the rules of which work on tableau sets.

The idea of the tableau system is as follows:

1. We start with a finite set of formulae $\Gamma_0 \subset \mathcal{L}$ that we want to prove unsatisfiable.
2. We transform $\Gamma_0 \subset \mathcal{L}$ by means of construction τ into a set of tableau formulae $\Gamma_0^\mathcal{T} \subset \mathfrak{L}$, such that for exactly one label $(\mathfrak{F}, \mathfrak{c}) \in \mathfrak{label}$ we have
   - $(\mathfrak{F}, \mathfrak{c}) \in \Gamma_0^\mathcal{T}$ and
   - $\gamma \in \Gamma_0$ implies $(\mathfrak{F}, \mathfrak{c}) \vdash \gamma \in \Gamma_0^\mathcal{T}$.
3. We then apply the tableau rules to the tableau set $\Gamma_0^\mathcal{T}$.
4. We show that $\Gamma_0^\mathcal{T}$ is $\mathcal{L}-$satisfiable (see definition 5.3.2), iff Γ_0 is satisfiable, i.e. there exists a model $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ and a situation $(F, c) \in |\mathcal{M}|$, such that $(F, c) \models_\mathcal{M} \gamma$ for all $\gamma \in \Gamma_0$.

Theorem 5.3.5 *Let* $\Gamma \subset \mathcal{L}$ *be a finite set of formulae of* $\mathcal{L}$ *and let* $\Gamma_{\mathcal{T}} \in \mathfrak{L}$ *be a finite set of tableau formulae of* $\mathfrak{L}$*, such that*

- $\mathfrak{label}(\Gamma_{\mathcal{T}}) = \{\mathfrak{l}_{\mathfrak{o}}\}$ *and*
- $\gamma \in \Gamma$ *iff* $\mathfrak{l}_{\mathfrak{o}} \vdash \gamma \in \mathfrak{L}$.

Then Γ *is satisfiable iff* $\Gamma_{\mathcal{T}}$ *is* $\mathcal{L}-$ *satisfiable.*

Proof:

1. Suppose, Γ is satisfiable.
   $\Longrightarrow$ there exists a model $\mathcal{M}$ and a situation $(\mathsf{F}, \mathsf{c}) \in |\mathcal{M}|$ such that for all $\gamma \in \Gamma:\ (\mathsf{F}, \mathsf{c}) \models_{\mathcal{M}} \gamma$
   $\Longrightarrow$ construct an $\mathcal{L}$-embedding $\ell : \mathfrak{label}(\Gamma_{\mathcal{T}}) \longrightarrow |\mathcal{M}|$ such that for all $\ell(\mathfrak{l}_{\mathfrak{o}}) = (\mathsf{F}, \mathsf{c})$
   $\Longrightarrow$ (* $\Gamma_{\mathcal{T}} \cap \mathfrak{L}_{\mathfrak{s}} = \emptyset$ *)
   ℓ is a satisfying $\mathcal{L}$-embedding for $\Gamma_{\mathcal{T}}$
   $\Longrightarrow$ $\Gamma_{\mathcal{T}}$ is $\mathcal{L}-$ satisfiable.
2. Suppose, $\Gamma_{\mathcal{T}}$ is $\mathcal{L}-$ satisfiable.
   $\Longrightarrow$ there exists a satisfying $\mathcal{L}$-embedding $\ell : \mathfrak{label}(\Gamma_{\mathcal{T}}) \longrightarrow |\mathcal{M}|$
   $\Longrightarrow$ for $\ell(\mathfrak{l}_{\mathfrak{o}}) = (\mathsf{F}, \mathsf{c})$ it holds that $(\mathsf{F}, \mathsf{c}) \models_{\mathcal{M}} \gamma$ for all $\gamma \in \Gamma$
   $\Longrightarrow$ Γ is satisfiable.

□

5.4 Tableau Rules

Our tableau system does not only break down formulae by logical implications. The structure formulae of the extended language $\mathfrak{L}$ take care of the correct structure of a potential model. Thus we can say that by breaking down formulae, at the same time, the tableau system "constructs" a part of a potential model. This means, that some of our rules are related to logical implications, while others are related to the requirements on the structure of possible models.

To make sure, that all necessary derivations are covered by the proof-system, we have to apply the rules given in section 5.4 in a certain order. We define the following three blocks of tableau rules:

1. • axioms for unsatisfiability and rules for logical implication (rules in section 5.4.1);

- rules that are related to the semantics for next (rules, which do not create new next successors) (rules in section 5.4.3);

2.
- (most) rules that are related to the structure of the $\mathsf{R}_{\mathsf{i}}^{\mathsf{K}}$-indistinguishability relations and the semantics of the knowledge operator (all rules in section 5.4.2, except for rule (**TR 9**), which belongs to block 3);
- rules that are related to the interaction between knowledge and actions (rules in section 5.4.4);

3. rules, that create new labels and are related to the structural requirements:
   - a rule that, semantically speaking, extends the temporal relation (rule 5.4.3), and
   - a rule that, semantically speaking, extends the indistinguishability relation ((**TR 9**).

Starting with the first block, we have to apply rules from each block until no further rules can be applied any more, then we can shift to the next block of rules.

This method ensures some kind of fairness for the application of tableau rules: In proposition 5.6.14 we will see that whenever a tableau rule can be applied to the tableau set occurring in a node of a tableau, then on each infinite tableau path starting from this node this rule will be served after a finite number of steps.

Let us now present the tableau rules.

Each tableau rule can be read as *The numerator is $\mathcal{L}-$ satisfiable iff one of the denominators is $\mathcal{L}-$satisfiable.* Note, that the *if and only if* in the previous sentence indicates our goal: We want to prove soundness and completeness of the system.

Let us fix a set of tableau labels $\mathfrak{label}$ for the remainder of this chapter. Whenever a denominator Γ_{den} contains a tableau label $\mathfrak{l}$ that is not explicitly mentioned in the numerator Γ_{num} of the rule, then this label needs to be *new* to the tableau set, i.e. $\mathfrak{l} \notin \mathfrak{label}(\Gamma_{\mathsf{num}})$. This constraint applies for the rules (**TR 9**) and (**TR 14**).

Furthermore, in some rules, we put formulae "asleep" which means, they may not be used any more as principal formulae for the rules (**TR 6**), (**TR 9**) and (**TR 14**). We mark these formulae as "sleeping" by coloring them.

If not stated otherwise, all formulae carry the same marking status in the denominators as they do in the numerator. Newly created tableau formulae do not carry any marks. Rules can be applied to marked formulae as well as to unmarked formulae. However, there are exceptions of this condition:

- Rules (**TR 6**) and (**TR 9**) may only be applied to unmarked knowledge formulae but they mark the respective knowledge formulae in the denominator.

- Rule (**TR 13**) may be applied to both marked and unmarked knowledge formulae and keep the marking status for the respective knowledge formula in the denominator. However, additionally it adds a new marked knowledge formula to the denominator.

- Rule (**TR 14**) may only be applied to unmarked tableau labels. The application of the rules marks the tableau label for which the rule is applied. This ensures, that for each situation, we will only have one successor for each operation.
  Rule (**TR 14**) is the only rule in which a tableau label as a formula plays a relevant role.

5.4.1 Axioms and Local Rules

Axioms for Unsatisfiability

Axioms identify sets of tableau formulae that are not $\mathcal{L}-$satisfiable, which means, there does not exist a satisfying $\mathcal{L}$-embedding for them.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash\phi,(\mathfrak{F},\mathfrak{c})\vdash\neg\phi}{} \quad (\mathbf{TR\ 1}) \qquad\qquad \frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash\bot}{} \quad (\mathbf{TR\ 2})$$

Rules Referring to Logical Implication

Next we present the rules that refer to logical implication. As we will see, these rules break down formulae local to one situation. They are standard for most tableau procedures and thus do not need further explanation.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash\top}{\Gamma_{\mathcal{T}}} \quad (\mathbf{TR\ 3})$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash\phi\wedge\psi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash\phi,(\mathfrak{F},\mathfrak{c})\vdash\psi} \quad (\mathbf{TR\ 4}) \qquad\qquad \frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash\phi\vee\psi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash\phi \ \mid\ \Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash\psi} \quad (\mathbf{TR\ 5})$$

Local Knowledge Rule (Reflexivity)

The following rule is again a local rule. It corresponds to the *reflexivity* of the R_i^K indistinguishability relation: If a situation (F, c) of a model $\mathcal{M}$ satisfies a formula $K_i\phi$, then because of the reflexivity of the knowledge indistinguishability relation it also satisfies formula ϕ. For each tableau formula $(\mathfrak{F}, \mathfrak{c}) \vdash K_i\phi$ in the tableau set we thus add a labelled tableau formula $(\mathfrak{F}, \mathfrak{c}) \vdash \phi$ to the set.

We do not remove the principal $(\mathfrak{F}, \mathfrak{c}) \vdash K_i\phi$ from the tableau set because it has impacts on other labels (situations) that might not be identifiable at the moment of the rule application. So the principal formula $(\mathfrak{F}, \mathfrak{c}) \vdash \phi$ will be a marked side formula in the denominator of the rule. The marking prevents it from being principal formula again for the same rule. For the application of rule (**TR 6**) we require the principal formula to be unmarked.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash K_i\phi \ \text{(unmarked)}}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash K_i\phi, (\mathfrak{F}, \mathfrak{c}) \vdash \phi} \qquad \textbf{(TR 6)}$$

Induction Rules

The next two rules are the rules for the unwinding of the (weak and strong) until operators. If the logic $\mathcal{L}$ contained a global, unlabelled next (often denoted as $\bigcirc$), a formula $\phi\ \mathcal{U}\psi$ could be unwound to two alternative successors ψ and $\neg\psi \wedge \phi \wedge \bigcirc\phi\ \mathcal{U}\psi$. However, such a global next does not exist in $\mathcal{L}$. We use the abbreviation $\langle\mathcal{O}\rangle_i\phi\ \mathcal{U}_i\psi := \bigvee_{a\in\mathcal{O}_i}\langle a\rangle_i\phi\ \mathcal{U}_i\psi$ instead that indicates, that there must be at least one next operation of agent i and after this operation the until formula must hold again. Note, that this abbreviation is merely a syntactic one and is possible only because the set of operations of each agent $\mathcal{O}_i$ is finite. In the case of $\phi\mathcal{W}_i\psi$ we cannot assume that there actually exists a next operation: $\phi\mathcal{W}_i\psi$ is also satisfied in a situation in which agent i does not perform any further operations. However, *if* there is a next operation of agent i then $\phi\mathcal{W}_i\psi$ must hold after the next operation is performed. We abbreviate this by $[\mathcal{O}]_i\phi\mathcal{W}_i\psi := \bigwedge_{a\in\mathcal{O}_i}[a]_i\phi\mathcal{W}_i\psi$.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \phi\ \mathcal{U}_i\psi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \psi \quad | \quad \Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i\phi\ \mathcal{U}_i\psi} \qquad \textbf{(TR 7)}$$

(where $\langle\mathcal{O}\rangle_i\phi$ abbreviates $\bigvee_{a\in\mathcal{O}_i}\langle a\rangle_i\phi$)

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash \phi\mathcal{W}_i\psi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash\psi \quad | \quad \Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})\vdash\neg\psi\wedge\phi\wedge[\mathcal{O}]_i\phi\mathcal{W}_i\psi} \qquad (\textbf{TR 8})$$

(where $[\mathcal{O}]_i\phi$ abbreviates $\bigwedge_{a\in\mathcal{O}_i}[a]_i\phi$)

5.4.2 Rules Concerning Epistemic Structure

As mentioned in the introduction to this section, structure formulae refer to the structure of the indistinguishability relations for knowledge and to the temporal relation of a potential model.

The rules in this section take care of the correct representation of the indistinguishability relations by tableau formulae and tableau rules.

- A formula $\neg K_i\phi$ is true in a situation (F,c), iff there exists an R_i^K-indistinguishable situation (F',c') in which ϕ does not hold. Suppose, $\Gamma_{\mathcal{T}}$ contains a tableau formula $(\mathfrak{F},\mathfrak{c})\vdash\neg K_i\phi$. We know that if we embed the label $(\mathfrak{F},\mathfrak{c})$ into a model, the model must contain at least one situation $(\mathfrak{F}',\mathfrak{c}')$ that is R_i^K-indistinguishable from $\ell((\mathfrak{F},\mathfrak{c}))$ and that does not satisfy ϕ.

  We cannot assume that the tableau label $\mathfrak{l}$, that is mapped on the situation (F',c'), has already occurred in $\mathfrak{label}(\Gamma_{\mathcal{T}})$. Therefor, we must introduce a new label in rule (**TR 9**). For this new label $(\mathfrak{F}',\mathfrak{c}')$ we require, that

  - the situation in which it gets embedded is R_i^K-indistinguishable from the situation (F,c) (in the tableau indicated by $(\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}',\mathfrak{c}')$)
  - it does not satisfy ϕ (in the tableau indicated by $(\mathfrak{F}',\mathfrak{c}')\vdash\neg\phi$).

  To ensure, that we do not apply this rule infinitely many times and thereby create infinitely many labels that satisfy the above requirements, we put the formula $(\mathfrak{F},\mathfrak{c})\vdash\neg K_i\phi$ asleep by marking it.

  Again, as before in rule (**TR 6**) we require an unmarked principal formula for this rule.

- The R_i^K-indistinguishability relation is an equivalence relation. Rule (**TR 6**) already takes care for the reflexivity of the structure. We define tableau rules that ensure symmetry and transitivity for the indistinguishability relations. If a set of tableau formulae $\Gamma_{\mathcal{T}}$ contains a structure formula $(\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}',\mathfrak{c}')$

we add the structure formula $(\mathfrak{F}', \mathfrak{c}')\mathsf{R}_i(\mathfrak{F}, \mathfrak{c})$ and if it contains the formulae $(\mathfrak{F}, \mathfrak{c})\mathsf{R}_i(\mathfrak{F}', \mathfrak{c}')$ and $(\mathfrak{F}', \mathfrak{c}')\mathsf{R}_i(\mathfrak{F}'', \mathfrak{c}'')$, we add the structure formula $(\mathfrak{F}, \mathfrak{c})\mathsf{R}_i(\mathfrak{F}'', \mathfrak{c}'')$.

- On the semantical side we have that two $\mathsf{R}_i^{\mathsf{K}}$-indistinguishable situations satisfy the same set of knowledge formulae of agent i: If two situations (F, c) and $(\mathsf{F}', \mathsf{c}') \in |\mathcal{M}|$ are $\mathsf{R}_i^{\mathsf{K}}$-indistinguishable ($((\mathsf{F}, \mathsf{c}), (\mathsf{F}', \mathsf{c}')) \in \mathsf{R}_i^{\mathsf{K}}$) then $(\mathsf{F}, \mathsf{c}) \models_{\mathcal{M}} \mathsf{K}_i\phi$ iff $(\mathsf{F}', \mathsf{c}') \models_{\mathcal{M}} \mathsf{K}_i\phi$ and, accordingly $(\mathsf{F}, \mathsf{c}) \models_{\mathcal{M}} \neg\mathsf{K}_i\phi$ iff $(\mathsf{F}', \mathsf{c}') \models_{\mathcal{M}} \neg\mathsf{K}_i\phi$ for all formulae $\phi \in \mathcal{L}$. We define two syntactic rules that correlate to this semantic fact: If we have the tableau formulae $(\mathfrak{F}, \mathfrak{c}) \vdash \mathsf{K}_i\phi$ and $(\mathfrak{F}, \mathfrak{c})\mathsf{R}_i(\mathfrak{F}', \mathfrak{c}')$ then we introduce a new tableau formula $(\mathfrak{F}', \mathfrak{c}') \vdash \mathsf{K}_i\phi$ and, accordingly, if we have two tableau formulae $(\mathfrak{F}, \mathfrak{c}) \vdash \neg\mathsf{K}_i\phi$ and $(\mathfrak{F}, \mathfrak{c})\mathsf{R}_i(\mathfrak{F}', \mathfrak{c}')$ then we introduce a new tableau formula $(\mathfrak{F}', \mathfrak{c}') \vdash \neg\mathsf{K}_i\phi$.

  As mentioned before, knowledge formulae can occur marked and unmarked in the tableau. The rules for the transfer of knowledge formulae across R_i-edges/paths can be applied to both marked and unmarked formulae. We marked a tableau formula $(\mathfrak{F}, \mathfrak{c}) \vdash \mathsf{K}_i\phi$ to prevent rule (**TR 6**) to be applied infinitely many times to the same formula. This marking is only relevant in the context of this rule. That means for the rule for transferring knowledge formulae, that the principal formula $(\mathfrak{F}, \mathfrak{c}) \vdash \mathsf{K}_i\phi$ will remain marked, iff it was marked before and remain unmarked, iff it was unmarked before. The newly added tableau formula $(\mathfrak{F}', \mathfrak{c}') \vdash \mathsf{K}_i\phi$ will be unmarked: rule (**TR 6**) has not been applied to this tableau formula yet.

  The case is different for formulae $(\mathfrak{F}, \mathfrak{c}) \vdash \neg\mathsf{K}_i\phi$. Here we use the marking to indicate, that already one label $\mathfrak{l}$ in this "R_i-equivalence" class exists, with $\mathfrak{l} \vdash \neg\phi$. Thus, in this rule, the newly added tableau formula $(\mathfrak{F}', \mathfrak{c}') \vdash \neg\mathsf{K}_i\phi$ will be marked. If the principal of the rule, i.e. tableau formula $(\mathfrak{F}, \mathfrak{c}) \vdash \neg\mathsf{K}_i\phi$, is marked as well, then there either already exists a tableau formula $(\mathfrak{F}', \mathfrak{c}') \vdash \neg\phi$ in this R_i-equivalence class or there is at least one more tableau formula, that is responsible to produce such a tableau formula. If the principal formula is unmarked, this means, that it will sooner or later create a formula $(\mathfrak{F}', \mathfrak{c}') \vdash \neg\phi$ with $(\mathfrak{F}', \mathfrak{c}')$ being in the same R_i-equivalence class as $(\mathfrak{F}, \mathfrak{c})$.

- The interpretation of a model is defined in such a way, that each proposition $p \in \mathcal{P}$ has the same value in all $\mathsf{R}_i^{\mathsf{K}}$-indistinguishable situations. Therefor, rule (**TR 12**) propagates propositions (or their negations) to labels that belong to the same R_i equivalence class.

Creation of New R_i-successors

The following rule corresponds to the first item above. For each **awake**, i.e. unmarked tableau formula $(\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i\phi \in \Gamma_l$ we employ a **new** tableau label $(\mathfrak{F}', \mathfrak{c}')$ and relate it to tableau label $(\mathfrak{F}, \mathfrak{c})$ my means of a structure formula $(\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}')$. For this new tableau label we further add the labelled formula $(\mathfrak{F}', \mathfrak{c}') \vdash \neg\phi$. Intuitively, on the semantical level this means that we identify a situation (F', c') that is R_i^K-indistinguishable to situation (F, c) and takes care, that the formula $\neg\phi$ is satisfied in at least one situation of the equivalence class. Further, since we require for a tableau set that $\Gamma_\mathcal{T} \cap \mathfrak{label} = \mathfrak{label}(\Gamma_\mathcal{T})$ we must add the tableau label $(\mathfrak{F}', \mathfrak{c}')$ itself as a formula to the denominator.

As explained above, the application of this rule marks the principal formula to indicate, that, semantically spoken, there already exists one situation in the equivalence class, which satisfies $\neg\phi$. For the application of rule (**TR 9**) we require the principal formula to be unmarked. The principal formula occurs as *marked* side formula in the denominator.

$$\frac{\Gamma_\mathcal{T},\ (\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i\phi \quad \text{(unmarked)}}{\Gamma_\mathcal{T},\ (\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i\phi, (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}', \mathfrak{c}') \vdash \neg\phi} \qquad (\textbf{TR 9})$$

Transitive and Symmetric Closure of the R_i-relation

The next two rules construct the symmetric and transitive closure of the R_i-relations.

$$\frac{\Gamma_\mathcal{T},\ (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}')}{\Gamma_\mathcal{T},\ (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}', \mathfrak{c}')R_i(\mathfrak{F}, \mathfrak{c})} \qquad (\textbf{TR 10})$$

$$\frac{\Gamma_\mathcal{T},\ (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}', \mathfrak{c}')R_i(\mathfrak{F}'', \mathfrak{c}'')}{\Gamma_\mathcal{T},\ (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}', \mathfrak{c}')R_i(\mathfrak{F}'', \mathfrak{c}''), (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}'', \mathfrak{c}'')} \qquad (\textbf{TR 11})$$

Transfer of Formulae Between R_i-indistinguishable Situations

As motivated in items 3 and 4 above, the rules (**TR 12**) and (**TR 13**) ensure, that in R_i-equivalent situations the same knowledge formulae and the same i-local propositions hold.

In rule (**TR 12**) we take *any marked or unmarked* formula of the form $(\mathfrak{F}, \mathfrak{c}) \vdash \chi$ with $\chi \in \{K_i\phi, P, \neg P \mid P \in \mathcal{P}_i, \phi \in \mathcal{L}\}$ and transfer it to an R_i related tableau label $(\mathfrak{F}', \mathfrak{c}')$, omitting any marking in the new formula $(\mathfrak{F}', \mathfrak{c}') \vdash \chi$. In other words, the rule can be applied to both marked and unmarked tableau formulae. The newly generated side formula in the denominator is unmarked, independent of the marking of the principal formula.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}, \mathfrak{c}) \vdash \chi \qquad \text{for } \chi \in \{K_i\phi, P, \neg P | P \in \mathcal{P}_i, \phi \in \mathcal{L}\}}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}, \mathfrak{c}) \vdash \chi, (\mathfrak{F}', \mathfrak{c}') \vdash \chi} \qquad (\textbf{TR 12})$$

The next rule can again be applied to both *marked and unmarked* formulae $(\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i\phi$. The marking status of the side formula $(\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i\phi$ will be the same as that of the principal formula $(\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i\phi$. However, the side formula $(\mathfrak{F}', \mathfrak{c}') \vdash \neg K_i\phi$ in the denominator will be marked, independently of the marking status of the principal formula. This marking ensures, that the side formula $(\mathfrak{F}', \mathfrak{c}') \vdash \neg K_i\phi$ does not create another (unnecessary) R_i-successor through rule (**TR 9**).

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i\phi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i\phi, (\mathfrak{F}', \mathfrak{c}') \vdash \neg K_i\phi} \qquad (\textbf{TR 13})$$

5.4.3 Rules Concerning Temporal Structure

The rules in this section take care of the correct representation of the temporal successor relation of a potential model by tableau formulae and tableau rules.

Creation of Next-Successors

If a formula $\langle a \rangle_i \phi$ with $a \in \mathcal{O}_i$ or a formula $\langle \mathcal{O} \rangle_i \phi$ is satisfied in a situation (F, c) of a potential model, then there must exist a successor situation (F, c') of (F, c) and an operation $a' \in \mathcal{O}$ such that $(F, c) \xrightarrow{a'} (F, c')$. (From a formula $\langle a \rangle_i \phi$ we can even conclude a stronger condition, namely that there exists a situation (F, ca) with $(F, c) \xrightarrow{a} (F, ca)$. However, in the context of this rule the weaker statement made above is sufficient.) Rule (**TR 14**) takes care of the creation of necessary next-successors.

Let us in the following abbreviate "$\langle \mathcal{O} \rangle_i \phi$ or $\langle a \rangle_i \phi$ for any $a \in \mathcal{O}_i$" by $\langle \cdot \rangle_i \phi$.

Although the model for the logic was defined with a partial order relation for the temporal relations, the tableau works in an interleaving fashion. The set of tableau formulae in the numerator contains a set of chains as defined in definition 5.3.4. Semantically speaking, each chain represents one interleaving of a potential model. If the tableau set Γ_{num} contains a formula $(\mathfrak{F}, \mathfrak{c}) \vdash \langle\cdot\rangle_i \phi$ and rule (**TR 14**) has not been applied for label $(\mathfrak{F}, \mathfrak{c})$ yet, i.e. label $(\mathfrak{F}, \mathfrak{c})$ is unmarked, then rule (**TR 14**) creates a denominator for each operation[1] $a' \in \mathcal{O}$. The denominator contains a new formula $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a'} (\mathfrak{F}, \mathfrak{c}a')$ for the respective operation a'. Further, rule (**TR 14**) marks the tableau label $(\mathfrak{F}, \mathfrak{c})$ so that the rule (**TR 14**) can be applied only once for each label. This prevents, that there are two structure formula $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a)$ and $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a'} (\mathfrak{F}, \mathfrak{c}a')$ in the same tableau set with $a \neq a'$ or $(\mathfrak{F}, \mathfrak{c}a) \neq (\mathfrak{F}, \mathfrak{c}a')$.

The following rule may be applied only for tableau labels $(\mathfrak{F}, \mathfrak{c})$ that are unmarked. The application of this rule generates one denominator for each operation $a_i \in \mathcal{O}$. The tableau label $(\mathfrak{F}, \mathfrak{c}a_i)$ in the ith denominator must be new to the tableau set.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \langle . \rangle_i \phi, (\mathfrak{F}, \mathfrak{c}) \qquad \text{(unmarked)}}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \langle . \rangle_i \phi, (\mathfrak{F}, \mathfrak{c}) \xrightarrow{a_1} (\mathfrak{F}, \mathfrak{c}a_1), (\mathfrak{F}, \mathfrak{c}), (\mathfrak{F}, \mathfrak{c}a_1) \quad | \dots | \quad \Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \langle . \rangle_i \phi, (\mathfrak{F}, \mathfrak{c}) \xrightarrow{a_n} (\mathfrak{F}, \mathfrak{c}a_n), (\mathfrak{F}, \mathfrak{c}), (\mathfrak{F}, \mathfrak{c}_n)} \quad \textbf{(TR 14)}$$

Saturation of Temporal Successors

In the previous rule, we have created possible next-successors for tableau labels. Semantically speaking, a structure formula $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a) \in \Gamma_{\mathcal{T}}$ indicates, that in a model the situation $\ell((\mathfrak{F}, \mathfrak{c}a))$, corresponding to tableau label $(\mathfrak{F}, \mathfrak{c}a)$, is reachable from the situation $\ell((\mathfrak{F}, \mathfrak{c}))$, corresponding to tableau label $(\mathfrak{F}, \mathfrak{c})$, by execution of an operation a.

If $(F, c a)$ is the situation resulting from performing an operation a in the situation (F, c) and if the situation (F, c) satisfies a formula $\langle a \rangle_i \phi$ or $[a]_i \phi$ then the resulting situation $(F, c a)$ satisfies the formula ϕ. These considerations lead to the following tableau rules:

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}) \vdash \langle a \rangle_i \phi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}a) \vdash \phi} \qquad \textbf{(TR 15)}$$

[1] Note, that this is possible only because $\mathcal{O}$ is finite.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{\mathfrak{a}} (\mathfrak{F},\mathfrak{c}\mathfrak{a}), (\mathfrak{F},\mathfrak{c}) \vdash [\mathfrak{a}]_{\mathfrak{i}}\phi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{\mathfrak{a}} (\mathfrak{F},\mathfrak{c}\mathfrak{a}), (\mathfrak{F},\mathfrak{c}\mathfrak{a}) \vdash \phi} \tag{TR 16}$$

If a situation (F, c) evolves to a situation (F, ca) by the performance of an operation a, then the situation (F, c) cannot satisfy a formula $\langle b\rangle_i\phi$ with $i \in ag(a)$ and $a \neq b$ because the operations for each agent are totally ordered. In the tableau we capture this contradiction by adding a tableau formula $(\mathfrak{F}, \mathfrak{c}) \vdash \perp$. Independently of the sub formula ϕ if the situation (F, c) evolves to (F, ca) by the performance of an operation a, then situation (F, c) satisfies the formula $[a]_i\phi$ where $i \in ag(a)$. We capture these considerations in the following tableau rules:

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{\mathfrak{a}} (\mathfrak{F},\mathfrak{c}\mathfrak{a}), (\mathfrak{F},\mathfrak{c}) \vdash \langle\mathfrak{b}\rangle_{\mathfrak{i}}\phi \ \text{ with } i \in ag(a), a \neq b}{(\mathfrak{F},\mathfrak{c}) \vdash \perp} \tag{TR 17}$$

If a situation (F, c) evolves to a situation (F, ca) by performing an action a and if situation (F, c) satisfies a formula $\langle b\rangle_i\phi$ (or a formula $[b]_i\phi$) with $i \notin ag(a)$, then the situation (F, ca) also satisfies the formula $\langle b\rangle_i\phi$ (or the formula $[b]_i\phi$, respectively). From these considerations we conclude the following two rules:

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{\mathfrak{a}} (\mathfrak{F},\mathfrak{c}\mathfrak{a}), (\mathfrak{F},\mathfrak{c}) \vdash \langle\mathfrak{b}\rangle_{\mathfrak{i}}\phi \ \text{ with } i \notin ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{\mathfrak{a}} (\mathfrak{F},\mathfrak{c}\mathfrak{a}), (\mathfrak{F},\mathfrak{c}\mathfrak{a}) \vdash \langle\mathfrak{b}\rangle_{\mathfrak{i}}\phi} \tag{TR 18}$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{\mathfrak{a}} (\mathfrak{F},\mathfrak{c}\mathfrak{a}), (\mathfrak{F},\mathfrak{c}) \vdash [\mathfrak{b}]_{\mathfrak{i}}\phi \ \text{ with } i \notin ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{\mathfrak{a}} (\mathfrak{F},\mathfrak{c}\mathfrak{a}), (\mathfrak{F},\mathfrak{c}\mathfrak{a}) \vdash [\mathfrak{b}]_{\mathfrak{i}}\phi} \tag{TR 19}$$

Because of the rules for until, $\mathcal{U}$, and for the weak until, $\mathcal{W}$, we introduced artificial global next $\langle\mathcal{O}\rangle_i\phi$ and $[\mathcal{O}]_i\phi$ as abbreviations for quantifications over all operations of $\mathcal{O}_i$. As for local next, we distinguish between the two cases, that agent i does take part in operation a and that agent i does not take part.

First consider, that agent i takes part in operation a:

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \langle\mathcal{O}\rangle_i\phi \text{ with } i \in ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash \phi} \qquad \textbf{(TR 20)}$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash [\mathcal{O}]_i\phi \text{ with } i \in ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash \phi} \qquad \textbf{(TR 21)}$$

The next rules consider operations a in which agent i does not take part:

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \langle\mathcal{O}\rangle_i\phi \text{ with } i \notin ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash \langle\mathcal{O}\rangle_i\phi} \qquad \textbf{(TR 22)}$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash [\mathcal{O}]_i\phi \text{ with } i \notin ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash [\mathcal{O}]_i\phi} \qquad \textbf{(TR 23)}$$

5.4.4 Interaction Between Knowledge and Action

Not all operations can be observed by all agents, usually only a subset of all agents takes part in an operation. For those agents i, that do not take part, the situations before and after the execution of the operation are i-equivalent. Condition 3 of definition 3.3.2 says, that i-equivalent situations are also R_i^K-indistinguishable. This leads to the following rule, which has the structure formula $(\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}')$ as principal formula in the numerator and the set of formulae $\{(\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}')\} \cup \{(\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F},\mathfrak{c}') | i \notin ag(a)\}$ as side formulae in the denominator.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F},\mathfrak{c}a) \text{ for all } i \notin ag(a)} \qquad \textbf{(TR 24)}$$

The next two rules correspond to condition 4 of definition 3.3.2. This condition makes the tableau rules quite complex. It ensures, that if a situation (F, c) evolves into situation (F, ca) by the execution of an operation a and there is a situation

(F', c') R_i^K-indistinguishable from (F, c) for *all* agents i involved in a, then operation a must also be possible in situation (F', c'). Tableau rule (**TR 25**) captures this constraint in the tableau. It can be applied, if the tableau set Γ_{num} contains a formula $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a)$ and it contains a tableau label $(\mathfrak{F}', \mathfrak{c}')$ such that for all $i \in ag(a)$ the formula $(\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}, \mathfrak{c}a) \in \Gamma_{num}$. The denominator Γ_{den} of rule (**TR 25**) is a superset of the numerator and additionally to the numerator it contains for all $i \in ag(a)$ the formula $(\mathfrak{F}', \mathfrak{c}') \vdash \langle a \rangle_i \top$.

Further, condition 4 of definition 3.3.2 requires, that after performing the a-step from situation (F', c') the resulting situation $(F', c'a)$ must be indistinguishable from situation (F, ca) for all agents i involved in operation a. This constraint leads to rule (**TR 26**). Rule (**TR 26**) can be applied if the tableau set Γ_{num} contains formulae $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a)$ and $(\mathfrak{F}', \mathfrak{c}') \xrightarrow{a} (\mathfrak{F}', \mathfrak{c}'a)$ and the set $\{(\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}, \mathfrak{c}a) \mid i \in ag(a)\}$ as principal formulae. The denominator Γ_{den} of rule (**TR 26**) is a superset of the numerator: and additionally to the numerator it contains the set $\{(\mathfrak{F}, \mathfrak{c}a)R_i(\mathfrak{F}', \mathfrak{c}'a) \mid i \in ag(a)\}$ as side formulae.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}') \ \text{ for all } i \in ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}', \mathfrak{c}') \vdash \langle a \rangle_i \top \ \text{ for all } i \in ag(a)} \quad (\textbf{TR 25})$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}', \mathfrak{c}') \xrightarrow{a} (\mathfrak{F}', \mathfrak{c}'a), (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}') \ \text{ for all } i \in ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}', \mathfrak{c}') \xrightarrow{a} (\mathfrak{F}', \mathfrak{c}'a), (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}, \mathfrak{c}a)R_i(\mathfrak{F}', \mathfrak{c}'a) \ \text{ for all } i \in ag(a)} \quad (\textbf{TR 26})$$

5.5 Definition of Tableaux

In this section we define tableaux for finite tableau sets $\Gamma_{\mathcal{T}} \subset \mathfrak{L}$. We distinguish between *general tableaux*, the root of which is labelled by an arbitrary tableau set $\Gamma_{\mathcal{T}}$ and *regular tableaux*. The root of a regular tableau is labelled by a tableau set $\Gamma_0^{\mathcal{T}}$ that is constructed from a set of formulae Γ_0 by construction τ. Note, that the root of a regular tableau does not contain any structure formulae (i.e. $\Gamma_0^{\mathcal{T}} \cap \mathfrak{L}_s = \emptyset$) and all labelled formulae in $\Gamma_0^{\mathcal{T}}$ carry the same label (i.e. $|\mathfrak{label}(\Gamma_0^{\mathcal{T}})| = 1$).

Many of the properties we show for the tableaux only apply to regular tableaux. We assume a certain structure of the set of tableau formulae occurring in the tableau that is guaranteed only for sets of formulae occurring in regular tableaux.

Definition 5.5.1 (general tableau) *A (general) tableau $\mathcal{T}(\Gamma_\mathcal{T})$ is a labelled, possibly infinite tree of finite grade where every node ν of the tree is labelled*[2] *with a finite tableau set $\Gamma(\nu)$, such that the following holds:*

- *The root is labelled with $\Gamma_\mathcal{T}$.*
- *If $\Gamma(\nu)$ is a tableau axiom, then ν is a leaf of the tree.*
- *If $\Gamma(\nu)$ is not an axiom, then*
  - *either no tableau rule is applicable to $\Gamma(\nu)$ and thus ν is a leaf or*
  - *the children of ν are created and labelled according to the rules:*
    $\Gamma(\nu)$ is the numerator and the labels of the children are the denominators of the rule.
    Three blocks of rule applications alternate:
    block 1 *= rules (***TR 3***), (***TR 4***), (***TR 5***), (***TR 6***), (***TR 7***), (***TR 8***), (***TR 15***), (***TR 16***), (***TR 17***), (***TR 18***), (***TR 19***), (***TR 20***), (***TR 21***), (***TR 22***), (***TR 23***),*
    block 2 *= rules (***TR 10***), (***TR 11***), (***TR 12***), (***TR 13***), (***TR 24***),(***TR 25***) and (***TR 26***)*
    block 3 *= rules (***TR 9***) and (***TR 14***)*
    * *The children of ν are generated by an application of a rule of block 1, if a rule of block 1 is applicable and if*
      · *ν is the root of the tree or*
      · *if ν is generated from ν' by the application of a rule of block 1 or*
      · *if ν is generated from ν' by the application of a rule of block 3 and no further application of a rule of block 3 is possible, or*
      · *if ν is generated from ν' by the application of a rule of block 2 and no further rules of block 2 or block 3 are applicable.*
    * *The children of ν are generated by the application of a rule of block 2, if a rule of block 2 is applicable and if*
      · *ν is generated from ν' by the application of a rule of block 2 or*
      · *if ν is generated by the application of a rule of block 1 and no further rules of block 1 can be applied any more, or*
      · *if ν is generated from ν' by the application of a rule of block 3 and no further rules of block 3 or block 1 are applicable.*

[2]Note, that we use the word labelled in the context of labelled formulae, labelled events and labelled tableau nodes. However, it should always be clear from the context which type of label we consider.

  * *The children of ν are generated by the application of a rule of block 3, if a rule of block 3 is applicable and if*
    - ν *is generated from* ν' *by the application of a rule of block 3 or*
    - *if* ν *is generated by the application of a rule of block 2 and no further rules of block 2 can be applied any more, or*
    - *if* ν *is generated from* ν' *by the application of a rule of block 1 and no further rules of block 1 or block 2 are applicable.*

- *Rules are applied provided that* **all** *their denominators are different from the numerator.*

We define the set of tableau formulae occurring on a path $\pi = \nu_0\nu_1\nu_2\ldots$ *as*

$$\Gamma_\pi := \bigcup_{k\in\mathbb{N}} \Gamma(\nu_k).$$

We define the set of tableau formulae occurring on a prefix $\pi^n = \nu_0\nu_1\nu_2\ldots\nu_n$ *of a path* π *as*

$$\Gamma_n := \bigcup_{k\leq n} \Gamma(\nu_k).$$

A specialization of general tableaux are regular tableaux.

Definition 5.5.2 (regular tableau) *A tableau* $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ *is called* **regular tableau** *iff there exists a set of formulae* $\Gamma_0 \subset \mathcal{L}$ *such that the root* $\Gamma_0^{\mathcal{T}}$ *of* $\mathcal{T}$ *is derived from* Γ_0 *by construction* τ, *such that* $\Gamma_0^{\mathcal{T}} = \tau(\Gamma_0)$.

Since we are primarily interested in the (non-)satisfiability of sets of formulae $\Gamma_0 \in \mathcal{L}$ we will in the remainder of this work only consider regular tableaux and assume regular tableaux whenever we talk about tableaux.

For regular tableaux we can make one important observation that is essential for the completeness proof of the tableau system: Consider a path π of a regular tableau $\mathcal{T}$. The set of formulae occurring on path π, Γ_π, contains a set of structure formulae of the type $\mathfrak{l} \xrightarrow{a} \mathfrak{l}a$, which forms a set of chains.

It cannot happen that a tableau label $\mathfrak{l} \in \mathfrak{label}(\Gamma_\pi)$ has two next-successors or two next-predecessors, i.e. for all labels $\mathfrak{l}, \mathfrak{l}_1, \mathfrak{l}_2 \in \mathfrak{label}(\Gamma_\pi)$ and for all $a, b \in \mathcal{O}$ the following holds:

- If $\mathfrak{l} \xrightarrow{a} \mathfrak{l}_1$ and $\mathfrak{l} \xrightarrow{b} \mathfrak{l}_2$ then $a = b$ and $\mathfrak{l}_1 = \mathfrak{l}_2$ and
- if $\mathfrak{l}_1 \xrightarrow{a} \mathfrak{l}$ and $\mathfrak{l}_2 \xrightarrow{b} \mathfrak{l}$ then $a = b$ and $\mathfrak{l}_1 = \mathfrak{l}_2$.

Further, each chain has a least label $\mathfrak{l}$ such that there is no label $\mathfrak{l}' \in \mathfrak{label}(\Gamma_\pi)$ and operation $\mathfrak{a} \in \mathcal{O}$ with $\mathfrak{l}' \xrightarrow{\mathfrak{a}} \mathfrak{l}$. Together with the previous constraint this implies that there can be no cycles in the chains. From this we can conclude, that each tableau label occurs in exactly one maximal chain in Γ_π.

Figure 5.5 illustrates constructions, that do *not* occur:

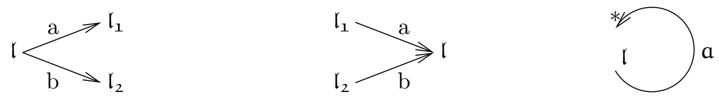


Figure 5.2: Combinations of structure formulae that cannot occur in Γ_π.

We define this property to be the **regular tableau property**:

Definition 5.5.3 (regular tableau property) *Let* $\Gamma_{\mathcal{T}}$ *be a set of tableau formulae and let* $\mathfrak{l} \longrightarrow^{+} \mathfrak{l}'$ *be the transitive closure of all tableau formulae of type* $\mathfrak{l} \xrightarrow{\mathfrak{a}} \mathfrak{l}' \in \Gamma_\pi$ *with* $\mathfrak{a} \in \mathcal{O}$.

$\Gamma_{\mathcal{T}}$ *has the* **regular tableau property** *iff it satisfies the following properties:*

1. $\mathfrak{l}_1 \xrightarrow{\mathfrak{a}} \mathfrak{l}_2 \in \Gamma_\pi$ *and* $\mathfrak{l}_1 \xrightarrow{\mathfrak{b}} \mathfrak{l}_2' \in \Gamma_\pi$ *implies* $\mathfrak{a} = \mathfrak{b}$ *and* $\mathfrak{l}_2 = \mathfrak{l}_2'$ *(no split),*
2. $\mathfrak{l}_1 \xrightarrow{\mathfrak{a}} \mathfrak{l}_2 \in \Gamma_\pi$ *and* $\mathfrak{l}_1' \xrightarrow{\mathfrak{b}} \mathfrak{l}_2 \in \Gamma_\pi$ *implies* $\mathfrak{a} = \mathfrak{b}$ *and* $\mathfrak{l}_1 = \mathfrak{l}_1'$ *(no join),*
3. $\mathfrak{l} \longrightarrow^{+} \mathfrak{l}'$ *implies* $\mathfrak{l} \neq \mathfrak{l}'$ *(no cycle).*

Lemma 5.5.4 *Let* $\mathcal{T}$ *be a regular tableau and let* π *be a path of* $\mathcal{T}$*.*

Γ_π *has the regular tableau property.*

For each finite prefix of a path π we can prove this lemma by induction over the length of the path.

The induction hypothesis is, that for each prefix $\nu_0\nu_1 \ldots \nu_n$ of the path the set $\Gamma_n = \bigcup_{k \leq n} \Gamma(\nu_k)$ has the regular tableau property.

Most tableau rules do not change the set of the structure formulae of type $\mathfrak{l} \xrightarrow{\mathfrak{a}} \mathfrak{l}\mathfrak{a}$. For these rules it is obvious that if Γ_{num} has the regular tableau property then also Γ_{den} has the regular tableau property.

The only rules that change the set of these structure formulae are rules (**TR 17**) and (**TR 14**). Rule (**TR 17**) deletes a structure formula of the form $\mathfrak{l} \xrightarrow{\mathfrak{a}} \mathfrak{l}'$. This obviously cannot lead to a violation against the lemma: the only applicable rule after application of rule (**TR 17**) is rule (**TR 2**). Thus, it cannot happen that a deleted label will be introduced again into the considered tableau set along a path.

Using a *new* label, rule (**TR 14**) adds a new structure formula of respective form. Since the new label did not occur in Γ_{num} before, the introduction of this new structure formula can neither lead to joins nor can it lead to cycles.

Each application of rule (**TR 14**) with formulae $(\mathfrak{F}, \mathfrak{c}) \vdash \langle\cdot\rangle_i\phi, (\mathfrak{F}, \mathfrak{c})$ as principal formulae marks the tableau label $(\mathfrak{F}, \mathfrak{c})$. Since the rule may only be applied to unmarked tableau labels and there is no rule, that unmarks a marked tableau label, the rule (**TR 14**) may be applied only once for each tableau label $(\mathfrak{F}, \mathfrak{c}) \in \mathfrak{label}(\Gamma_\pi)$.

Rule (**TR 14**) is the only rule that creates structure formulae of the type $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a)$, so this rule cannot create two structure formula $\mathfrak{l} \xrightarrow{a} \mathfrak{l}'$ and $\mathfrak{l} \xrightarrow{b} \mathfrak{l}''$ with $a = b$ or $l' = l''$.

We now capture this informal proof-sketch more formally:

Proof of lemma 5.5.4
Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be a regular tableau and let π be a path of $\mathcal{T}(\Gamma_0^{\mathcal{T}})$.

Suppose, Γ_π does not have the regular tableau property. Then, Γ_π must either contain

- two structure formulae $\mathfrak{l}_1 \xrightarrow{a} \mathfrak{l}_2$ and $\mathfrak{l}_1 \xrightarrow{b} \mathfrak{l}_2'$ with $a \neq b$ or $\mathfrak{l}_2 \neq \mathfrak{l}_2'$ or
- it must contain two structure formulae $\mathfrak{l}_1 \xrightarrow{a} \mathfrak{l}_2$ and $\mathfrak{l}_1' \xrightarrow{b} \mathfrak{l}_2$ with $a \neq b$ or $\mathfrak{l}_1 \neq \mathfrak{l}_1'$ or
- it must contain a cycle $\mathfrak{l} \longrightarrow^+ \mathfrak{l}'$ with $\mathfrak{l} = \mathfrak{l}'$.

1. Suppose, $\mathfrak{l}_1 \xrightarrow{a} \mathfrak{l}_2 \in \Gamma_\pi$ and $\mathfrak{l}_1 \xrightarrow{b} \mathfrak{l}_2' \in \Gamma_\pi$ with $\mathfrak{l}_2 \neq \mathfrak{l}_2'$ or $a \neq b$.
   $\Longrightarrow$ there must exist a least prefix $v_0 \ldots v_m \ldots v_n$ of path π such that
   a) the set $\Gamma(v_m)$ is derived from $\Gamma(v_{m-1})$ by the application of rule (**TR 14**) and w.l.o.g. $\mathfrak{l}_1 \xrightarrow{a} \mathfrak{l}_2 \in \Gamma_m \setminus \Gamma_{m-1}$ and
   b) $\Gamma(v_n)$ is derived from $\Gamma(v_{n-1})$ by the application of rule (**TR 14**) and $\mathfrak{l}_1 \xrightarrow{b} \mathfrak{l}_2' \in \Gamma_n \setminus \Gamma_{n-1}$

   $\Longrightarrow$ for all $k \geq m$ it holds that $\mathfrak{l}_1 \in \mathfrak{label}(\Gamma(v_k))$ is marked as sleeping
   $\Longrightarrow$ Rule (**TR 14**) cannot be applied to label $\mathfrak{l}_1$ in node v_{n-1}
   $\Longrightarrow$ contradiction to the assumption, that $\mathfrak{l}_1 \xrightarrow{b} \mathfrak{l}_2' \in \Gamma_n \setminus \Gamma_{n-1}$.

2. Suppose, $\mathfrak{l}_1 \xrightarrow{a} \mathfrak{l}_2 \in \Gamma_\pi$ and $\mathfrak{l}_1' \xrightarrow{b} \mathfrak{l}_2 \in \Gamma_\pi$ with $\mathfrak{l}_2 \neq \mathfrak{l}_2'$ or $a \neq b$.
   $\Longrightarrow$ there must exist a least prefix $v_0 \ldots v_m \ldots v_n$ of path π such that

a) the set $\Gamma(\nu_m)$ is derived from $\Gamma(\nu_{m-1})$ by the application of rule (**TR 14**) and w.l.o.g. $\mathfrak{l}_1 \xrightarrow{a} \mathfrak{l}_2 \in \Gamma_m \setminus \Gamma_{m-1}$ and

b) $\Gamma(\nu_n)$ is derived from $\Gamma(\nu_{n-1})$ by the application of rule (**TR 14**) and $\mathfrak{l}_1' \xrightarrow{b} \mathfrak{l}_2 \in \Gamma_n \setminus \Gamma_{n-1}$

$\Longrightarrow$ (* no rule deletes an existing label from a tableau set[3] *)
$\mathfrak{l}_2 \in \mathfrak{label}(\Gamma(\nu_k))$ for all $k \geq m$
$\Longrightarrow$ (* because rule (**TR 14**) adds a *new* label *)
rule (**TR 14**) cannot add the tableau formula $\mathfrak{l}_1' \xrightarrow{b} \mathfrak{l}_2$ to the tableau set in node $\Gamma(\nu_n)$, since $\mathfrak{l}_2$ is not new
$\Longrightarrow$ this is a contradiction to the assumption that $\mathfrak{l}_1' \xrightarrow{b} \mathfrak{l}_2 \in \Gamma_n \setminus \Gamma_{n-1}$.

3. Suppose, $\chi = \mathfrak{l} \xrightarrow{a_0} \mathfrak{l}_1 \xrightarrow{a_1} \mathfrak{l}_2 \xrightarrow{a_2} \dots \xrightarrow{a_{n-1}} \mathfrak{l}_n \xrightarrow{a_n} \mathfrak{l}$ is a chain of Γ_π.
$\Longrightarrow$ there exists a least prefix $\nu_0\nu_1, \dots \nu_m$ of π such that χ is a chain of Γ_m.
$\Longrightarrow$ rule (**TR 14**) is applied to node ν_{m-1} and w.l.o.g. $\mathfrak{l} \xrightarrow{a_0} \mathfrak{l}_1 \xrightarrow{a_1} \mathfrak{l}_2 \xrightarrow{a_2} \dots \xrightarrow{a_{n-1}} \mathfrak{l}_n$ is a chain of Γ_{m-1} and $\mathfrak{l}_n \xrightarrow{a_n} \mathfrak{l} \in \Gamma_m \setminus \Gamma_{m-1}$
$\Longrightarrow$ $\mathfrak{l} \in \mathfrak{label}(\Gamma(\nu_{m-1}))$
$\Longrightarrow$ $\mathfrak{l}$ is not new to $\Gamma(\nu_m)$ which is a contradiction to the application of rule (**TR 14**).

□

From lemma 5.5.4 we can easily conclude the following corollaries:

Corollary 5.5.5 *Let Γ be a tableau set that has the regular tableau property. Every subset $\Gamma' \subseteq \Gamma$ has the regular tableau property.*

Corollary 5.5.6 *Let Γ be a tableau set that has the regular tableau property and let $\mathfrak{l} \xrightarrow{a} \mathfrak{l}' \in \Gamma$. Then there does not exist a chain χ in Γ with $\chi = \mathfrak{l} \xrightarrow{a_0} \cdots \xrightarrow{a_k} \cdots \xrightarrow{a_n} \mathfrak{l}'$ and $a_k \neq a$.*

Our aim of constructing a tableau for a tableau set $\Gamma_{\mathcal{T}}$ is to prove $\mathcal{L}$-satisfiability or $\mathcal{L}$-unsatisfiability of $\Gamma_{\mathcal{T}}$. What rejection conditions are needed for tableaux so that all $\mathcal{L}$-unsatisfiable tableau sets are rejected and all $\mathcal{L}$-satisfiable tableau sets are accepted?

[3]except for rule (**TR 17**), but after the application of this rule only axiom (**TR 2**) can be applied to close the path, rule (**TR 14**) cannot be applied again on that path. Rule (**TR 3**) cannot delete a tableau label as for the activator $\mathfrak{l} \vdash \top$ it holds that either $\mathfrak{l} = \mathfrak{l}_o$ or there exists a structure formula in the numerator containing $\mathfrak{l}$. In either case the tableau label is not deleted from the set by application of rule (**TR 3**).

If we only had to deal with finite tableaux, we could use the standard definition for acceptance or rejection of tableaux: A leaf, that carries an inconsistent set of formulae cannot be satisfied and thus must be rejected. A leaf, that carries a consistent set will be accepted. If all leaves are rejected, the tableau is rejected, if at least one leaf is accepted, the tableau is accepted. However, the tableau system for $\mathfrak{L}$ may construct infinite paths due to several reasons. Some of these paths may be accepted, because they represent a possible run of a model, others may have to be rejected.

There are several reasons for possibly infinite tableau paths. The following list gives some examples:

- The repeated application of the induction rules (**TR 7**) and (**TR 8**) may lead to infinite paths in a tableau.

  Consider for example a tableau set $\Gamma_0^{\mathcal{T}} = \{\mathfrak{l} \vdash \top\ \mathcal{U}_i \bot, \mathfrak{l}\}$.

  Repeated applications of rules (**TR 7**), (**TR 4**), (**TR 3**), (**TR 14**), (**TR 20**) lead to an infinite tableau. A sketch of a tableau proof for $\Gamma_0^{\mathcal{T}}$ is given in the appendix on page 239.

  Semantically speaking, the infinite path represents an infinite interleaving of an infinite run in which obviously the formula $\top$ is always satisfied and the formula $\bot$ is never satisfied. However, such a run does not satisfy the "strong until" $\top \mathcal{U}_i \bot$. Generally, in a formula $\phi\ \mathcal{U}_i \psi$ the sub formula ψ must eventually become true to fulfill the whole formula. This is not necessary in case of formula $\phi \mathcal{W}_i \psi$. The semantics of formula $\phi \mathcal{W}_i \psi$ is "either $\phi \mathcal{U}_i \psi$ or $\Box_i \phi$", so $\top\ \mathcal{W}_i \bot$ would be satisfied on the infinite path described above.

  Thus, in the tableau we have to distinguish between infinite paths that have to be rejected (those, that correspond to a satisfying interleaving) and those, that need to be accepted.

- The combination of knowledge formulae and "next" formulae may lead to an infinite R_i equivalence class, which in turn leads to an infinite tableau path: Consider for example the tableau set $\Gamma_0^{\mathcal{T}} = \{\mathfrak{l} \vdash \mathsf{K}_i \langle a \rangle_j \top, \mathfrak{l}\}$ where $i \notin ag(a)$. Repeated application of rules (**TR 6**), (**TR 14**), (**TR 15**), (**TR 24**), and (**TR 12**) would generate an infinite tableau path. (A more detailed proof sketch is given in the appendix on page 240.) In terms of the tableau this means that if we have a tableau formula $(\mathfrak{F}, \mathfrak{c}) \vdash \mathsf{K}_i \langle a \rangle_j \phi \in \Gamma_\pi$ (for π being an infinite path) then by application of rules (**TR 6**) and (**TR 14**) we will also have $(\mathfrak{F}, \mathfrak{c}) \vdash \langle a \rangle_i \phi \in \Gamma_\pi$ and $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a) \in \Gamma_\pi$ for some tableau label $(\mathfrak{F}, \mathfrak{c}a)$, and finally also $(\mathfrak{F}, \mathfrak{c}a) \vdash \mathsf{K}_i \langle a \rangle_j \phi \in \Gamma_\pi$. Here, we already see, that if none of the other formulae in the set Γ_π leads to a contradiction, we will have infinitely many applications of the rules (**TR 6**) and (**TR 14**) on a path and consequently we create an infinite tableau path.

On the semantical level this means that in some situation (F, c) an agent i *knows* that another agent j is performing some operation a next. Here, it is important that agent i does not take part in operation a and thus the situations before and after performing a (the situations (F, c) and (F, ca)) are equivalent from agent i's point of view. Since agent i cannot distinguish between (F, c) and (F, ca) both situations are R_i^K-indistinguishable. This again implies that if $(F, c) \models_{\mathcal{M}} K_i\langle a\rangle_j\phi$ we also have $(F, ca) \models_{\mathcal{M}} K_i\langle a\rangle_j\phi$. It is easy to see, that this formula requires an infinite repetition of operation a by agent j and consequently an infinite model.

- The third possibility is due to a particular fairness problem. Each branch of a tableau tree is (semantically) related to a particular interleaving of a run. Now consider the case where the set of formulae contains a formula of type $(\mathfrak{F}, \mathfrak{c}) \vdash \langle a\rangle_i\phi$. If on this branch operations of agent i are ignored for an infinite number of steps, which means that no further a'-step with $a' \in \mathcal{O}_i$ is deduced on the rest of this (infinite) branch. On the semantical level this is equivalent to saying that in some situation (F, c) agent i does not perform any more actions though there is another action required by the formula $\langle a\rangle_i\phi$. This of course means, that the formula $\langle a\rangle_i\phi$ is not satisfied in the situation (F, c). Consequently, an unfair tableau path must get rejected. The only point is that we have to make sure, that there exists a path in the tableau, which represents a fair interleaving. That this is the case can easily be seen: Each time, rule (**TR 14**) is applied, *all* possible operations are fanned out, which implies, that each possible interleaving is represented in the tableau.

Some of the infinite paths need to be rejected, others need to be accepted: We reject paths, that are infinite because they represent an unfair interleaving or because of an infinite occurrence of a formula $\phi\ \mathcal{U}_i\psi$ and we accept all other infinite paths.

Before we give a formal definition for the rejection / acceptance of a tableau, we first define more formally the two criteria due to which infinite paths must be rejected.

Definition 5.5.7 ($\mathcal{U}$-trace) *Let $\mathcal{T}$ be a tableau and let $\pi = \nu_0\nu_1\nu_2\ldots$ be an infinite path of $\mathcal{T}$. The path π contains an until-trace ($\mathcal{U}$-trace), iff Γ_π contains a (not necessarily maximal) infinite chain $\chi = \mathfrak{l}_0 \xrightarrow{a_0} \mathfrak{l}_1 \xrightarrow{a_1} \ldots$ such that the following holds:*

- $\mathfrak{l}_0 \vdash \phi\ \mathcal{U}_i\psi \in \Gamma_\pi$
- *for all $\mathfrak{l}_m$ in χ with $\mathfrak{l}_m \vdash \phi\ \mathcal{U}_i\psi \in \Gamma_\pi$ there exists a subchain $\chi' = \mathfrak{l}_m \xrightarrow{a_m} \ldots \xrightarrow{a_{n-1}} \mathfrak{l}_n$ of χ with $m < n$, such that*

– $\mathfrak{l}_m \vdash \neg\psi \in \Gamma_\pi$
– *for all* a_k*, with* $m \leq k < n-1$ *we have that* $a_k \notin \mathcal{O}_i$*, and*
– $\mathfrak{l}_n \vdash \phi\mathcal{U}_i\psi \in \Gamma_\pi$

Definition 5.5.8 (fair tableau paths) *Let* $\mathcal{T}$ *be a tableau and let* $\pi = \nu_0, \nu_1, \nu_2, \ldots$ *be a path in* $\mathcal{T}$*.*

We call π **fair** *iff*
for each tableau formula $\mathfrak{l} \vdash \langle\cdot\rangle_i\phi \in \Gamma_\pi$ *there exists a chain* $\chi = \mathfrak{l} \xrightarrow{a_0} \mathfrak{l}_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} \mathfrak{l}_n$ *in* Γ_π *with* $a_{n-1} \in \mathcal{O}_i$*.*

A path, which is not fair, is called **unfair***.*

Definition 5.5.9 (acceptance/rejection condition) *Given a tableau* $\mathcal{T}$ *for a set of tableau formulae* $\Gamma_\mathcal{T} \subset \mathfrak{L}$*. A* **path in a tableau is rejected***, iff*

1. *it is finite and its leaf is labelled with an axiom,*
2. *it contains an infinite* $\mathcal{U}$*-trace or*
3. *it is unfair.*

A **path in a tableau is accepted***, iff it is not rejected.*

A **tableau is rejected***, iff all its paths are rejected.*
A **tableau is accepted***, iff it is not rejected.*

We give some example of tableau proofs in the appendix C.

5.6 Basic Properties of Regular Tableaux

5.6.1 Fairness on Application of Tableau Rules

The application of tableau rules is more or less non deterministic. For the correct functioning of the tableau method we need to ensure that every rule that can be applied will be served after a finite number of steps. Otherwise, by ignoring an applicable rule infinitely often on a tableau path, the tableau procedure could create an infinite tableau path that is wrongly accepted. Consider for example the tableau set $\{\mathfrak{l} \vdash K_i\langle a\rangle_j\bot, \mathfrak{l}\}$ with $i, j \in Ag, a \in \mathcal{O}, i \notin ag(a)$. Without dividing the tableau rules into blocks and applying them block wise, we could derive the following tableau path and always (i.e. for infinitely many steps) ignore the application of rule (**TR 15**), the tableau would accept the formula to be $\mathcal{L}-$satisfiable, though it is obviously not:

$$
\begin{array}{c}
\Gamma(\nu_0) = \{\, \mathfrak{l} \vdash K_i\langle a\rangle_j\bot \,, \mathfrak{l}\} \\
\Downarrow \text{(* rule (\textbf{TR 6}) *)} \\
\Gamma(\nu_1) = \{\mathfrak{l} \vdash K_i\langle a\rangle_j\bot,\ \mathfrak{l} \vdash \langle a\rangle_j\bot \,, \mathfrak{l}\} \\
\Downarrow \text{(* rule (\textbf{TR 14}) *)} \\
\Gamma(\nu_2) = \{\mathfrak{l} \vdash K_i\langle a\rangle_j\bot, \mathfrak{l} \vdash \langle a\rangle_j\bot,\ \mathfrak{l} \xrightarrow{a} \mathfrak{l}a \,, \mathfrak{l}, \mathfrak{l}a\} \\
\Downarrow \text{(* rule (\textbf{TR 24}) *)} \\
\Gamma(\nu_3) = \{\, \mathfrak{l} \vdash K_i\langle a\rangle_j\bot \,, \mathfrak{l} \vdash \langle a\rangle_j\bot, \mathfrak{l} \xrightarrow{a} \mathfrak{l}a,\ \mathfrak{l}R_i\mathfrak{l}a \,, \mathfrak{l}, \mathfrak{l}a\} \\
\Downarrow \text{(* rule (\textbf{TR 12}) *)} \\
\Gamma(\nu_4) = \{\mathfrak{l} \vdash K_i\langle a\rangle_j\bot, \mathfrak{l} \vdash \langle a\rangle_j\bot, \mathfrak{l} \xrightarrow{a} \mathfrak{l}a, \mathfrak{l}R_i\mathfrak{l}a,\ \mathfrak{l}a \vdash K_i\langle a\rangle_j\bot \,, \mathfrak{l}, \mathfrak{l}a\} \\
\Downarrow \text{(* rule (\textbf{TR 6}) *)} \\
\vdots
\end{array}
$$

Note the difference between the notion of fairness in definition 5.5.8 and the type of inequitability we consider in this example: In this example, tableau rule (**TR 15**) is applicable to infinitely many tableau sets on the path but it is never actually applied. So, in this case we consider an inequitable sequence of tableau rule applications.

In definition 5.5.8 we talk about unfair paths π in a different sense: The existence of a formula $\mathfrak{l} \vdash \langle\cdot\rangle_i\phi$ requires a tableau formula $\mathfrak{l} \xrightarrow{o} \mathfrak{l}'$ with $o \in \mathcal{O}_i$. However, each time rule (**TR 14**) is applied on π, we choose a denominator that considers a next operation $\wr \notin \mathcal{O}_i$, so that we have an unfair sequence of choices of denominators. Such a path represents an unfair interleaving of a potential model.

To prevent the generation of such inequitable sequences of applications of rules mentioned in the above example, we have divided the tableau rules into three blocks, the exertion of which alters: Rules of each block are applied as long as possible and when no rule of that block can be applied any more, we apply rules of the next block as long as possible and so forth. For our example, this means that before we could apply rule (**TR 24**) from block 2 we had to apply all applicable rules of block 1, in particular rule (**TR 15**) which we ignored in our example. We show that even on infinite tableau paths this block order ensures a certain kind of "fairness" in the application of the tableau rules: Since we deal with finite sets of tableau rules, rules of each block can be applied only a finite number of times before rules of the "next" block must be applied. It is the combination of the blocks that creates infinite paths. This guarantees that if a rule is applicable then it will either be applied within a finite number of steps or it will not be applied because the modification achievable through its application has already been achieved through the application of other rules. Consider for example a tableau set $\Gamma_{\mathcal{T}} = \Gamma'_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash K_i\phi\}$. Obviously, for $\Gamma_{\mathcal{T}}$ rule (**TR 6**) can be applied. There is no rule applicable to $\Gamma_{\mathcal{T}}$ that "disables" rule (**TR 6**). However, it is still possible that rule (**TR 6**) will never be applied: the resulting formula $(\mathfrak{F}, \mathfrak{c}) \vdash \phi$ might already be contained in $\Gamma'_{\mathcal{T}}$ and the definition of tableaux requires for the application of rules that all denominators differ from the numerator. We therefore distinguish between enabled rules (those that are applicable and have to be applied after a finite number of steps) and satisfied rules that are applicable, but the result of which is already contained in the set.

Definition 5.6.1 (enabled rules, satisfied rules) *Let $\Gamma_{\mathcal{T}}$ be a tableau set and let $\frac{\Gamma_{num}}{\Gamma_{d_1} \mid \dots \mid \Gamma_{d_n}}$ be a tableau rule.*

We call a rule $\frac{\Gamma_{num}}{\Gamma_{d_1} \mid \dots \mid \Gamma_{d_n}}$ **enabled** *for $\Gamma_{\mathcal{T}}$ iff $\Gamma_{\mathcal{T}}$ complies to Γ_{num} such that the rule is applicable to $\Gamma_{\mathcal{T}}$.*

We call $\frac{\Gamma_{num}}{\Gamma_{d_1} \mid \dots \mid \Gamma_{d_n}}$ **satisfied** *iff it is enabled but for at least one of the denominators Γ_{d_k}, the application of the rule would not have any effect, i.e. $\Gamma_{\mathrm{num}} = \Gamma_{d_k}$.*

Note, that only rules that are **enabled but not satisfied** for a tableau set $\Gamma_{\mathcal{T}}$ may actually be applied to $\Gamma_{\mathcal{T}}$.

Let us now consider a finite tableau set $\Gamma_{\mathcal{T}}$.

To show that rules of block 1 can be applied only a finite number of times in succession, we define a norm for the size of tableau sets. The considered measure will be a Laurent polynomial. For this norm, we will define a Noetherian ordering. We then show that the application of a rule of block 1 strictly decreases the size of the considered tableau set. Since the ordering is Noetherian, rules of block 1 can be applied only a finite number of times in succession. The size of a tableau set depends on both the structure formulae and the labelled formulae.

For the application of rules of block 2 we again define a norm for tableau sets. This measure is based on some kind of saturation argument:

- Rules (**TR 10**), (**TR 11**), (**TR 24**) and (**TR 26**) fill up the R_i-Relations, i.e. they add new tableau formulae of the type $l \mathrm{R}_i l'$. Since the set of labels in each tableau set occurring at a tableau node is finite and is not modified by rules of block 2 and since no structure formula $l \mathrm{R}_i l'$ is deleted by a rule of block 2, the rules mentioned above can be applied only a finite number of times in succession before a rule of another block is applied.
- Rules (**TR 12**), (**TR 13**) and (**TR 25**) fill up labels with propositions, knowledge formulae and with next-formulae of the type $l \vdash \langle a \rangle_i \top$. Propositions and knowledge formulae are simply migrated to other labels, there is no rules of block 2 that deletes or creates new knowledge formulae or propositions. The next formulae are newly created by rule (**TR 25**). However, there is no rule in block 2 that deletes such a formula again and there are only finitely many agents $i \in Ag$ and finitely many operations $a \in \mathcal{O}$ and thereby only finitely

many temporal next formulae $\langle a \rangle_i \top$. Now the argument is again that there exist only finitely many labels in a tableau set at any tableau node and only finitely many tableau formulae. None of the rules of block 2 creates new tableau labels. Thus, rules of block 2 can be applied only finitely many times in succession before a rule of another block is applied.

Block 3 consists of rules (**TR 9**) and (**TR 14**). When applied to a tableau set $\Gamma(\nu)$ at a tableau node ν, rule (**TR 9**) adds a new label $\mathfrak{l}'$ to the set of tableau labels in $\Gamma(\nu)$, adds a formula $\mathfrak{l}R_i\mathfrak{l}'$ and marks the activator of the rule, so that it cannot activate this rule again. The effect of rule (**TR 14**) are similar: When applied to a tableau set $\Gamma(\nu)$ at a tableau node ν, rule (**TR 14**) creates a number of children. For each child, it adds a new label $\mathfrak{l}\mathfrak{a}$ to the set of tableau labels in $\Gamma(\nu)$, adds a new formula $\mathfrak{l} \xrightarrow{a} \mathfrak{l}\mathfrak{a}$ and marks the activating label $\mathfrak{l}$. Rule (**TR 14**) has a tableau formula $\mathfrak{l} \vdash \langle \cdot \rangle_i \phi$ and the appropriate unmarked label $\mathfrak{l}$ as activator. The application of the rule marks the activating label $\mathfrak{l}$. Both rules of block 3 mark their activators and do not unmark marked activators again. Further, because formulae have only a finite number of sub formulae, these rules can create only finitely many new complete activators for either rule (**TR 9**) or (**TR 14**). So, rules of block 3 can be applied only a finite number of times in succession before a rule of block 1 or block 2 must be applied again.

From the above considerations we can conclude that rules of each block can be applied only a finite number of times in succession. Only the iteration of blocks yields infinite tableau paths, as shown in figure 5.3.

Let us now capture this more formally.

Lemma 5.6.2 (finite application of each block) *Let $\Gamma_{\mathcal{T}}$ be a finite tableau set having the regular tableau property. For $k \in \{1, 2, 3\}$ it holds that starting from $\Gamma_{\mathcal{T}}$ rules of exclusively block k can be applied only finitely many times in succession.*

We prove this lemma for each block separately. By defining a norm $\| \cdot \|^{b1}$ for finite tableau sets that have the regular tableau property and a Noetherian order relation for this norm we prove that rules of block 1 can be applied only finitely many times in succession.

Then we do the same for rules of block 2: We define a norm $\| \cdot \|^{b2}$ for finite tableau sets having the regular tableau property and a Noetherian order relation and show that the application of a rule of block 2 decreases the size of the considered tableau set. Finally, we repeat this procedure for rules of block 3.

Proposition 5.6.3 *Let $\Gamma_{\mathcal{T}}$ be a finite tableau set having the regular tableau property. Starting with $\Gamma_{\mathcal{T}}$, rules of block 1 may be applied only a finite number of times in succession.*

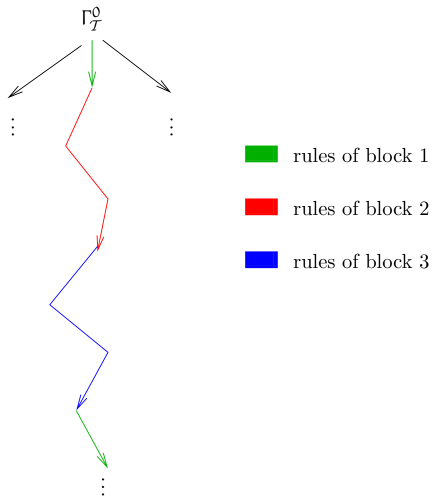


Figure 5.3: Rules of each block can be applied only a finite number of times in succession

Let us first define a norm $|\cdot|^{b1}$ for logical formulae that may occur in a tableau and a norm $\|\cdot\|^{b1}$ for finite tableau sets having the regular tableau property. The norms will be Laurent polynomials[4]. We then define a Noetherian ordering on the set of Laurent polynomials. Finally, we show that each application of a rule of block 1 strictly decreases (according to the defined Noetherian ordering) the size of the tableau set to which the rule was applied. Since the ordering is Noetherian, i.e. there cannot be infinite descending chains, there can be only finitely many applications of rules of block 1.

As constituted in section 5.2, negation symbols occur in tableau sets only in front of the knowledge operators K_i and in front of local propositions $p \in \mathcal{P}$. In addition, knowledge formulae of type $K_i\phi$ can occur in a tableau marked as well as unmarked.

We first define a norm $|\cdot|^{b1}$ for logical formulae that may occur in a tableau set (preceded by a tableau label). We define the norm in such a way, that if a formula ϕ occurs in the principal formula of a rule of block 1, then the size of the logical formula occurring in the side formula of the rule is less than (or equal to) the size of ϕ.

Definition 5.6.4 (norm $|\cdot|^{b1}$ for a logical formula) *Let $(Ag, \mathcal{O}, \mathcal{P})$ be the static part of an information system. Let $i \in Ag$ be an agent and let $a \in \mathcal{O}$ be an operation.*

[4] A Laurent polynomial is like a polynomial except that it may have positive as well as negative powers.

Let $\mathcal{L}^* \subset \mathcal{L}$ be the set of those logical formulae that may occur in a tableau set having the regular tableau property (i.e. the negation symbol $\neg$ may occur only in front of a local proposition or in front of a knowledge operator K_i, formulae of the form $\mathsf{K}_i\psi$ may be either marked or unmarked).

Additionally, let x be a positive integer variable, $x > 0$. We define the norm $|\cdot|^{b1}$ for formulae $\phi \in \mathcal{L}^*$ *inductively as a Laurent polynomial in x and x^{-1} with coefficients in $\mathbb{N}$:*

$$|\cdot|^{b1} : \mathcal{L}^* \longrightarrow \mathbb{N}[x, x^{-1}]$$

$\lvert\top\rvert^{b1} = \lvert\bot\rvert^{b1}$	$= 1$	
$\lvert p\rvert^{b1} = \lvert\neg p\rvert^{b1}$	$= 1$	
$\lvert\phi \wedge \psi\rvert^{b1}$	$= \lvert\phi\rvert^{b1} + 1 + \lvert\psi\rvert^{b1}$	
$\lvert\phi \vee \psi\rvert^{b1}$	$= \lvert\phi\rvert^{b1} + 1 + \lvert\psi\rvert^{b1}$	
$\lvert\phi\ \mathcal{U}_i\psi\rvert^{b1}$	$= \lvert\phi\rvert^{b1} + 3 + \lvert\neg\psi\rvert^{b1}$	
$\lvert\phi\ \mathcal{W}_i\psi\rvert^{b1}$	$= \lvert\phi\rvert^{b1} + 3 + \lvert\neg\psi\rvert^{b1}$	
$\lvert\mathsf{K}_i\phi\rvert^{b1}$	$= \lvert\phi\rvert^{b1} + 1$	*for $\mathsf{K}_i\phi$ unmarked*
$\lvert\mathsf{K}_i\phi\rvert^{b1}$	$= 0$	*for $\mathsf{K}_i\phi$ marked*
$\lvert\neg\mathsf{K}_i\phi\rvert^{b1}$	$= \lvert\phi\rvert^{b1} + 1$	
$\lvert\langle a\rangle_i\phi\rvert^{b1}$	$= \frac{\lvert\phi\rvert^{b1}}{x} + 1$	
$\lvert[a]_i\phi\rvert^{b1}$	$= \frac{\lvert\phi\rvert^{b1}}{x} + 1$	
$\lvert\langle\mathcal{O}\rangle_i\phi\rvert^{b1}$	$= \frac{\lvert\phi\rvert^{b1}}{x} + 1$	
$\lvert[\mathcal{O}]_i\phi\rvert^{b1}$	$= \frac{\lvert\phi\rvert^{b1}}{x} + 1$	

Each finite tableau set that has the regular tableau property $\Gamma_{\mathcal{T}} \subset \mathfrak{L}$ is partitioned into a set of structure formulae $\Gamma_s \subset \mathfrak{L}_{\mathfrak{s}}$, a set of labelled formulae $\Gamma_l \subset \mathfrak{L}_{\mathfrak{l}}$ and a set of tableau labels $\Gamma_{\mathfrak{label}}$ with $\Gamma_{\mathfrak{label}} = \mathfrak{label}(\Gamma_{\mathcal{T}})$. Looking at the structure formulae we can identify a set of maximal chains of tableau labels. Since in each tableau node, the set of tableau formulae is finite, the number of (maximal) chains as well as each maximal chain in this set is finite.

A tableau formula $\mathfrak{l} \vdash \phi$ of $\mathfrak{L}_{\mathfrak{l}}$ consists of two components: A tableau label $\mathfrak{l}$ and a logical formula ϕ of $\mathcal{L}^*$. The size of such a tableau formula is dependent on both components and is defined relative to a finite tableau set with $\mathfrak{l} \in \mathfrak{label}(\Gamma_{\mathcal{T}})$. Each set of tableau formulae occurring in a regular tableau contains a finite set of finite chains, where each label occurs in exactly one chain (see lemma 5.5.3, regular tableau property). The "later" a label $\mathfrak{l}_k$ occurs in the chain, the "less" the formula preceded by this label counts.

Based on the norm $|\cdot|^{b1}$ for logical formulae in $\mathcal{L}^*$ we now define a norm $\|\cdot\|^{b1}$ for tableau formulae in $\mathfrak{L}_{\mathfrak{l}}$. Again, the norm is defined as a Laurent polynomial in x and x^{-1} and with coefficients in $\mathbb{N}$. The size of a tableau formula $\mathfrak{l} \vdash \phi$ is dependent on the position in the maximal chain at which the tableau label $\mathfrak{l}$ occurs. Hence, we define the size of a tableau formula $\mathfrak{l} \vdash \phi$ relative to the tableau set in which the tableau formula occurs.

Definition 5.6.5 (norm $\|\cdot\|^{b1}$ for (sets of) tableau formula) *Let $\Gamma_{\mathcal{T}}, \Gamma'_{\mathcal{T}}$ be two finite tableau sets that have the regular tableau property and that are chain equivalent ($\Gamma_{\mathcal{T}} \simeq \Gamma'_{\mathcal{T}}$). And let $\chi = \mathfrak{l}_0 \xrightarrow{a_0} \ldots \xrightarrow{a_{k-1}} \mathfrak{l}_k \xrightarrow{a_k} \ldots \xrightarrow{a_{n-2}} \mathfrak{l}_{n-1}$ be a maximal chain of $\Gamma_{\mathcal{T}}$ (where n is the number of tableau labels contained in the chain) such that the tableau label $\mathfrak{l}_k$ occurs on the $(k+1)$th position in the chain. Further, let x be a positive integer variable, $x > 0$.*

We define the size of a tableau formula $\mathfrak{l}_k \vdash \phi$ *relative to the set of tableau formulae $\Gamma_{\mathcal{T}}$ as a Laurent polynomial in x and x^{-1} and coefficients in $\mathbb{N}$:*

$$\|\mathfrak{l}_k \vdash \phi\|^{b1}_{\Gamma_{\mathcal{T}}} := |\phi|^{b1} \cdot x^{n-k} \tag{2}$$

We define the size of the set of tableau formulae $\Gamma'_{\mathcal{T}}$ *relative to $\Gamma_{\mathcal{T}}$ as the sum of all sizes relative to $\Gamma_{\mathcal{T}}$ of labelled formulae in $\Gamma'_{\mathcal{T}}$:*

$$\|\Gamma'_{\mathcal{T}}\|^{b1}_{\Gamma_{\mathcal{T}}} := \sum_{\gamma \in \Gamma'_{\mathcal{T}} \cap \mathfrak{L}_{\mathfrak{l}}} \|\gamma\|^{b1}_{\Gamma_{\mathcal{T}}} \tag{3}$$

For $\Gamma_{\mathcal{T}} = \Gamma'_{\mathcal{T}}$ we also write $\|\Gamma_{\mathcal{T}}\|^{b1}$ instead of $\|\Gamma_{\mathcal{T}}\|^{b1}_{\Gamma'_{\mathcal{T}}}$.

Example:
Let $\Gamma_{\mathcal{T}} = \{\mathfrak{l} \vdash \langle a\rangle_i\langle b\rangle_i\langle c\rangle_i P \wedge \langle b\rangle_i\langle d\rangle_i P, \mathfrak{l} \xrightarrow{a} \mathfrak{l}a, \mathfrak{l}, \mathfrak{l}a\}$ be a tableau set. The norm $\|\Gamma_{\mathcal{T}}\|^{b1}$ is a Laurent polynomial:

The only maximal chain of $\Gamma_{\mathcal{T}}$ is $\mathfrak{l} \xrightarrow{a} \mathfrak{l}a$. As a consequence we have $n = 2$. It is obvious that $\Gamma_{\mathcal{T}}$ has the regular tableau property.

$$
\begin{aligned}
\|\Gamma_{\mathcal{T}}\|^{b1}_{\Gamma_{\mathcal{T}}} =& \|\{l \vdash \langle a\rangle_i\langle b\rangle_i\langle c\rangle_i P \wedge \langle b\rangle_i\langle d\rangle_i P\}\|^{b1}_{\Gamma_{\mathcal{T}}} \\
=& |\langle a\rangle_i\langle b\rangle_i\langle c\rangle_i P \wedge \langle b\rangle_i\langle d\rangle_i P|^{b1} \cdot x^2 \\
=& \left(|\langle a\rangle_i\langle b\rangle_i\langle c\rangle_i P|^{b1} + 1 + |\langle b\rangle_i\langle d\rangle_i P|^{b1}\right) \cdot x^2 \\
=& |\langle a\rangle_i\langle b\rangle_i\langle c\rangle_i P|^{b1} \cdot x^2 + 1 \cdot x^2 + |\langle b\rangle_i\langle d\rangle_i P|^{b1} \cdot x^2 \\
=& \left(\left(\frac{|\langle b\rangle_i\langle c\rangle_i P|^{b1}}{x}\right) + 1\right) \cdot x^2 + 1 \cdot x^2 + \left(\frac{|\langle d\rangle_i P|^{b1}}{x} + 1\right) \cdot x^2 \\
=& \left(\left(\frac{\frac{|\langle c\rangle_i P|^{b1}}{x} + 1}{x}\right) + 1\right) \cdot x^2 + 1 \cdot x^2 + \left(\frac{\frac{|P|^{b1}}{x} + 1}{x} + 1\right) \cdot x^2 \\
=& \left(\left(\frac{\frac{\frac{|P|^{b1}}{x} + 1}{x} + 1}{x}\right) + 1\right) \cdot x^2 + 1 \cdot x^2 + \left(\frac{\frac{1}{x} + 1}{x} + 1\right) \cdot x^2 \\
=& \left(\left(\frac{\frac{\frac{1}{x} + 1}{x} + 1}{x}\right) + 1\right) \cdot x^2 + 1 \cdot x^2 + 1 + x + x^2 \\
=& \left(\frac{1}{x} + 1 + x + x^2\right) + (1 + x + x^2) \\
=& 3x^2 + 2x + 2 + \frac{1}{x} \\
=& 3x^2 + 2x + 2 + 1x^{-1}
\end{aligned}
$$

By means of a Noetherian ordering on Laurent polynomials with coefficients in $\mathbb{N}$ we show that each application of a rule of block 1 decreases the size of the considered tableau set. As the ordering will be Noetherian, there can be only finite sequences of tableau sets where each set $\Gamma(\nu_{i+1})$ is derived from $\Gamma(\nu_i)$ by the application of a rule of block 1.

A partial ordering $(>, A)$ for an arbitrary set A is a Noetherian ordering if every descending chain $a_1 > a_2 > a_3 > \ldots$ has a least element.

Note, that for each tableau set $\Gamma_{\mathcal{T}}$ that contains at least one labelled formula of type $l \vdash \phi$, the size $\|\Gamma_{\mathcal{T}}\|^{b1}$, given by a Laurent polynomial $\sum_{i \in \mathbb{Z}} a_i x^i$, has a coefficient $a_i \neq 0$ for some $i \in \mathbb{N}$, i.e. it has a term with a positive power, the coefficient of which is in $\mathbb{N}^+$.

The *lexicographical ordering* defined on polynomials with coefficients in $\mathbb{N}$ is a Noetherian ordering. Here, we have to deal with Laurent polynomials with coefficients in $\mathbb{N}$. We now define a Noetherian ordering on Laurent polynomials that is comparable to the lexicographical ordering on polynomials: We consider only the polynomial part of the Laurent polynomial, i.e. we consider only the terms with positive powers and define that a Laurent polynomial $A(x) = \sum_{-m \leq i \leq n} a_i x^i$ is greater ($\succ$) than a Laurent polynomial $B(x) = \sum_{-m \leq i \leq n} b_i x^i$ if the polynomial part $\sum_{0 \leq i \leq n} a_i x^i$ of the Laurent polynomial $A(x)$ is greater (according to the lexicographical ordering) than the polynomial part $\sum_{0 \leq i \leq n} b_i x^i$ of the Laurent polynomial $B(x)$.

Note, that according to this definition, two polynomials that differ only in terms with negative powers are not comparable.

Definition 5.6.6 (Noetherian ordering for Laurent polynomials) *Let* $\sum_{i \in \mathbb{Z}} a_i x^i$ *and* $\sum_{i \in \mathbb{Z}} b_i x^i$ *with* $a_i, b_i \in \mathbb{N}$ *and* $a_i, b_i \neq 0$ *for only finitely many* $i \in \mathbb{Z}$ *be Laurent polynomials.*

Then

$$\sum_{i \in \mathbb{Z}} a_i x^i \succ \sum_{i \in \mathbb{Z}} b_i x^i$$

:iff there exists a $h \in \mathbb{N}$ *such that* $a_i = b_i$ *for all* $i > h$ *and* $a_i > b_i$ *for* $i = h$.

Proposition 5.6.7 *The ordering* $\succ$ *defined on Laurent polynomials in* x *and* x^{-1} *with coefficients in* $\mathbb{N}$ *is a Noetherian ordering.*

Proof:

1. $(\succ, \mathbb{N}[x, x^{-1}])$ is a partial ordering:

    a) $\succ$ is irreflexive (not $A(x) \succ A(x)$). This condition holds trivially.

    b) $\succ$ is asymmetric (not $A(x) \succ B(x)$ and $B(x) \succ A(x)$). This condition holds trivially.

    c) $\succ$ is transitive (if $A(x) \succ B(x)$ and $B(x) \succ C(x)$ then $A(x) \succ C(x)$). This condition holds trivially as well.

2. Every descending chain $A_1(x) \succ A_2(x) \succ A_3(x) \succ \dots$ is finite. This follows directly from the facts that the relation $\succ$ is based on the relation $>$ for integers and that all coefficients of the Laurent polynomials and in particular of the terms with positive powers are natural numbers including 0.

$\square$

The norm for tableau sets is defined as Laurent polynomials. The order relation $\succ$ is a Noetherian order relation. Consequently, if we can show that each application of a rule of block 1 reduces the size of the tableau set according to the relation $\succ$, then rules of block 1 can be applied only finitely many times in succession.

Proof of proposition 5.6.3

For better readability we will use $\underline{\Gamma}$ for the numerator Γ_{num} of a rule and $\overline{\Gamma}$ for the considered denominator Γ_{den}.

- **Rule (TR 3):**
Suppose, that $\overline{\Gamma} = \Gamma_{\mathcal{T}}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{l_k \vdash \top\}$ by the application of rule (**TR 3**) and suppose, that the maximal chain in $\underline{\Gamma}$ containing l_k is $\chi = l_0 \xrightarrow{a_0} \dots \xrightarrow{a_k} l_k \xrightarrow{a_{k+1}} \dots \xrightarrow{a_{n-2}} l_{n-1}$.
$$\begin{aligned} \|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|l_k \vdash \top\|^{b1}_{\underline{\Gamma}} \\ &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + \|l_k \vdash \top\|^{b1}_{\overline{\Gamma}} && \text{(* } \underline{\Gamma} \simeq \overline{\Gamma} \text{ *)} \\ &\succ \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} \\ &= \|\overline{\Gamma}\|^{b1} \end{aligned}$$

- **Rule (TR 4):**
Suppose, that $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{l_k \vdash \phi, l_k \vdash \psi\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{l_k \vdash \phi \wedge \psi\}$ by the application of rule (**TR 4**) and suppose, that the maximal chain in $\underline{\Gamma}$ containing l_k is $\chi = l_0 \xrightarrow{a_0} \dots \xrightarrow{a_k} l_k \xrightarrow{a_{k+1}} \dots \xrightarrow{a_{n-2}} l_{n-1}$.
$$\begin{aligned} \|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|l_k \vdash \phi \wedge \psi\|^{b1}_{\underline{\Gamma}} \\ &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi \wedge \psi|^{b1} \cdot x^{n-k} && \text{(* def. 5.6.5 *)} \\ &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k} + |\psi|^{b1} \cdot x^{n-k} + 1 \cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\ &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k} + |\psi|^{b1} \cdot x^{n-k} + 1 \cdot x^{n-k} && \text{(* } \underline{\Gamma} \simeq \overline{\Gamma} \text{ *)} \\ &\succ \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k} + |\psi|^{b1} \cdot x^{n-k} && \text{(* with } h = n - k \\ & && \text{and def. 5.6.6*)} \\ &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + \|l_k \vdash \phi\|^{b1}_{\overline{\Gamma}} + \|l_k \vdash \psi\|^{b1}_{\overline{\Gamma}} && \text{(* def. 5.6.5 *)} \\ &= \|\overline{\Gamma}\|^{b1} \end{aligned}$$

- **Rule (TR 5):**
Suppose, that, without loss of generality, $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota_k \vdash \phi\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota_k \vdash \phi \vee \psi\}$ by the application of rule (**TR 5**) and the maximal chain in $\underline{\Gamma}$ containing ι_k is $\chi = \iota_0 \xrightarrow{a_0} \ldots \xrightarrow{a_k} \iota_k \xrightarrow{a_{k+1}} \ldots \xrightarrow{a_{n-2}} \iota_{n-1}$.

$$\begin{aligned}
\|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|\iota_k \vdash \phi \vee \psi\|^{b1}_{\underline{\Gamma}} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi \vee \psi|^{b1} \cdot x^{n-k} && \text{(* def. 5.6.5 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k} + |\psi|^{b1} \cdot x^{n-k} + 1 \cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k} + |\psi|^{b1} \cdot x^{n-k} + 1 \cdot x^{n-k} && (\text{* } \underline{\Gamma} \simeq \overline{\Gamma}) \\
&\succ \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k} && \text{(* with } h = n-k \\
& && \text{and def. 5.6.6 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + \|\iota_k \vdash \phi\|^{b1}_{\overline{\Gamma}} && \text{(* def. 5.6.5 *)} \\
&= \|\overline{\Gamma}\|^{b1}
\end{aligned}$$

- **Rule (TR 6):**
Suppose, that $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota_k \vdash K_i\phi, \iota_k \vdash \phi\}$ (where $\iota_k \vdash K_i\phi$ is marked) is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota_k \vdash K_i\phi\}$ (where $\iota_k \vdash K_i\phi$ is unmarked) by the application of rule (**TR 6**) and suppose that the maximal chain in $\underline{\Gamma}$ containing ι_k is $\chi = \iota_0 \xrightarrow{a_0} \ldots \xrightarrow{a_k} \iota_k \xrightarrow{a_{k+1}} \ldots \xrightarrow{a_{n-2}} \iota_{n-1}$.

$$\begin{aligned}
\|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|\iota_k \vdash K_i\phi\|^{b1}_{\underline{\Gamma}} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |K_i\phi|^{b1} \cdot x^{n-k} && \text{(* def. 5.6.5 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + (|\phi|^{b1} + 1) \cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + (|\phi|^{b1} + 1) \cdot x^{n-k} && (\text{ * } \underline{\Gamma} \simeq \overline{\Gamma} \text{ *}) \\
&\succ \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + (0 + |\phi|^{b1}) \cdot x^{n-k} && \text{(* with } h = n-k \\
& && \text{and def. 5.6.6 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + \|\iota_k \vdash K_i\phi\|^{b1}_{\overline{\Gamma}} + \|\iota_k \vdash \phi\|^{b1}_{\overline{\Gamma}} && \text{(* def. 5.6.5 *)} \\
&= \|\overline{\Gamma}\|^{b1}
\end{aligned}$$

- **Rule (TR 7):**
We distinguish between two cases:

1. Suppose, that $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota_k \vdash \phi \wedge \neg\psi \wedge \langle\mathcal{O}\rangle_i\phi\ \mathcal{U}_i\psi\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota_k \vdash \phi\ \mathcal{U}_i\psi\}$ by the application of rule (**TR 7**) and suppose

that the maximal chain in $\underline{\Gamma}$ containing ℓ_k is $\chi = \ell_0 \xrightarrow{a_0} \ldots \xrightarrow{a_k} \ell_k \xrightarrow{a_{k+1}} \ldots \xrightarrow{a_{n-2}} \ell_{n-1}$.

$$
\begin{aligned}
\|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|\ell_k \vdash \phi\ \mathcal{U}_i \psi\|^{b1}_{\underline{\Gamma}} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi\ \mathcal{U}_i \psi|^{b1} \cdot x^{n-k} && \text{(* def. 5.6.5 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k} + |\neg\psi|^{b1} \cdot x^{n-k} + 3 \cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\
&\succ \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k} + |\neg\psi|^{b1} \cdot x^{n-k} && \text{(* with } h = n-k \\
&\quad +2 \cdot x^{n-k} + |\phi\ \mathcal{U}_i \psi|^{b1} \cdot x^{n-k-1} && \text{and def. 5.6.6 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k} + |\neg\psi|^{b1} \cdot x^{n-k} + 2 \cdot && \text{(* } \underline{\Gamma} \simeq \overline{\Gamma} \text{ *)} \\
&\quad x^{n-k} + |\phi\ \mathcal{U}_i \psi|^{b1} \cdot x^{n-k-1} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + |\phi \wedge \neg\psi \wedge \langle\mathcal{O}\rangle_i \phi\ \mathcal{U}_i \psi|^{b1} \cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + \|\ell_k \vdash \phi \wedge \neg\psi \wedge \langle\mathcal{O}\rangle_i \phi\ \mathcal{U}_i \psi\|^{b1}_{\overline{\Gamma}} && \text{(* def. 5.6.5 *)} \\
&= \|\overline{\Gamma}\|^{b1}
\end{aligned}
$$

2. Suppose, that $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\ell_k \vdash \psi\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\ell_k \vdash \phi\ \mathcal{U}_i \psi\}$ by the application of rule (**TR 7**) and suppose that the maximal chain in $\underline{\Gamma}$ containing ℓ_k is $\chi = \ell_0 \xrightarrow{a_0} \ldots \xrightarrow{a_{n-2}} \ell_{n-1}$.

$$
\begin{aligned}
\|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|\ell_k \vdash \phi\ \mathcal{U}_i \psi\|^{b1}_{\underline{\Gamma}} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi\ \mathcal{U}_i \neg\psi|^{b1} \cdot x^{n-k} && \text{(* def. 5.6.5 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k} + |\neg\psi|^{b1} \cdot x^{n-k} + 3 \cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\
&\succ \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\psi|^{b1} \cdot x^{n-k} && \text{(* with } h = n-k \\
& && \text{and def. 5.6.6 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + |\psi|^{b1} \cdot x^{n-k} && \text{(* } \underline{\Gamma} \simeq \overline{\Gamma} \text{ *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + \|\ell_k \vdash \psi\|^{b1}_{\overline{\Gamma}} && \text{(* def. 5.6.5 *)} \\
&= \|\overline{\Gamma}\|^{b1}
\end{aligned}
$$

- **Rule (TR 8):**
  This proof is analogous to the proof for rule (**TR 7**).

- **Rule (TR 15):**
  Suppose, that $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\ell_k \xrightarrow{a} \ell_{k+1}, \ell_{k+1} \vdash \phi\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\ell_k \xrightarrow{a} \ell_{k+1}, \ell_k \vdash \langle a\rangle_i \phi\}$ by the application of rule (**TR 15**) and further suppose, that the maximal chain in $\underline{\Gamma}$ containing ℓ_k is $\chi = \ell_0 \xrightarrow{a_0} \ldots \xrightarrow{a_k} \ell_k \xrightarrow{a_{k+1}} \ldots \xrightarrow{a_{n-2}} \ell_{n-1}$.

$$\begin{aligned}
\|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|\iota_k \vdash \langle a\rangle_i\phi\|^{b1}_{\underline{\Gamma}} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\langle a\rangle_i\phi|^{b1}\cdot x^{n-k} && \text{(* def. 5.6.5 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \tfrac{|\phi|^{b1}}{x}\cdot x^{n-k} + 1\cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + 1\cdot x^{n-k} + |\phi|^{b1}\cdot x^{n-k-1} \\
&\succ \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi|^{b1}\cdot x^{n-k-1} && \text{(* with } h = n-k \\
& && \text{and def. 5.6.6 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + \|\iota_{k+1} \vdash \phi\|^{b1}_{\overline{\Gamma}} && \text{(* def. 5.6.5, } \underline{\Gamma} \simeq \overline{\Gamma} \text{ *)} \\
&= \|\overline{\Gamma}\|^{b1}
\end{aligned}$$

- **Rule (TR 16):**
  This proof is analogous for the proof of rule (**TR 15**).

- **Rule (TR 17):**
  Suppose, that $\overline{\Gamma} = \{\iota_k \vdash \bot\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota_k \xrightarrow{a} \iota_{k+1}, \iota_k \vdash \langle b\rangle_i\phi\}$ by the application of rule (**TR 17**) and further suppose, that the maximal chain in $\underline{\Gamma}$ containing ι_k is $\chi = \iota_0 \xrightarrow{a_0} \ldots \xrightarrow{a_k} \iota_k \xrightarrow{a_{k+1}} \ldots \xrightarrow{a_{n-2}} \iota_{n-1}$. Since $\iota_k \xrightarrow{a} \iota_{k+1} \in \underline{\Gamma}$ it holds that $k < n-1$.

$$\begin{aligned}
\|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|\iota_k \vdash \langle b\rangle_i\phi\|^{b1}_{\underline{\Gamma}} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\langle b\rangle_i\phi|^{b1}\cdot x^{n-k} && \text{(* def. 5.6.5 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \tfrac{|\phi|^{b1}}{x}\cdot x^{n-k} + 1\cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + 1\cdot x^{n-k} + |\phi|^{b1}\cdot x^{n-k-1} \\
&\succ 1\cdot x^{1} && \text{(* with } h = n-k \\
& && \text{and def. 5.6.6 *)} \\
&= |\bot|^{b1}\cdot x^{1} && \text{(* with def. 5.6.4 *)} \\
&= \|\iota_k \vdash \bot\|^{b1}_{\{\iota_k \vdash \bot\}} && \text{(* def. 5.6.5 *)} \\
&= \|\overline{\Gamma}\|^{b1}
\end{aligned}$$

- **Rule (TR 18):**
  Suppose, that $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota_k \xrightarrow{a} \iota_{k+1}, \iota_{k+1} \vdash \langle b\rangle_i\phi\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota_k \xrightarrow{a} \iota_{k+1}, \iota_k \vdash \langle b\rangle_i\phi\}$ by the application of rule (**TR 18**) and further suppose that the maximal chain in $\underline{\Gamma}$ containing ι_k is $\chi = \iota_0 \xrightarrow{a_0} \ldots \xrightarrow{a_k} \iota_k \xrightarrow{a_{k+1}} \ldots \xrightarrow{a_{n-2}} \iota_{n-1}$.

$$
\begin{aligned}
\|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|l_k \vdash \langle b\rangle_i\phi\|^{b1}_{\underline{\Gamma}} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\langle b\rangle_i\phi|^{b1} \cdot x^{n-k} && \text{(* def. 5.6.5 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \tfrac{|\phi|^{b1}}{x} \cdot x^{n-k} + 1 \cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + 1 \cdot x^{n-k} + |\phi|^{b1} \cdot x^{n-k-1} \\
&\succ \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \tfrac{|\phi|^{b1}}{x} \cdot x^{n-k-1} + 1 \cdot x^{n-k-1} && \text{(* with } h = n-k \\
& && \text{and def. 5.6.6 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + \|l_{k+1} \vdash \langle b\rangle_i\phi\|^{b1}_{\overline{\Gamma}} && \text{(* def. 5.6.5, } \underline{\Gamma} \simeq \overline{\Gamma} \text{ *)} \\
&= \|\overline{\Gamma}\|^{b1}
\end{aligned}
$$

- **Rule (TR 19):**
  This proof is analogous to the proof for rule (**TR 18**).

- **Rule (TR 20):** Suppose, that $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{l_k \xrightarrow{a} l_{k+1}, l_{k+1} \vdash \phi\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{l_k \xrightarrow{a} l_{k+1}, l_k \vdash \langle\mathcal{O}\rangle_i\phi\}$ by the application of rule (**TR 20**) and further suppose, that the maximal chain in $\underline{\Gamma}$ containing l_k is $\chi = l_0 \xrightarrow{a_0} \dots \xrightarrow{a_k} l_k \xrightarrow{a_{k+1}} \dots \xrightarrow{a_{n-2}} l_{n-1}$.

$$
\begin{aligned}
\|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|l_k \vdash \langle\mathcal{O}\rangle_i\phi\|^{b1}_{\underline{\Gamma}} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\langle\mathcal{O}\rangle_i\phi|^{b1} \cdot x^{n-k} && \text{(* def. 5.6.5 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \tfrac{|\phi|^{b1}}{x} \cdot x^{n-k} + 1 \cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + 1 \cdot x^{n-k} + |\phi|^{b1} \cdot x^{n-k-1} \\
&\succ \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\phi|^{b1} \cdot x^{n-k-1} && \text{(* with } h = n-k \\
& && \text{and def. 5.6.6 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + \|l_{k+1} \vdash \phi\|^{b1}_{\overline{\Gamma}} && \text{(* def. 5.6.5, } \underline{\Gamma} \simeq \overline{\Gamma} \text{ *)} \\
&= \|\overline{\Gamma}\|^{b1}
\end{aligned}
$$

- **Rule (TR 21):**
  This proof is analogous to the proof for rule (**TR 20**).

- **Rule (TR 22):**
  Suppose, that $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{l_k \xrightarrow{a} l_{k+1}, l_{k+1} \vdash \langle\mathcal{O}\rangle_i\phi\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{l_k \xrightarrow{a} l_{k+1}, l_k \vdash \langle\mathcal{O}\rangle_i\phi\}$ and further suppose, that the maximal chain in $\underline{\Gamma}$ containing l_k is $\chi = l_0 \xrightarrow{a_0} \dots \xrightarrow{a_k} l_k \xrightarrow{a_{k+1}} \dots \xrightarrow{a_{n-2}} l_{n-1}$.

$$
\begin{aligned}
\|\underline{\Gamma}\|^{b1} &= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \|\mathfrak{l}_k \vdash \langle\mathcal{O}\rangle_i\phi\|^{b1}_{\underline{\Gamma}} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + |\langle\mathcal{O}\rangle_i\phi|^{b1} \cdot x^{n-k} && \text{(* def. 5.6.5 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \tfrac{|\phi|^{b1}}{x} \cdot x^{n-k} + 1 \cdot x^{n-k} && \text{(* def. 5.6.4 *)} \\
&\succ \|\Gamma_{\mathcal{T}}\|^{b1}_{\underline{\Gamma}} + \tfrac{|\phi|^{b1}}{x} \cdot x^{n-k-1} + 1 \cdot x^{n-k-1} && \text{(* with } h = n-k \\
& && \text{and def. 5.6.6 *)} \\
&= \|\Gamma_{\mathcal{T}}\|^{b1}_{\overline{\Gamma}} + \|\mathfrak{l}_{k+1} \vdash \langle\mathcal{O}\rangle_i\phi\|^{b1}_{\overline{\Gamma}} && \text{(* def. 5.6.5, } \underline{\Gamma} \simeq \overline{\Gamma} \text{ *)} \\
&= \|\overline{\Gamma}\|^{b1}
\end{aligned}
$$

- **Rule (TR 23):**
  This proof is analogous to the proof for rule (**TR 22**).

□

Next we show that rules of block 2 can be applied only finitely often in succession.

Proposition 5.6.8 *Let $\Gamma_{\mathcal{T}}$ be a finite tableau set having the regular tableau property. Rules of block 2 can be applied only a finite number of times to $\Gamma_{\mathcal{T}}$ in succession.*

Similar to proposition 5.6.3 we prove this proposition by defining a norm for the set of tableau formulae and a Noetherian order for this norm and show that each application of a rule of block 2 decreases the size of the tableau set.

Since the tableau set $\Gamma_{\mathcal{T}}$ is finite, we have that

- the number of tableau labels contained in $\Gamma_{\mathcal{T}}$, $\mathfrak{label}(\Gamma_{\mathcal{T}})$, is finite and
- the number of knowledge formulae and propositions that are sub formulae of tableau formulae contained in the tableau set is finite. (We denote the set of sub formulae of $\Gamma_{\mathcal{T}}$ by *sub*($\Gamma_{\mathcal{T}}$).)

Also, the number of agents and operations is finite.

Each of the tableau rules of block 2 adds either a structure formula of type $\mathfrak{l} \xrightarrow{a} \mathfrak{l}'$ or of type $\mathfrak{l}R_i\mathfrak{l}'$ or a labelled formula of type $\mathfrak{l} \vdash P, \mathfrak{l} \vdash \neg P, \mathfrak{l} \vdash K_i\phi, \mathfrak{l} \vdash \neg K_i\phi$ or $\mathfrak{l} \vdash \langle a\rangle_i\top$ as side formula to the considered tableau set. Note, that the rules neither delete any of those structure formulae or labelled formulae, nor do they add a new tableau label.

We define the size of a tableau set $\Gamma_{\mathcal{T}}$ as the sum of the differences between

1. the potential number of R_i-related pairs of tableau labels occurring in $\Gamma_{\mathcal{T}}$ and the actual number of R_i-related pairs of tableau labels occurring in $\Gamma_{\mathcal{T}}$,

2. the possible number of knowledge formulae and propositions and the actual number of knowledge formulae and propositions occurring in $\Gamma_{\mathcal{T}}$ and
3. the possible number of next formulae of type $\langle a \rangle_i \top$ and the actual number of next formulae $\langle a \rangle_i \top$ occurring in $\Gamma_{\mathcal{T}}$.

To calculate the potential number of knowledge formulae and of propositions that can occur in the tableau set we need to calculate all sub formulae of formulae in $\Gamma_{\mathcal{T}}$. Therefor, we now define the set of sub formulae of a formula $\phi \in \mathcal{L}$, with the restriction that a negation symbol may occur in ϕ only in front of a knowledge operator or in front of a proposition.

Definition 5.6.9 (sub formulae sub(ϕ) of a logical formula ϕ) *Let $\phi \in \mathcal{L}$ be a formula in which all negation symbols occur only in front of a proposition or in front of a knowledge operator K_i. We define the set of sub formulae of ϕ, $sub(\phi)$, inductively:*

$\phi \in sub(\phi)$	
If $\top \in sub(\phi)$	*then* $\bot \in sub(\phi)$
If $\neg \mathsf{P} \in sub(\phi)$	*then* $\mathsf{P} \in sub(\phi)$
If $\phi \wedge \psi \in sub(\phi)$	*then* $\phi, \psi \in sub(\phi)$
If $\phi \vee \psi \in sub(\phi)$	*then* $\phi, \psi \in sub(\phi)$
If $\mathsf{K}_i\phi \in sub(\phi)$	*then* $\phi \in sub(\phi)$
If $\neg\mathsf{K}_i\phi \in \phi$	*then* $\mathsf{K}_i\phi \in sub(\phi)$
If $\phi\ \mathcal{U}_i\psi \in sub(\phi)$	*then* $\phi, \psi \in sub(\phi)$
If $\phi\ \mathcal{W}_i\psi \in sub(\phi)$	*then* $\phi, \psi \in sub(\phi)$

Definition 5.6.10 (sub formulae sub($\Gamma_{\mathcal{T}}$) of a tableau set $\Gamma_{\mathcal{T}}$) *Let $\Gamma_{\mathcal{T}}$ be a finite tableau set. We define the set of sub formulae $sub(\Gamma_{\mathcal{T}})$ of $\Gamma_{\mathcal{T}}$ as*

$$sub(\Gamma_{\mathcal{T}}) := \bigcup_{\mathfrak{l} \in \mathfrak{label}(\Gamma_{\mathcal{T}})} \bigcup_{\mathfrak{l} \vdash \gamma \in \Gamma_{\mathcal{T}}} sub(\gamma)$$

Now we can define another norm, $\|\cdot\|^{b2}$ for tableau sets. This norm will be used to prove that rules of block 2 can be applied only finitely often in succession to a tableau set.

Definition 5.6.11 (norm $\|\cdot\|^{b2}$ for finite tableau sets) *Let $\Gamma_{\mathcal{T}}$ be a finite tableau set that has the regular tableau property. We define the size $\|\Gamma_{\mathcal{T}}\|^{b2}$ of $\Gamma_{\mathcal{T}}$ as follows:*

1. *The set of possible R_i-related pairs of tableau labels:*
   $X_1 := \{\mathfrak{l}\mathsf{R}_i\mathfrak{l}' \mid \mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_{\mathcal{T}}), i \in \mathsf{Ag}\}$.

The set of R_i*-related pairs of tableau labels occurring in* $\Gamma_{\mathcal{T}}$*:*
$Y_1 := X_1 \cap \Gamma_{\mathcal{T}}$.

2. *The set of possible knowledge formulae and propositions:*
$$\begin{aligned} X_2 := \ & \{\mathfrak{l} \vdash K_i\phi \mid K_i\phi \in sub(\Gamma_{\mathcal{T}}), \mathfrak{l} \in \mathfrak{label}(\Gamma_{\mathcal{T}})\} \\ & \cup \{\mathfrak{l} \vdash \neg K_i\phi \mid \neg K_i\phi \in sub(\Gamma_{\mathcal{T}}), \mathfrak{l} \in \mathfrak{label}(\Gamma_{\mathcal{T}})\} \\ & \cup \{\mathfrak{l} \vdash P \mid P \in sub(\Gamma_{\mathcal{T}}), \mathfrak{l} \in \mathfrak{label}(\Gamma_{\mathcal{T}})\} \\ & \cup \{\mathfrak{l} \vdash \neg P \mid \neg P \in sub(\Gamma_{\mathcal{T}}), \mathfrak{l} \in \mathfrak{label}(\Gamma_{\mathcal{T}})\} \end{aligned}$$
The set of knowledge formulae and propositions occurring in $\Gamma_{\mathcal{T}}$*:*
$Y_2 := X_2 \cap \Gamma_{\mathcal{T}}$.

3. *The set of possible next-formulae of type* $K_i\top$*:*
$X_3 := \{\mathfrak{l} \vdash \langle a \rangle_i \top \mid \mathfrak{l} \in \mathfrak{label}(\Gamma_{\mathcal{T}}), a \in \mathcal{O}, i \in Ag\}$.
The set of next-formulae occurring in $\Gamma_{\mathcal{T}}$*:*
$Y_3 := X_3 \cap \Gamma_{\mathcal{T}}$.

We define the size $\|\Gamma_{\mathcal{T}}\|^{b2}$ *of a tableau set* $\Gamma_{\mathcal{T}}$ *as the sum of the three distances:*

$$\|\Gamma_{\mathcal{T}}\|^{b2} := \sum_{i \in \{1,2,3\}} |X_i| - |Y_i|$$

where $|A|$ *denotes of the number of elements in* A.

Since the size $\|\Gamma_{\mathcal{T}}\|^{b2}$ of a finite tableau set $\Gamma_{\mathcal{T}}$ is a natural number and $>$ is a Noetherian order on natural numbers, it holds that if the application of a rule of block 2 strictly decreases the size of a tableau set $\Gamma_{\mathcal{T}}$, then rules of block 2 can be applied only finitely many times in succession.

We now prove that each application of a rule of block 2 strictly decreases the size $\|\Gamma_{\mathcal{T}}\|^{b2}$ of the tableau set.

Proof of lemma 5.6.8:

Let $\underline{\Gamma}$ denote the numerator of a rule and $\overline{\Gamma}$ denotes the considered denominator.

1. Suppose, $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l}R_i\mathfrak{l}', \mathfrak{l}'R_i\mathfrak{l}\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l}R_i\mathfrak{l}'\}$ by the application of rule (**TR 10**).

   By definition 5.5.1 of a tableau, a tableau rule can only be applied if all denominators are different from the numerator. This means that $\mathfrak{l}'R_i\mathfrak{l} \notin \underline{\Gamma}$, otherwise, the rule could not have been applied. This implies that $\|\underline{\Gamma}\|^{b2} > \|\overline{\Gamma}\|^{b2}$.

2. Suppose, $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l}R_i\mathfrak{l}', \mathfrak{l}'R_i\mathfrak{l}'', \mathfrak{l}R_i\mathfrak{l}''\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l}R_i\mathfrak{l}', \mathfrak{l}'R_i\mathfrak{l}''\}$ by the application of rule (**TR 11**).

   By definition 5.5.1 of a tableau, $\mathfrak{l}R_i\mathfrak{l}'' \notin \underline{\Gamma}$, otherwise, the rule could not have been applied. This implies that $\|\underline{\Gamma}\|^{b2} > \|\overline{\Gamma}\|^{b2}$.

3. Suppose, $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l}R_i\mathfrak{l}', \mathfrak{l} \vdash \chi, \mathfrak{l}' \vdash \chi\}$ with χ of the form $K_i\phi, \neg K_i\phi, P$ or $\neg P$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l}R_i\mathfrak{l}', \mathfrak{l} \vdash \chi\}$ by the application of either rule (**TR 12**) or rule (**TR 13**).

   By definition 5.5.1 of a tableau, $\mathfrak{l}' \vdash \chi \notin \underline{\Gamma}$, otherwise, the rule could not have been applied. This implies that $\|\underline{\Gamma}\|^{b2} > \|\overline{\Gamma}\|^{b2}$.

4. Suppose, $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \xrightarrow{a} \mathfrak{l}a, \mathfrak{l}R_i\mathfrak{l}a \mid i \notin ag(a)\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \xrightarrow{a} \mathfrak{l}a\}$ by the application of rule (**TR 24**).

   By definition 5.5.1 of a tableau, there must be at least one $i \in ag(a)$ with $\mathfrak{l}R_i\mathfrak{l}a \notin \underline{\Gamma}$, otherwise the rule could not have been applied. This implies that $\|\underline{\Gamma}\|^{b2} > \|\overline{\Gamma}\|^{b2}$.

5. Suppose, $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \xrightarrow{a} \mathfrak{l}a, \mathfrak{l}R_i\mathfrak{l}', \mathfrak{l}' \vdash \langle a \rangle_i \top \mid i \notin ag(a)\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \xrightarrow{a} \mathfrak{l}a, \mathfrak{l}R_i\mathfrak{l}' \mid i \in ag(a)\}$ by the application of rule (**TR 25**).

   By definition 5.5.1 of a tableau, there must be at least one $i \in ag(a)$ with $\mathfrak{l}' \vdash \langle a \rangle_i \top \notin \underline{\Gamma}$, otherwise the rule could not have been applied. This implies that $\|\underline{\Gamma}\|^{b2} > \|\overline{\Gamma}\|^{b2}$.

6. Suppose, $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \xrightarrow{a} \mathfrak{l}a, \mathfrak{l}' \xrightarrow{a} \mathfrak{l}'a, \mathfrak{l}R_i\mathfrak{l}', \mathfrak{l}aR_i\mathfrak{l}'a \mid i \notin ag(a)\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \xrightarrow{a} \mathfrak{l}a, \mathfrak{l}' \xrightarrow{a} \mathfrak{l}'a, \mathfrak{l}R_i\mathfrak{l}' \mid i \notin ag(a)\}$ by the application of rule (**TR 26**).

   By definition 5.5.1 of a tableau, there must be at least one $i \in ag(a)$ with $\mathfrak{l}aR_i\mathfrak{l}'a \notin \underline{\Gamma}$, otherwise the rule could not have been applied. This implies that $\|\underline{\Gamma}\|^{b2} > \|\overline{\Gamma}\|^{b2}$.

□

Finally, we have to show that also rules of block 3 may be applied only finitely many times in succession.

Proposition 5.6.12 *Let* $\Gamma_{\mathcal{T}}$ *be a finite tableau set having the regular tableau property. Rules of block 3 can be applied only a finite number of times to* $\Gamma_{\mathcal{T}}$ *in succession.*

Again, as in the cases before, we define a norm for tableau sets and a Noetherian ordering for this norm and show that each application of a rule of block 3 strictly decreases the size of the set.

Rule (**TR 14**) requires an unmarked tableau label $\mathfrak{l}$ together with a formula $\mathfrak{l} \vdash \langle \cdot \rangle_i \phi$ as principal formula. The application of rule (**TR 14**) marks the tableau label so that the rule can be applied only once for each tableau label. The side formulae of this rule can not be used as principal formulae again for any of the rules of block 3.

Rule **TR 9** requires an unmarked tableau formula $\mathfrak{l} \vdash \neg K_i\phi$ as principal formula. The application of rule (**TR 9**) marks its principal formula so that it can be applied only once for each tableau formula $\mathfrak{l} \vdash \neg K_i\phi$.

The problem is now that the rule **TR 9** adds the formula $\mathfrak{l}' \vdash \neg\phi$ for a new and unmarked tableau label $\mathfrak{l}'$ as side formula to the tableau set. This formula itself may again become a principal formula for either rule (**TR 14**) or rule (**TR 9**). In this case, however, we either have that $\phi = [\cdot]_i\psi$ or $\phi = K_i\psi$ for some agent i and some formula ψ is a subformula of $\neg K_i\phi$. By counting all occurrences of knowledge operators K_j in the tableau formula $\neg K_i\phi$ and multiplying them by 2 (each knowledge operator may create one next formula that can become principal formula in rule (**TR 14**)), we can estimate an upper bound of the number of new principal formulae for rules (**TR 9**) and (**TR 14**) that can be generated be rule (**TR 9**).

Thus, for norm $\|\cdot\|^{b3}$ we simply count the number of knowledge symbols K_j occurring in unmarked negated knowledge formulae and the number of unmarked tableau labels $\mathfrak{l}$ for which there exists a tableau formula of type $\langle\cdot\rangle_i\phi$.

Definition 5.6.13 (norm $\|\cdot\|^{b3}$ for finite tableau sets) *Let $\Gamma_{\mathcal{T}}$ be a finite tableau set that has the regular tableau property.*

Let $X(\Gamma_{\mathcal{T}}) := \{\mathfrak{l} \vdash \neg K_i\phi \mid \mathfrak{l} \vdash \neg K_i\phi \in \Gamma_{\mathcal{T}}$ (unmarked)$\}$ be the set of all unmarked negated knowledge formulae in $\Gamma_{\mathcal{T}}$.

Let $|\mathfrak{l} \vdash \neg K_i\phi|^{b3}$ denote the number of knowledge operators K_j with $j \in Ag$ occurring in $\neg K_i\phi$.

Further, let $\mathfrak{label}^u(\Gamma_{\mathcal{T}})$ denote the set of unmarked labels $\mathfrak{l}$ in $\mathfrak{label}(\Gamma_{\mathcal{T}})$ for which there exists a tableau formula $\mathfrak{l} \vdash \langle\cdot\rangle_i\phi \in \Gamma_{\mathcal{T}}$ for some $i \in Ag$ and some $\phi \in \mathcal{L}$.

We then define the size $\|\Gamma_{\mathcal{T}}\|^{b3}$ as follows:

$$\|\Gamma_{\mathcal{T}}\|^{b3} := 2 \cdot \Big(\sum_{x \in X(\Gamma_{\mathcal{T}})} |x|^{b3}\Big) + |\mathfrak{label}^u(\Gamma_{\mathcal{T}})|$$

Since the size $\|\Gamma_{\mathcal{T}}\|^{b3}$ of a finite tableau set $\Gamma_{\mathcal{T}}$ is a natural number and $>$ is a Noetherian ordering on natural numbers, it holds that if the application of a rule of block 3 strictly decreases the size of the tableau set $\Gamma_{\mathcal{T}}$, then rules of block 3 can be applied only finitely many times.

We now show that each application of a rule of block 3 to a finite tableau set $\Gamma_{\mathcal{T}}$ decreases the size $\|\Gamma_{\mathcal{T}}\|^{b3}$ of the tableau set.

Proof of proposition 5.6.12:

As before, $\underline{\Gamma}$ denotes the numerator of a rule and $\overline{\Gamma}$ denotes the considered denominator.

1. Suppose, $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota \vdash \neg\bar{K}_i\phi, \iota R_i \iota', \iota', \iota' \vdash \neg\phi\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota \vdash \neg K_i\phi | \iota \vdash \neg K_i\phi \text{ (unmarked)}\}$ by the application of rule (**TR 9**).

   The principal formula $\iota \vdash \neg K_i\phi$ contains one more knowledge operator K_j than the side formula $\iota' \vdash \phi$. The number of unmarked tableau labels ι for which there exists a next formula $\langle\cdot\rangle_i\phi$ is maximally increased in the denominator by one compared to the numerator. However, since, the number of knowledge operators counts twice ($2 \cdot (\sum_{x \in X(\underline{\Gamma})} |x|^{b3})$) it holds that $\|\underline{\Gamma}\|^{b3} > \|\overline{\Gamma}\|^{b3}$

2. Suppose, $\overline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota \vdash \langle . \rangle_i\phi, \iota \xrightarrow{a} \iota a, \iota, \iota a\}$ is derived from $\underline{\Gamma} = \Gamma_{\mathcal{T}} \cup \{\iota \vdash \langle . \rangle_i\phi, \iota \text{(unmarked)}\}$ by the application of rule (**TR 14**).

   The number of knowledge operators occurring in tableau formulae starting with a negative knowledge operator, i.e. formulae of type $\iota \vdash \neg K_i\phi$, remains the same when rule (**TR 14**) is applied. The number of unmarked tableau labels ι for which there exists a tableau formula $\iota \vdash \langle\cdot\rangle_i\phi$ for some $i \in \mathsf{Ag}$ and some $\phi \in \mathcal{L}$ decreases by 1 by the application of rule (**TR 14**).

   Consequently, it holds that $\|\underline{\Gamma}\|^{b3} > \|\overline{\Gamma}\|^{b3}$.

□

In propositions 5.6.3, 5.6.8 and 5.6.12 we have shown for each block of rules that for each finite tableau set $\Gamma_{\mathcal{T}}$ that has the regular tableau property rules of only one block can be applied only finitely often in succession. This directly implies the proof of lemma 5.6.2.

Proof of lemma 5.6.2:
Lemma 5.6.2 follows directly from propositions 5.6.3, 5.6.8 and 5.6.12.

□

Let $\mathcal{T}(\Gamma_{\mathcal{T}})$ be a regular tableau for a finite tableau set $\Gamma_{\mathcal{T}}$, let π be an accepted path in $\mathcal{T}(\Gamma_{\mathcal{T}})$ and let ν be a node on π.

According to the tableau definition 5.5.1 rules are applied in a kind of "block order": First, rules of block 1 are applied as long as possible, then rules of block are applied as long as possible then rules of block 3, afterwards rules of block 1 again and so forth. By lemma 5.6.2 we know that rules of each block can be applied only finitely many times to a finite tableau set. Consequently, if $\Gamma(\nu)$ is an instance of the numerator of a tableau rule, i.e. if a tableau rule is enabled for $\Gamma(\nu)$, then on each accepted path starting from $\Gamma(\nu)$ there must either be another tableau rule that disables this tableau rule or this rule will eventually be applied and thus satisfied or this rule will never be applied but eventually be satisfied by another rule. It is easy to verify that only the axioms (**TR 1**) and (**TR 2**) and tableau rule (**TR 17**) can disable other tableau rules. The axiom and the rule, however, directly lead to rejected paths and can thus not occur on accepted tableau paths.

Consequently, it holds that on each accepted tableau path all enabled rules will eventually be satisfied.

Proposition 5.6.14 (satisfaction of enabled rules) *Let $\mathcal{T}$ be a regular tableau for a set of formulae $\Gamma_0^{\mathcal{T}}$, let π be an accepted path in $\mathcal{T}$, let ν be a node in π and let $\Gamma(\nu)$ be the tableau set occurring in node ν.*

If in node ν a tableau rule $\frac{\Gamma_{num}}{d_1 \mid \ldots \mid d_n}$ is enabled then it will either be applied or satisfied after a finite number of steps.

Proof: Since π is an accepted path, neither the axioms (**TR 1**) and (**TR 2**) nor rule (**TR 17**) has been applied to a node in this path.

Further, none of the rules disables another rule: This is obvious for all rules, that do not delete the principal formulae. These are rules (**TR 6**), (**TR 9**), (**TR 10**), (**TR 11**), (**TR 12**), (**TR 13**), (**TR 14**), (**TR 24**), (**TR 25**) and rule (**TR 26**). It is also obvious that rules (**TR 3**), (**TR 4**), (**TR 5**), (**TR 7**) and (**TR 8**) do not disable another rule as their principal formulae do not occur as principal formulae in any other rule.

For rules (**TR 15**), (**TR 16**), (**TR 18**), (**TR 19**), (**TR 20**), (**TR 21**), (**TR 22**), (**TR 23**) we need the argument, that for each tableau label ι there can be only one tableau label ι' and one operation $a \in \mathcal{O}$ with $\iota \xrightarrow{a} \iota'$ (see corollary 5.5.5). Hence, each tableau formula $\iota \vdash \langle\cdot\rangle_i\phi$ or $\iota \vdash [\cdot]_i\phi$ can be a principal formula for only one of the above rules (**TR 15**) to (**TR 23**). Consequently, the application of one of the above rules cannot disable the application of another rule by deleting one of the principal formulae.

□

We now observe a number of properties concerning structure formulae of the type $\iota R_i \iota'$.

The tableau rules include two rules, namely rule (**TR 10**) and rule (**TR 11**), that are responsible for creating a transitive and symmetric closure of the R_i-rules, in the sense, that if for an accepted tableau path π the set of formulae Γ_π contains a structure formula $\iota R_i \iota'$ then it also contains the formula $\iota' R_i \iota$ and if it contains structure formulae $\iota R_i \iota'$ and $\iota' R_i \iota''$ then it also contains the formula $\iota R_i \iota''$. However, since the application of the rules is nondeterministic, we often simply talk about R_i-sequences and R_i-equivalence between tableau labels.

Definition 5.6.15 (R_i-sequence, R_i-equivalence) *Let $\mathcal{T}$ be a regular tableau and let π be a path of $\mathcal{T}$.*

Let $l, l' \in \mathfrak{label}(\Gamma_\pi)$ be two tableau labels and let $i \in \mathsf{Ag}$ be an agent.

We call lR_i^*l' an R_i-sequence of Γ_π if there exist tableau labels $l_0 \dots l_n$ such that $l = l_0, l' = l_n$ and for all $k \in \{0, \dots, n-1\}$ it holds that $l_k R_i l_{k+1} \in \Gamma_\pi$ or $l_{k+1} R_i l_k$.

A tableau label $l \in \mathfrak{label}(\Gamma_\pi)$ is R_i-**equivalent** *to tableau label $l' \in \mathfrak{label}(\Gamma_\pi)$ iff there exists an R_i-sequence lR_i^*l' in Γ_π.*

We can observe, that the relation between structure formulae of the type $l \xrightarrow{a} l'$ and structure formulae of the type lR_il' is more restrictive in tableau sets $\Gamma_\mathcal{T}$ that have the regular tableau property than the relation between temporal relations and the R_i-indistinguishability relations in a model:
If two R_i-equivalent tableau labels $l, l' \in \mathfrak{label}(\Gamma_\mathcal{T})$ belong to the same maximal chain in $\Gamma_\mathcal{T}$ then either there exists a sub chain $l \xrightarrow{a_0} l_1 \xrightarrow{a_1} \dots \xrightarrow{a_n} l'$ or there exists a sub chain $l' \xrightarrow{a_0} l_1 \xrightarrow{a_1} \dots \xrightarrow{a_n} l$ in $\Gamma_\mathcal{T}$ and for all $k \leq n$ it holds that $a_k \notin \mathcal{O}_i$.

Proposition 5.6.16 *Let $\mathcal{T}$ be a regular tableau and let π be an accepted path of $\mathcal{T}$. The following holds:*

For each pair of tableau labels $l, l' \in \mathfrak{label}(\Gamma_\pi)$ and for every agent $i \in \mathsf{Ag}$ it holds that

- *if there exists a (undirected) sequence lR_i^*l' in Γ_π and*
- *if there exists a chain $l \xrightarrow{o_0} \dots \xrightarrow{o_n} l'$ in Γ_π*

then $o_j \notin \mathcal{O}_i$ for all $0 \leq j \leq n$.

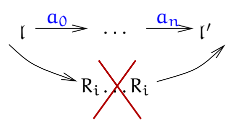


Figure 5.4: If two labels are connected by a chain that contains operations of $\mathcal{O}_i$, they cannot be R_i-equivalent.

None of the tableau rules that can occur on an accepted tableau path deletes structure formulae. The only rules by which structure formulae can be deleted, are the axioms (rule (**TR 1** and rule (**TR 2**)) and rule (**TR 17**). However, none of these rules can occur on an accepted tableau path, the axioms cannot occur on an accepted tableau path by definition of accepted paths and rule (**TR 17**) cannot occur on an accepted path because the only rule that can be applied after rule (**TR 17**) is the axiom (**TR 2**). If the structure formulae of path π violate proposition 5.6.16 then there must already exist a finite prefix $\nu_0\nu_1\ldots\nu_k$ of π that violates the proposition. We prove this proposition by induction over the length of the tableau path. Most of the tableau rules do not change the set of the structure formulae. For these rules it is obvious, that if the proposition was fulfilled for the set of formulae before the application of the rule, then it is also fulfilled afterwards. Thus we really only have to do the proof for those rules, that modify the set of structure formulae in the tableau.

Proof of lemma 5.6.16:

Let $\mathcal{T}$ be a regular tableau and let π be an accepted path of $\mathcal{T}$. We show, that for every prefix $\nu_0\nu_1\ldots\nu_k$ of π property $P(k)$ holds, where $P(k)$ is defined as follows: For each pair of tableau labels $\mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_k)$ and for every agent $i \in Ag$ it holds that

- if $\mathfrak{l}$ and $\mathfrak{l}'$ are R_i-equivalent in Γ_k and
- if there exists a chain $\mathfrak{l} \xrightarrow{o_0} \ldots \xrightarrow{o_n} \mathfrak{l}'$ in Γ_k

then for all $0 \leq j \leq n$ it holds that $o_j \notin \mathcal{O}_i$.

Induction begin: $P(0)$ holds:
$\Gamma_0 = \Gamma^0_{\mathcal{T}}$ contains only one label and does not contain any structure formulae.

Induction hypothesis:
Let $\nu_0\nu_1\ldots\nu_k$ be a prefix of π. Then $P(k)$ holds.

Induction step:
We make a case differentiation over the rules of the tableau system and show, that $P(k+1)$ holds.

1. Γ_{k+1} cannot be derived from Γ_k be application of rules (**TR 1**), (**TR 2**) or (**TR 17**) because these rules do not occur on accepted paths.
2. If Γ_{k+1} is derived from Γ_k by application of rules (**TR 3**), (**TR 4**), (**TR 5**), (**TR 6**), (**TR 7**), (**TR 8**), (**TR 12**), (**TR 13**), (**TR 15**), (**TR 16**), (**TR 18**), (**TR 19**), (**TR 20**), (**TR 21**), (**TR 22**), (**TR 23**) and (**TR 25**) then Prop(Γ_{k+1}) holds because none of them changes the set of the structure formulae.
3. Γ_{k+1} is derived from Γ_k by application of rule (**TR 9**).
   Suppose, $\Gamma_k = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \vdash \neg K_i\phi\}$
   $\Longrightarrow$ $\Gamma_{k+1} = \Gamma_k \cup \{\mathfrak{l} R_i \mathfrak{l}', \mathfrak{l}' \vdash \neg\phi, \mathfrak{l}'\}$ for a *new* label $\mathfrak{l}'$
   $\Longrightarrow$ (* because $\mathfrak{l}'$ is a new label *)
   for all $o \in \mathcal{O}$ and for all $\mathfrak{l}'' \in \mathfrak{label}(\Gamma_{k+1}) : \mathfrak{l}'' \xrightarrow{o} \mathfrak{l}' \notin \Gamma_{k+1}$ and $\mathfrak{l}' \xrightarrow{o} \mathfrak{l}'' \notin \Gamma_{k+1}$
   $\Longrightarrow$ (* with supposition that $P(k)$ holds *)
   $P(k+1)$ holds

4. Γ_{k+1} is derived from Γ_k by application of rule (**TR 10**).
   Suppose $\Gamma_k = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l}R_i\mathfrak{l}'\}$
   $\Longrightarrow$ $\Gamma_{k+1} = \Gamma_k \cup \{\mathfrak{l}'R_i\mathfrak{l}\}$
   (* rule (**TR 10**) does not change R_i-reachability *)
   $\Longrightarrow$ $P(k+1)$ holds

5. Γ_{k+1} is derived from Γ_k by application of rule (**TR 11**).
   Suppose $\Gamma_k = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l}R_i\mathfrak{l}', \mathfrak{l}'R_i\mathfrak{l}''\}$
   $\Longrightarrow$ $\Gamma_{k+1} = \Gamma_k \cup \{\mathfrak{l}R_i\mathfrak{l}''\}$
   $\Longrightarrow$ (* because rule (**TR 11**) does not change R_i-reachability *)
   $P(k+1)$ holds

6. Γ_{k+1} is derived from Γ_k by application of rule (**TR 24**).
   Suppose $\Gamma_k = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \xrightarrow{\alpha} \mathfrak{l}\alpha\}$.
   $\Longrightarrow$ $\Gamma_{k+1} = \Gamma_k \cup \{\mathfrak{l}R_i\mathfrak{l}\alpha \mid i \notin ag(\alpha)\}$
   $\Longrightarrow$ We distinguish between agents $j \in ag(\alpha)$ and $j \notin ag(\alpha)$:

   a) let $j \in ag(\alpha)$:
      for all $\mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_{k+1})$: $\mathfrak{l}R_j\mathfrak{l}' \in \Gamma_{k+1}$ iff $\mathfrak{l}R_j\mathfrak{l}' \in \Gamma_k$ and $\mathfrak{l} \xrightarrow{\alpha} \mathfrak{l}' \in \Gamma_{k+1}$ iff $\mathfrak{l} \xrightarrow{\alpha} \mathfrak{l}' \in \Gamma_k$
   b) let $j \notin ag(\alpha)$:
      $\Longrightarrow$ (* because $\mathfrak{l} \xrightarrow{\alpha} \mathfrak{l}\alpha$ and with corollary 5.5.6 *)
      there does not exist a path $\mathfrak{l} \xrightarrow{o_0} \ldots \xrightarrow{o_k} \ldots \xrightarrow{o_n} \mathfrak{l}\alpha$ with $o_k \in \mathcal{O}_j$

   $\Longrightarrow$ $P(k+1)$ holds

7. Γ_{k+1} is derived from Γ_k by application of rule (**TR 26**).
   $\Gamma_k = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \xrightarrow{\alpha} \mathfrak{l}\alpha, \mathfrak{l}' \xrightarrow{\alpha} \mathfrak{l}'\alpha, \mathfrak{l}R_i\mathfrak{l}' \mid i \in ag(\alpha)\}$
   $\Longrightarrow \Gamma_{k+1} = \Gamma_k \cup \{\mathfrak{l}\alpha R_i\mathfrak{l}'\alpha \mid i \in ag(\alpha)\}$

   We distinguish between two cases.

   a) $\mathfrak{l} \xrightarrow{\alpha} \mathfrak{l}\alpha$ and $\mathfrak{l}' \xrightarrow{\alpha} \mathfrak{l}'\alpha$ belong to different maximal chains:

$$\begin{tikzcd}
\mathfrak{l}_o \arrow[r] & \cdots \arrow[r] & \mathfrak{l} \arrow[r, "\alpha"] \arrow[d, "R_i", "{\substack{\text{for all}\\ i\in ag(\alpha)}}"'] & \mathfrak{l}\alpha \arrow[r] \arrow[d, "R_i"] & \cdots \\
\mathfrak{l}'_o \arrow[r] & \cdots \arrow[r] & \mathfrak{l}' \arrow[r, "\alpha"] & \mathfrak{l}'\alpha \arrow[r] & \cdots
\end{tikzcd}$$

$\Longrightarrow$ (* by lemma 5.5.4 and corollary 5.5.5 *)
there does not exist a chain $\chi \in \Gamma_{k+1}$ such that both $\mathfrak{l}$ and $\mathfrak{l}'$ belong to χ.
$\Longrightarrow$ there does not exist a chain χ in Γ_{k+1} with $\chi = \mathfrak{l}a \xrightarrow{a_o} \ldots \xrightarrow{a_n} \mathfrak{l}'a$
$\Longrightarrow$ (* because $P(k)$ holds *)
$P(k+1)$ holds.

b) $\mathfrak{l} \xrightarrow{a} \mathfrak{l}a$ and $\mathfrak{l}' \xrightarrow{a} \mathfrak{l}'a$ belong to the same maximal chain:

$$\begin{tikzcd}
\mathfrak{l}_o \arrow[r] & \cdots \arrow[r] & \mathfrak{l} \arrow[r, "a"] \arrow[rrr, bend right, "R_i"', "\text{forall } i\in ag a"'] & \mathfrak{l}a \arrow[r] \arrow[rrr, bend left, "R_i", "\text{for all } i\in ag a"'] & \cdots \arrow[r] & \mathfrak{l}' \arrow[r, "a"] & \mathfrak{l}'a \arrow[r] & \cdots
\end{tikzcd}$$

Suppose, $P(k+1)$ does not hold.
$\Longrightarrow$ (* because $P(k)$ holds and rule (**TR 26**) is applied *)
w.l.o.g. there must exist a chain χ in Γ_{k+1} and an $i \in ag(a)$ with $\chi = \mathfrak{l} \xrightarrow{a} \mathfrak{l}a \xrightarrow{a_0} \cdots \xrightarrow{a_k} \cdots \xrightarrow{a_n} \mathfrak{l}' \xrightarrow{a} \mathfrak{l}'a$ with $a_k \in \mathcal{O}_i$
$\Longrightarrow$ (* because $\mathfrak{l}R_i\mathfrak{l}' \in \Gamma_k$ *)
contradiction to the assumption, that $P(k)$ holds.

8. Γ_{k+1} is derived from Γ_k by application of rule (**TR 14**).
$\Gamma_k = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \vdash \langle\cdot\rangle_i\phi\}$
$\Longrightarrow$ $\Gamma_{k+1} = \Gamma_k \cup \{\mathfrak{l} \xrightarrow{a} \mathfrak{l}a, \mathfrak{l}a\}$ for any $a \in \mathcal{O}$ and for a *new* tableau label $\mathfrak{l}a$.
$\Longrightarrow$ (* $\mathfrak{l}a \notin \mathfrak{label}(\Gamma_k)$ *)
there does not exist any R_i-sequence in Γ_{k+1} containing $\mathfrak{l}a$
$\Longrightarrow$ $P(k+1)$ holds.

We have shown, that whatever rule we apply in a node k on an accepted path π, if Γ_k has property P (i.e. $P(k)$ holds) then also Γ_{k+1} has property P (i.e. $P(k+1)$ holds).

Since for all $n \in \mathbf{N}$ we have that $\Gamma_n \subset \Gamma_{n+1}$, it holds that if Γ_π contains a subset of structure rules that violate lemma 5.6.16, then this violation subset must already be contained in a finite prefix of π, Γ_m. We have shown, that such a finite prefix does not exist and thus lemma 5.6.16 holds.

$\square$

Proposition 5.6.17 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be a regular tableau for a set of formulae $\Gamma_0^{\mathcal{T}}$ and let π be a path of $\mathcal{T}(\Gamma_0^{\mathcal{T}})$.*

For all tableau labels $\mathfrak{l}, \mathfrak{l}'$ with $\mathfrak{l} \neq \mathfrak{l}'$ that belong to the same maximal chain χ there exists an agent $i \in Ag$ such that $\mathfrak{l}R_i\mathfrak{l}' \notin \Gamma_\pi$.

This proposition follows almost directly from lemma 5.6.16 shown above.

Proof:
Suppose, $\mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_\pi)$ and $\mathfrak{l} \neq \mathfrak{l}'$. If $\mathfrak{l}$ and $\mathfrak{l}'$ belong to the same maximal chain χ then w.l.o.g there exists a subchain $\mathfrak{l} \xrightarrow{o_0} \ldots \xrightarrow{o_n} \mathfrak{l}'$ of χ. According to lemma 5.6.16 it holds that there does not exist a sequence $\mathfrak{l}R_i^*\mathfrak{l}'$ in Γ_π for any $i \in \bigcup_{0 \leq j \leq n-1} ag(a_j)$. Since $\mathfrak{l} \neq \mathfrak{l}'$ it holds that $\bigcup_{0 \leq j \leq n-1} ag(a_j) \neq \emptyset$ and consequently, there exists at least one agent i such that $\mathfrak{l}R_i\mathfrak{l}' \notin \Gamma_\pi$.

□

Let us now make another observation about the relation between structure formulae of the form $\mathfrak{l} \xrightarrow{a} \mathfrak{l}a$ and structure formulae of the form $\mathfrak{l}R_i\mathfrak{l}'$. This observation will come into use when proving completeness of the tableau system.

For two labels $\mathfrak{l}$ and $\mathfrak{l}'$ occurring on the same tableau path π it holds that if they do not belong to the same maximal chain there exists maximally one agent $i \in \mathsf{Ag}$ such that $\mathfrak{l}R_i\mathfrak{l}' \in \Gamma_\pi$. We can put it even stronger, for two different maximal chains, there exists at most one $i \in \mathsf{Ag}$ for which two labels belonging to these two chains are R_i-equivalent.

Lemma 5.6.18 *Let $\mathcal{T}$ be a regular tableau for a set of formulae $\Gamma_0^{\mathcal{T}}$ and let π be a path of $\mathcal{T}$. Further, let χ, χ' be maximal chains occurring in Γ_π with $\chi \neq \chi'$.*

There exists an agent $i \in \mathsf{Ag}$ such that for all agents $j \neq i$ and for all tableau labels $\mathfrak{l}$ belonging to chain χ and for all tableau labels $\mathfrak{l}'$ belonging to chain χ' it holds that $\mathfrak{l}R_j\mathfrak{l}' \notin \Gamma_\pi$.

We again use induction over the length of the tableau path. If the lemma does not hold, then there must exist two agents $i, j \in \mathsf{Ag}, i \neq j$, such that there exist labels $\mathfrak{l}_i, \mathfrak{l}_i', \mathfrak{l}_j, \mathfrak{l}_j'$ with $\mathfrak{l}_i, \mathfrak{l}_j$ belonging to chain χ and $\mathfrak{l}_i', \mathfrak{l}_j'$ belonging to chain χ', such that $\mathfrak{l}_iR_i\mathfrak{l}_i'$ and $\mathfrak{l}_jR_j\mathfrak{l}_j'$.

If these two structure formulae exist in Γ_π, then there must already be a finite prefix $\nu_0\nu_1\ldots\nu_n$ of π, such that $\mathfrak{l}_iR_i\mathfrak{l}_i' \in \Gamma_n$ and $\mathfrak{l}_jR_j\mathfrak{l}_j' \in \Gamma_n$.

We will show, that this is not the case for each finite prefix of π and thus cannot be the case for π either.

Proof of lemma 5.6.18:

For a prefix $\nu_0\nu_1\ldots\nu_n$ of π we define $P(n)$ to be following property:
$P(n)$ holds, :iff for each pair of chains χ, χ' in Γ_n with $\chi \neq \chi'$ there exists an agent $i \in \mathsf{Ag}$ such that for all agents $j \neq i$ and for all tableau labels $\mathfrak{l}$ belonging to a maximal chain χ and for all tableau labels $\mathfrak{l}'$ belonging to a maximal chain χ', it holds that $\mathfrak{l}R_j\mathfrak{l}' \notin \Gamma_n$.

Induction begin:
$\Gamma_0 = \Gamma^0_{\mathcal{I}}$ does not contain any structure formulae, thus $P(0)$ is true.

Induction hypothesis:
Suppose, $P(m)$ is true. We show, that then also $P(m+1)$ is true.

Induction step:
We distinguish among the various rules that may have been applied to derive $\Gamma(\nu_{m+1})$ from $\Gamma(\nu_m)$.

1. Suppose, Γ_{m+1} is derived from Γ_m by application of one of the rules (**TR 1**), (**TR 2**) or (**TR 17**). Then $P(m+1)$ trivially holds.
2. Suppose, Γ_{m+1} is derived from Γ_m by application of one of the following rules: (**TR 3**), (**TR 4**), (**TR 5**), (**TR 6**), (**TR 7**), (**TR 8**), (**TR 12**), (**TR 13**), (**TR 15**), (**TR 16**), (**TR 18**), (**TR 19**), (**TR 20**), (**TR 21**), (**TR 22**), (**TR 23**), (**TR 25**). None of these rules modifies the set of structure formulae. Thus, $P(m+1)$ holds.
3. Suppose, Γ_{m+1} is derived from Γ_m by the application of rule (**TR 9**):
   Suppose, $\Gamma_m = \Gamma \cup \{\mathfrak{l} \vdash \neg K_h\phi\}$ for any $h \in Ag$.
   Then $\Gamma_{m+1} = \Gamma_m \cup \{\mathfrak{l}R_h\mathfrak{l}', \mathfrak{l}' \vdash \neg\phi, \mathfrak{l}'\}$ with $\mathfrak{l}' \notin \mathfrak{label}(\Gamma_m)$.
   Since the label $\mathfrak{l}'$ is *new* to Γ_{m+1}, it does not occur in any maximal chain χ of Γ_m but is itself a new maximal chain. There does not exist any $R_{h'}$-sequence for $h' \neq h$ from any label $\mathfrak{l} \in \mathfrak{label}(\Gamma_m)$. Since $\mathfrak{l}R_h\mathfrak{l}'$ is the only new structure formula in Γ_{m+1}, the property $P(m+1)$ holds.
4. Suppose, Γ_{m+1} is derived from Γ_m by the application of rule (**TR 10**):
   Suppose, $\Gamma_m = \Gamma \cup \{\mathfrak{l}R_h\mathfrak{l}'\}$. Then $\Gamma_{m+1} = \Gamma_m \cup \{\mathfrak{l}'R_h\mathfrak{l}\}$. Since the application of this rule does not modify R_h-sequences, property $P(m+1)$ holds.
5. Suppose, Γ_{m+1} is derived from Γ_m by the application of rule (**TR 11**):
   Suppose, $\Gamma_m = \Gamma \cup \{\mathfrak{l}R_h\mathfrak{l}', \mathfrak{l}'R_h\mathfrak{l}''\}$. Then $\Gamma_{m+1} = \Gamma_m \cup \{\mathfrak{l}R_h\mathfrak{l}''\}$. Since the application of this rule does not modify R_h-sequences, property $P(m+1)$ holds.
6. Suppose, Γ_{m+1} is derived from Γ_m by the application of rule (**TR 24**):
   Suppose, $\Gamma_m = \Gamma \cup \{\mathfrak{l} \xrightarrow{a} \mathfrak{l}a\}$. Then $\Gamma_{m+1} = \Gamma_m \cup \{\mathfrak{l}R_h\mathfrak{l}a| \text{ for all } h \notin ag(a)\}$. Since $\mathfrak{l}$ and $\mathfrak{l}a$ belong to the same maximal chain, the application of this rule does not modify R_h-sequences between different chains. Thus, property $P(m+1)$ holds.
7. Suppose, Γ_{m+1} is derived from Γ_m by the application of rule (**TR 26**):
   Suppose, $\Gamma_m = \Gamma \cup \{\mathfrak{l} \xrightarrow{a} \mathfrak{l}a, \mathfrak{l}R_h\mathfrak{l}', \mathfrak{l}' \xrightarrow{a} \mathfrak{l}'a\}$. Then $\Gamma_{m+1} = \Gamma_m \cup \{\mathfrak{l}aR_h\mathfrak{l}'a\}$. Since $\mathfrak{l}$ and $\mathfrak{l}a$ as well as $\mathfrak{l}'$ and $\mathfrak{l}'a$ belong to the same chains χ and χ' respectively, property $P(m+1)$ holds.

8. Suppose, Γ_{m+1} is derived from Γ_m by the application of rule (**TR 14**): Suppose, $\Gamma_m = \Gamma_{\mathcal{T}} \cup \{\mathfrak{l} \vdash \langle\cdot\rangle_h \phi\}$, then $\Gamma_{m+1} = \Gamma_m \cup \{\mathfrak{l} \xrightarrow{a} \mathfrak{l}a, \mathfrak{l}a\}$ for a new label $\mathfrak{l}a$ and for any $a \in \mathcal{O}$. Since for all tableau labels $\mathfrak{l}' \in \mathfrak{label}(\Gamma_{m+1})$ and for all agents $i \in \mathsf{Ag}$ it holds that $\mathfrak{l}' R_i \mathfrak{l}a \notin \Gamma_{m+1}$ and $\mathfrak{l}a R_i \mathfrak{l}' \notin \Gamma_{m+1}$ the property $P(m+1)$ holds.

We have shown, that property P holds for every finite prefix of path π. Except for the axioms and rule (**TR 17**), there is no tableau rule that deletes structure formulae from a tableau set. Consequently, whenever two labels $\mathfrak{l}, \mathfrak{l}'$ belong to the same maximal chain in Γ_m, they will belong to the same maximal chain in Γ_π. Thus, property $P(\pi)$ holds. $\square$

Corollary 5.6.19 *Let $\mathcal{T}$ be a regular tableau for a set of formulae $\Gamma_0^{\mathcal{T}}$ and let π be a path of $\mathcal{T}$. Further, let χ, χ' be maximal chains occurring in Γ_π with $\chi \neq \chi'$, let $\mathfrak{l}$ be a tableau label of chain χ and let $\mathfrak{l}'$ be a tableau label of chain χ'.*

If $\mathfrak{l} R_i \mathfrak{l}' \in \Gamma_\pi$ for some agent $i \in \mathsf{Ag}$, then for all agents $j \neq i$ and for all tableau labels $\mathfrak{l}$ belonging to chain χ and for all tableau labels $\mathfrak{l}'$ belonging to chain χ' it holds that $\mathfrak{l} R_j \mathfrak{l}' \notin \Gamma_\pi$.

Remember that for the knowledge relations in models as defined in definition 3.3.2 page 29 we require that for each pair of situations of a run there is at least one agent i for which these two situations are not R_i^k-indistinguishable. We can make a similar observation in the tableau: for each pair of tableau labels $\mathfrak{l}, \mathfrak{l}'$, if there exists a chain χ such that both tableau labels occur in the chain, then there is at least one agent $i \in \mathsf{Ag}$, such that $\mathfrak{l} R_i \mathfrak{l}'$ is not in the considered tableau set. We will need this lemma when proving completeness of the tableau system.

Chapter 6

Soundness and Completeness of the Tableau System

In the previous chapter we have developed a tableau system for proving unsatisfiability of sets of tableau formulae. The tableau method should reject a regular set of tableau formulae $\Gamma_0^{\mathcal{T}}$ if and only if the set $\Gamma_0^{\mathcal{T}}$ is not $\mathcal{L}-$satisfiable. The two most interesting questions are the following:

- Does the existence of a rejecting tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ for a regular tableau set $\Gamma_0^{\mathcal{T}}$ imply that the tableau set $\Gamma_0^{\mathcal{T}}$ is not $\mathcal{L}-$satisfiable?
- Does the fact that a tableau set $\Gamma_0^{\mathcal{T}}$ is not $\mathcal{L}-$satisfiable imply that there exists a rejecting tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$?

In the next section we will answer the first question positively. The second question can be answered positively only in case we make certain (syntactic) restrictions for the logic: Recall that the logic $\mathcal{L}$ is defined as a family of sets of formulae.

Definition 4.1.1 allows formulae of the form $K_i\phi$ without putting any restriction on the type of ϕ. In this general case, we can find sets of formulae that are not $\mathcal{L}-$satisfiable though there does not exist a rejecting tableau. However, if we restrict the language such that only formulae of the form $K_i\phi$ are allowed for $\phi \in \Phi_{\{i\}}$ then there exists a rejecting tableau of each set of $\mathcal{L}-$satisfiable formulae.

6.1 Soundness

We call the tableau system *sound* if the existence of a rejecting tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ for a regular tableau set $\Gamma_0^{\mathcal{T}}$ implies that the tableau set $\Gamma_0^{\mathcal{T}}$ is not $\mathcal{L}-$satisfiable.

Let us first consider the question about soundness of the tableau system.

Theorem 6.1.1 (Soundness of the tableau system) *The given tableau system is sound: If a regular tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ of this system rejects a finite set of formulae $\Gamma_0^{\mathcal{T}}$, then $\Gamma_0^{\mathcal{T}}$ is not $\mathcal{L}-$satisfiable.*

The formal proof of this theorem follows at the end of this section. We first introduce some lemmas that will serve as building blocks for the proof of the main theorem.

The argumentation of the proof of theorem 6.1.1 is as follows:

1. Suppose, $\Gamma_0^{\mathcal{T}}$ is a finite $\mathcal{L}-$satisfiable set of tableau formulae and there exists a rejecting regular tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$.
2. Since $\Gamma_{\mathcal{T}}^0$ is $\mathcal{L}-$ satisfiable, there must exist a model $\mathcal{M}$ and a satisfying $\mathcal{L}$-embedding for $\Gamma_0^{\mathcal{T}}$ from the set of tableau labels of $\Gamma_0^{\mathcal{T}}$ (i.e. $\{\mathfrak{l}_{\mathfrak{o}}\}$) into the set of situations of $\mathcal{M}$.
3. In lemma 6.1.3 we show that from the existence of an $\mathcal{L}$-embedding for the root $\Gamma_0^{\mathcal{T}}$ of a tableau we can conclude the existence of a tableau path π in $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ such that there exists a satisfying $\mathcal{L}$-embedding $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ for Γ_π.
4. Now we distinguish between two cases:
    a) π is a finite path. Because of the assumption that $\mathcal{T}$ is a rejecting tableau, the leaf of the finite path π must be labelled with an axiom. This means that path π either contains a tableau formula $(\mathfrak{F}, \mathfrak{c}) \vdash \bot$ or two tableau formulae $(\mathfrak{F}, \mathfrak{c}) \vdash \phi$ and $(\mathfrak{F}, \mathfrak{c}) \vdash \neg\phi$. This, however, contradicts the existence of a satisfying $\mathcal{L}$-embedding for each node on the path.
    b) π is an infinite path. As $\mathcal{T}$ is rejected, we again distinguish between two cases:
        i. π is fair and contains an infinite $\mathcal{U}$-trace. In this case, lemma 6.1.6 shows that Γ_π is not $\mathcal{L}-$satisfiable, which is a contradiction to the conclusion drawn from lemma 6.1.3 above. Consequently, $\mathcal{T}$ cannot contain a fair path that has an infinite $\mathcal{U}$-trace.
        ii. π is an unfair path. Lemma 6.1.4 proves that unfair paths do not need to be considered any further: If Γ_π is $\mathcal{L}-$satisfiable then there also exists a fair path π' for which $\Gamma_{\pi'}$ is $\mathcal{L}-$satisfiable. This path is either finite and the leaf is labelled with an axiom or it contains an infinite $\mathcal{U}$-trace. However, both of these options have already been excluded above. Consequently, $\mathcal{T}$ cannot contain an unfair path π for which there exists a satisfying $\mathcal{L}$-embedding.

This proof-sketch shows that if $\Gamma_0^{\mathcal{T}}$ is $\mathcal{L}-$satisfiable then there cannot exist any rejected tableau $\mathcal{T}$ for $\Gamma_0^{\mathcal{T}}$ because a rejected tableau would require that each path π of the tableau is either

1. a finite path the leaf of which is labelled by an axiom, or

2. an infinite and unfair path, or

3. an infinite, fair path that contains an $\mathcal{U}$-trace.

All three options were excluded above. Consequently, there cannot exist any satisfying $\mathcal{L}$-embedding.

First we show that if for a regular tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ the root $\Gamma_0^{\mathcal{T}}$ is $\mathcal{L}-$satisfiable with some $\mathcal{L}$-embedding $\ell : \mathfrak{label}(\Gamma_0^{\mathcal{T}}) \longrightarrow |\mathcal{M}|$, then there exists a path $\pi = v_0 v_1 \ldots$ in the tableau such that

1. for every node v_m on π the tableau set $\Gamma(v_m)$ is $\mathcal{L}-$ satisfiable with some $\mathcal{L}$-embedding $\ell_m : \mathfrak{label}(\Gamma(v_m)) \longrightarrow |\mathcal{M}|$ and

2. for all tableau labels $\mathfrak{l} \in \mathfrak{label}(\Gamma_{m-1})$ it holds that $\ell_{m-1}(\mathfrak{l}) = \ell_m(\mathfrak{l})$.

Note, that this means that all tableau sets on this path can be embedded into the same model.

Lemma 6.1.2 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be a regular tableau and let $\mathcal{M}$ be a model.*
If there exists a satisfying $\mathcal{L}$-embedding

$$\ell_0 : \mathfrak{label}(\Gamma_0^{\mathcal{T}}) \longrightarrow |\mathcal{M}|$$

for $\Gamma_0^{\mathcal{T}}$ then there exists a path $\pi = v_0 v_1 v_2 \ldots$ in the tableau such that for each node v_m on this path there exists a satisfying $\mathcal{L}$-embedding $\ell_m : \mathfrak{label}(\Gamma_m) \longrightarrow |\mathcal{M}|$ with $\ell_m((\mathfrak{F}, \mathfrak{c})) = \ell_{m-1}((\mathfrak{F}, \mathfrak{c}))$ for all labels $(\mathfrak{F}, \mathfrak{c}) \in \mathfrak{label}(\Gamma_{m-1})$.

We prove this lemma by induction over the length of the tableau path and step by step identify the considered path. To do so, we must make a case differentiation among all 26 possible tableau rules. For most of the rules this proof is rather simple: We show that the $\mathcal{L}$-embedding for the numerator of a rule is also a satisfying $\mathcal{L}$-embedding for one of its denominators. Only if the tableau rule introduces a new tableau label, the embedding function must be adjusted. This is the case for the tableau rules (**TR 9**) and (**TR 14**).

Proof:
Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be a regular tableau, let $\mathcal{M}$ be a model and let $\ell_0 : \mathfrak{label}(\Gamma_0^{\mathcal{T}}) \longrightarrow |\mathcal{M}|$ be a satisfying $\mathcal{L}$-embedding for $\Gamma_0^{\mathcal{T}}$. We show that for each $n \in \mathbf{N}$ there exists a path $\pi_n = v_0 v_1 v_2 \ldots v_n$ in $\mathcal{T}$ that has property $\mathsf{P}(\pi_n)$, where we define property $\mathsf{P}(\pi_n)$ as follows:
For each node v_m on π_n there exists a satisfying $\mathcal{L}$-embedding $\ell_m : \mathfrak{label}(\Gamma_m) \longrightarrow |\mathcal{M}|$ with $\ell_m((\mathfrak{F}, \mathfrak{c})) = \ell_{m-1}((\mathfrak{F}, \mathfrak{c}))$ for all labels $(\mathfrak{F}, \mathfrak{c}) \in \mathfrak{label}(\Gamma_{m-1})$.

Induction begin:
ν_0 has the required property by assumption.

Induction hypothesis:
We have already identified a path π_m that has property $P(\pi_m)$.

Induction step:
We show that there exists an extension of π_m to π_{m+1} such that π_{m+1} has property $P(\pi_{m+1})$

1. Suppose, rule (**TR 1**) has been applied to node ν_m.
   There does not exist any satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \phi, (\mathfrak{F}, \mathfrak{c}) \vdash \neg\phi\}$. Thus, property $P(\pi_m)$ cannot hold, which contradicts the induction hypothesis. This implies that rule (**TR 1**) cannot have been applied in node ν_m.

2. Suppose, rule (**TR 2**) has been applied to node ν_m.
   There does not exist any satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \bot\}$. Thus, property $P(\pi_m)$ does not hold, which contradicts the induction hypothesis. This implies that rule (**TR 2**) cannot have been applied.

3. Suppose, rule (**TR 3**) has been applied to node ν_m.
   Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \top\}$. Rule (**TR 3**) has only one denominator and thus, node ν_m has only one possible successor ν_{m+1}. Then $\mathcal{L}$-embedding $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}}$, because $\Gamma(\nu_{m+1}) \subset \Gamma(\nu_m)$.
   Consequently, property $P(\pi_{m+1})$ holds.

4. Suppose, rule (**TR 4**) has been applied to node ν_m.
   Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \phi \wedge \psi\}$.
   $\Longrightarrow$ ($*$ by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding $*$)
   $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi \wedge \psi$
   $\Longrightarrow$ ($*$ by definition of the semantics of $\phi \wedge \psi$ $*$)
   $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \psi$
   $\Longrightarrow$ ($*$ by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding $*$)
   $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \phi, (\mathfrak{F}, \mathfrak{c}) \vdash \psi\}$
   Consequently, property $P(\pi_{m+1})$ holds.

5. Suppose, rule (**TR 5**) has been applied to node ν_m.
   Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \phi \vee \psi\}$.

$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi \vee \psi$
$\Longrightarrow$ (* by definition of the semantics of $\phi \vee \psi$ *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi$ or $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \psi$
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \phi\}$
or
$\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \psi\}$
We choose π_{m+1} accordingly.
Consequently, property $P(\pi_{m+1})$ holds.

6. Suppose, rule (**TR 6**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash K_i\phi\}$.
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\Longrightarrow$ $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} K_i\phi$
$\Longrightarrow$ (* by definition of the semantics of $K_i\phi$ and reflexivity of R_i^K *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} K_i\phi$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_{m+1}) = \Gamma(\nu_m) \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \phi\}$
Consequently, property $P(\pi_{m+1})$ holds.

7. Suppose, rule (**TR 7**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \phi\ \mathcal{U}_i\psi\}$.
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi\ \mathcal{U}_i\psi$
$\Longrightarrow$ (* by remark 2, page 44 and def. 4.2.1 *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \psi$ or $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} (\phi \wedge \neg\psi \wedge \langle\mathcal{O}\rangle_i\phi\ \mathcal{U}_i\psi)$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \psi\}$
or
$\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \phi \wedge \neg\psi \wedge \langle\mathcal{O}\rangle_i\phi\ \mathcal{U}_i\psi\}$
We choose π_{m+1} accordingly.
Consequently, property $P(\pi_{m+1})$ holds.

8. Suppose, rule (**TR 8**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \phi\mathcal{W}_i\psi\}$.

$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi \mathcal{W}_i \psi$
$\Longrightarrow$ (* by remark 3 page 44 and def. 4.2.1 *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \psi$ or $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} (\phi \wedge \neg\psi \wedge [\mathcal{O}_i]\phi \mathcal{W}_i \psi)$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \psi\}$
or
$\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \phi \wedge \neg\psi \wedge [\mathcal{O}_i]\phi \mathcal{W}_i \psi\}$
We choose π_{m+1} accordingly.
Consequently, property $P(\pi_{m+1})$ holds.

9. Suppose, rule (**TR 9**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i \phi\}$.
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \neg K_i \phi$
$\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\neg K_i \phi$ *)
there exists a situation $(F', c') \in |\mathcal{M}|$ such
that $(\ell_m((\mathfrak{F}, \mathfrak{c})), (F', c')) \in R_i^K$
and $(F', c') \models_{\mathcal{M}} \neg\phi$
$\Longrightarrow$ (* because $\mathfrak{label}(\Gamma(\nu_m)) \subset \mathfrak{label}(\Gamma(\nu_{m+1}))$ *)
we construct $\ell_{m+1} : \mathfrak{label}(\Gamma(\nu_{m+1})) \longrightarrow |\mathcal{M}|$ from ℓ_m in such a way that
$$\ell_{m+1}(\mathfrak{l}) := \begin{cases} \ell_m(\mathfrak{l}) & \text{for } \mathfrak{l} \in \mathfrak{label}(\Gamma_m) \\ (F', c') & \text{for } \mathfrak{l} \in \mathfrak{label}(\Gamma_{m+1}) \setminus \mathfrak{label}(\Gamma_m) \end{cases}$$
$\Longrightarrow$ ℓ_{m+1} is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma(\nu_m) \cup \{(\mathfrak{F}', \mathfrak{c}') \vdash \neg\phi, (\mathfrak{F}, \mathfrak{c}) R_i (\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}', \mathfrak{c}')\}$
Consequently, property $P(\pi_{m+1})$ holds.

10. Suppose, rule (**TR 10**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) R_i (\mathfrak{F}', \mathfrak{c}')\}$.
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying$\mathcal{L}$-embedding *)
$(\ell_m((\mathfrak{F}, \mathfrak{c})), \ell_m((\mathfrak{F}', \mathfrak{c}'))) \in R_i^K$
$\Longrightarrow$ (* because R_i^K is symmetric *)
$(\ell_m((\mathfrak{F}', \mathfrak{c}')), \ell_m((\mathfrak{F}, \mathfrak{c}))) \in R_i^K$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma(\nu_m) \cup \{(\mathfrak{F}', \mathfrak{c}') R_i (\mathfrak{F}, \mathfrak{c})\}$
Consequently, property $P(\pi_{m+1})$ holds.

11. Suppose, rule (**TR 11**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) =$

$\Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}', \mathfrak{c}')R_i(\mathfrak{F}'', \mathfrak{c}'')\}$.
$\Longrightarrow$ (* by definition 5.3.2 of $\mathcal{L}$-embedding *)
$(\ell_m((\mathfrak{F}, \mathfrak{c})), \ell_m((\mathfrak{F}', \mathfrak{c}'))) \in R_i^K$ and $(\ell_m((\mathfrak{F}', \mathfrak{c}')), \ell_m((\mathfrak{F}'', \mathfrak{c}''))) \in R_i^K$
$\Longrightarrow$ (* because R_i^K is transitive *)
$(\ell_m((\mathfrak{F}, \mathfrak{c})), \ell_m((\mathfrak{F}'', \mathfrak{c}''))) \in R_i^K$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma(\nu_m) \cup \{(\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}'', \mathfrak{c}'')\}$
Consequently, property $P(\pi_{m+1})$ holds.

12. Suppose, rule (**TR 12**) or rule (**TR 13**) has been applied to node ν_m.

    a) Suppose, $\ell : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}, \mathfrak{c}) \vdash \chi\}$ for $\chi \in \{P, \neg P | P \in \mathcal{P}_i\}$.
    $\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
    $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \chi$ and $(\ell_m((\mathfrak{F}, \mathfrak{c})), \ell_m((\mathfrak{F}', \mathfrak{c}'))) \in R_i^K$
    $\Longrightarrow$ (* by definition 3.3.4 of an interpretation *)
    $\ell_m((\mathfrak{F}', \mathfrak{c}')) \models_{\mathcal{M}} \chi$
    $\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
    $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
    $\Gamma(\nu_{m+1}) = \Gamma(\nu_m) \cup \{(\mathfrak{F}', \mathfrak{c}') \vdash \chi\}$
    Consequently, property $P(\pi_{m+1})$ holds.

    b) Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}, \mathfrak{c}) \vdash \chi\}$ for $\chi \in \{\neg K_i\phi, K_i\phi \mid \phi \in \mathcal{L}\}$.
    $\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
    $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \chi$ and $(\ell_m((\mathfrak{F}, \mathfrak{c})), \ell_m((\mathfrak{F}', \mathfrak{c}'))) \in R_i^K$
    $\Longrightarrow$ (* by definition of the semantics of $K_i\phi$ *)
    $\ell_m((\mathfrak{F}', \mathfrak{c}')) \models_{\mathcal{M}} \chi$
    $\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
    $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
    $\Gamma(\nu_{m+1}) = \Gamma(\nu_m) \cup \{(\mathfrak{F}', \mathfrak{c}') \vdash \chi\}$
    Consequently, property $P(\pi_{m+1})$ holds.

13. Suppose, rule (**TR 14**) has been applied to node ν_m.
    Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \langle . \rangle_i \phi\}$.

$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \langle . \rangle_i \phi$
$\Longrightarrow$ (* $\langle . \rangle_i \phi$ is an abbreviation for either $\langle a \rangle_i \phi$ or $\langle \mathcal{O} \rangle_i \phi$ *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \langle a \rangle_i \phi$ with $a \in \mathcal{O}_i$ or
$\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \langle \mathcal{O} \rangle_i \phi$
$\Longrightarrow$ (* by definition of the semantics of $\langle \mathcal{O} \rangle_i \phi$*)
there exists an $o \in \mathcal{O}_i$, s.t. $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \langle o \rangle_i \phi$
$\Longrightarrow$ (* by definition of the semantics of $\langle o \rangle_i \phi$ *)
there exist $(F, c'), (F, r) \in |\mathcal{M}|$, such that $(F, c') \equiv_i \ell_m((\mathfrak{F}, \mathfrak{c}))$
and $(F, c') \xrightarrow{o} (F, r)$ and $(F, r) \models_{\mathcal{M}} \phi$
$\Longrightarrow$ We distinguish between two cases:

First case: $\ell_m((\mathfrak{F}, \mathfrak{c})) = (F, c')$;
operation o is the next operation for all agents $j \in ag(o)$, i.e.
$\ell_m((\mathfrak{F}, \mathfrak{c})) = (F, c')$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{o} (F, r)$ and $(F, r) \models_{\mathcal{M}} \phi$.
As rule (**TR 14**) creates a denominator for each operation in $\mathcal{O}$,
we can choose ν_{m+1} such that for $(\mathfrak{F}, \mathfrak{co}) \in \mathfrak{label}(\Gamma_{m+1}) \setminus \mathfrak{label}(\Gamma_m)$
we have $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{o} (\mathfrak{F}, \mathfrak{co}) \in \Gamma(\nu_{m+1})$
and we construct ℓ_{m+1} as follows:

$$\ell_{m+1}(\mathfrak{l}) := \begin{cases} \ell_m(\mathfrak{l}) & \text{for } \mathfrak{l} \in \mathfrak{label}(\Gamma_m) \\ (F, r) & \text{for } \mathfrak{l} = (\mathfrak{F}, \mathfrak{co}) \end{cases}$$

Second case: $\ell_m((\mathfrak{F}, \mathfrak{c})) \neq (F, c')$;
there exists an agent $j \in ag(o), j \neq i$, such that
operation o is *not* the next operation for agent j,
but instead some operation $o' \in \mathcal{O}_j \setminus \mathcal{O}_i$ is, i.e.
there exist $j \in ag(o), o' \in \mathcal{O}_j \setminus \mathcal{O}_i$ and $(F, c'') \in |\mathcal{M}|$, such that
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{o'} (F, c'')$ and $c'' \subseteq c'$.
As rule (**TR 14**) creates a denominator for each operation in $\mathcal{O}$,
we can choose ν_{m+1} such that for
$(\mathfrak{F}, \mathfrak{co}') \in \mathfrak{label}(\Gamma_{m+1}) \setminus \mathfrak{label}(\Gamma_m)$ we have $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{o'} (\mathfrak{F}, \mathfrak{co}') \in \Gamma(\nu_{m+1})$
and construct ℓ_{m+1} from ℓ_m in such a way that

$$\ell_{m+1}(\mathfrak{l}) = \begin{cases} \ell_m(\mathfrak{l}) & \text{for } \mathfrak{l} \in \mathfrak{label}(\Gamma_m) \\ (F, c'') & \text{for } \mathfrak{l} = (\mathfrak{F}, \mathfrak{co}') \end{cases}$$

$\Longrightarrow$ ℓ_{m+1} is a satisfying $\mathcal{L}-$embedding for $\Gamma(\nu_{m+1})$

Consequently, property $P(\pi_{m+1})$ holds.

14. Suppose, rule (**TR 15**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{ca}), (\mathfrak{F}, \mathfrak{c}) \vdash \langle a \rangle_i \phi\}$.

$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell((\mathfrak{F}, \mathfrak{c}a))$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \langle a \rangle_i \phi$
$\Longrightarrow$ (* by definition of the semantics of $\langle a \rangle_i \phi$ *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and there exist situations $(F, c'), (F, r) \in |\mathcal{M}|$ such that
$\ell_m((\mathfrak{F}, \mathfrak{c})) \equiv_i (F, c')$ and $(F, c') \xrightarrow{a} (F, r)$ and $(F, r) \models_{\mathcal{M}} \phi$
$\Longrightarrow$ (* because of lemmas 3.2.4 and 4.3.2 on pages 25 and 41 *)
$\ell_m((\mathfrak{F}, \mathfrak{c}a)) \equiv_{ag(a)} (F, r)$ and $\ell((\mathfrak{F}, \mathfrak{c}a)) \models_{\mathcal{M}} \phi$
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}a) \vdash \phi\}$
Consequently, property $P(\pi_{m+1})$ holds.

15. Suppose, rule (**TR 16**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}) \vdash [a]_i \phi\}$.
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} [a]_i \phi$
$\Longrightarrow$ (* by definition of the semantics of $[a]_i \phi$ *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and
for all $(F, c'), (F, r) \in |\mathcal{M}|$, such that
$\ell_m((\mathfrak{F}, \mathfrak{c})) \equiv_i (F, c')$ and $(F, c') \xrightarrow{a} (F, r)$ holds that $(F, r) \models_{\mathcal{M}} \phi$
$\Longrightarrow$ $\ell_m((\mathfrak{F}, \mathfrak{c}a)) \models_{\mathcal{M}} \phi$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}a) \vdash \phi\}$
Consequently, property $P(\pi_{m+1})$ holds.

16. Suppose, rule (**TR 17**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}) \vdash \langle b \rangle_i \phi\}$ with $i \in ag(a), a \neq b$.
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \langle b \rangle_i \phi$
Let $\ell_m((\mathfrak{F}, \mathfrak{c})) = (F, c)$ and $\ell_m((\mathfrak{F}, \mathfrak{c}a)) = (F, ca)$.

$\Longrightarrow$ ($*$ by definition of the semantics of $\langle b \rangle_i \phi$ $*$)
$(F, c) \xrightarrow{a} (F, ca)$ and
there exist situations $(F, c'), (F, r) \in |\mathcal{M}|$ such that
$(F, c) \equiv_i (F, c')$ and $(F, c') \xrightarrow{b} (F, r)$
$\Longrightarrow$ ($*$ because $i \in ag(a) \cap ag(b)$ and i-events are totally ordered $*$)
contradiction to the induction hypothesis that
ℓ_m is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m)$
$\Longrightarrow$ there does not exist any satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m)$
$\Longrightarrow$ contradiction to the assumption that $P(\pi_m)$ holds.
$\Longrightarrow$ rule (**TR 17**) cannot have been applied.

17. Suppose, rule (**TR 18**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_\mathcal{T} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}) \vdash \langle b \rangle_i \phi\}$ with $i \notin ag(a)$, $a \neq b$.
$\Longrightarrow$ ($*$ by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \langle b \rangle_i \phi$ with $i \notin ag(a)$
$\Longrightarrow$($*$ by definition 3.2.3 $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \langle b \rangle_i \phi$
and $\ell_m((\mathfrak{F}, \mathfrak{c})) \equiv_i \ell_m((\mathfrak{F}, \mathfrak{c}a))$
$\Longrightarrow$($*$ by lemma 4.3.2 $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \langle b \rangle_i \phi$
and$\ell_m((\mathfrak{F}, \mathfrak{c}a)) \models_\mathcal{M} \langle b \rangle_i \phi$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_{m+1}) = \Gamma_\mathcal{T} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}a) \vdash \langle b \rangle_i \phi\}$

18. Suppose, rule (**TR 19**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_\mathcal{T} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}) \vdash [b]_i \phi\}$ with $i \notin ag(a)$, $a \neq b$.
$\Longrightarrow$ ($*$ by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} [b]_i \phi$ with $i \notin ag(a)$
$\Longrightarrow$($*$ by definition 3.2.3 $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \equiv_i \ell_m((\mathfrak{F}, \mathfrak{c}a))$
$\Longrightarrow$($*$ by lemma 4.3.2 $*$)
$\ell((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} [b]_i \phi$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_\mathcal{T} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}a) \vdash [b]_i \phi\}$
Consequently, property $P(\pi_{m+1})$ holds.

19. Suppose, rule (**TR 20**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_\mathcal{T} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}) \vdash \langle \mathcal{O} \rangle_i \phi\}$ with $i \in ag(a)$

$\Longrightarrow$ ($*$ by definition of $\langle\mathcal{O}\rangle_i\phi$ $*$)
ℓ_m is a satisfying $\mathcal{L}$-embedding for
$\Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}) \vdash \bigvee_{o \in \mathcal{O}_i} \langle o\rangle_i \phi\}$
$\Longrightarrow$ ($*$ by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a)), \ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \bigvee_{o \in \mathcal{O}_i} \langle o\rangle_i \phi$
$\Longrightarrow$ ($*$ by definition of the semantics of $\vee$ and $\langle o\rangle_i\phi$ $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$, and there exists $o \in \mathcal{O}_i$ and $(F, c), (F, r) \in |\mathcal{M}|$, such that
$\ell_m((\mathfrak{F}, \mathfrak{c})) \equiv_i (F, c')$ and $(F, c') \xrightarrow{o} (F, r)$ and $(F, r) \models_{\mathcal{M}} \phi$
$\Longrightarrow$ ($*$ because $a \in \mathcal{O}_i$ and E_i is totally ordered $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c}a)) \models_{\mathcal{M}} \phi$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}a) \vdash \phi\}$
Consequently, property $P(\pi_{m+1})$ holds.

20. Suppose, rule (**TR 21**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}) \vdash [\mathcal{O}]_i\phi\}$ with $i \in ag(a)$.
$\Longrightarrow$ ($*$ by definition of $[\mathcal{O}]_i\phi$ $*$)
ℓ_m is a satisfying $\mathcal{L}$-embedding for
$\Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}) \vdash \bigwedge_{o \in \mathcal{O}_i} [o]_i \phi\}$
$\Longrightarrow$ ($*$ by definition of 5.3.2 of a satisfying $\mathcal{L}$-embedding $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \bigwedge_{o \in \mathcal{O}_i} [o]_i \phi$
$\Longrightarrow$ ($*$ by definition of the semantics of $\wedge$ $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and for all $o \in \mathcal{O}_i$: $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} [o]_i\phi$
$\Longrightarrow$ ($*$ by definition of the semantics of $[o]_i\phi$ $*$)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and
for all $o \in \mathcal{O}_i, (F, c'), (F, r) \in |\mathcal{M}|$ it holds that
if $\ell_m((\mathfrak{F}, \mathfrak{c})) \equiv_i (F, c')$ and $(F, c') \xrightarrow{o} (F, r)$ then $(F, r) \models_{\mathcal{M}} \phi$
$\Longrightarrow$ ($*$ because $\ell_m((\mathfrak{F}, \mathfrak{c})) \equiv_i \ell_m((\mathfrak{F}, \mathfrak{c}))$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ $*$)
$\ell_m((F, ca)) \models_{\mathcal{M}} \phi$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}a) \vdash \phi\}$
Consequently, property $P(\pi_{m+1})$ holds.

21. Suppose, rule (**TR 22**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c}) \vdash \langle\mathcal{O}\rangle_i\phi\}$ with $i \notin ag(a)$.

$\Longrightarrow$ (* by definition of $\langle\mathcal{O}\rangle_i\phi$ *)
ℓ_m is a satisfying $\mathcal{L}$-embedding for
$\Gamma_{\mathcal{T}} \cup \{(\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \bigvee_{o\in\mathcal{O}_i}\langle o\rangle_i\phi\}$
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F},\mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F},\mathfrak{c}a)), \ell_m((\mathfrak{F},\mathfrak{c})) \models_{\mathcal{M}} \bigvee_{o\in\mathcal{O}_i}\langle o\rangle_i\phi$
$\Longrightarrow$ (* because $\ell_m((\mathfrak{F},\mathfrak{c})) \equiv_j \ell_m((\mathfrak{F},\mathfrak{c}a))$ with $j \notin ag(a)$ and thus $i \notin ag(a)$*)
$\ell_m((\mathfrak{F},\mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F},\mathfrak{c}a))$ and $\ell_m((\mathfrak{F},\mathfrak{c}a)) \models_{\mathcal{M}} \bigvee_{o\in\mathcal{O}_i}\langle o\rangle_i\phi$ with $i \notin ag(a)$
$\Longrightarrow$ (* by definition of $\langle\mathcal{O}\rangle_i\phi$*)
$\ell_m((\mathfrak{F},\mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F},\mathfrak{c}a))$ and $\ell_m((\mathfrak{F},\mathfrak{c}a)) \models_{\mathcal{M}} \langle\mathcal{O}\rangle_i\phi$ for $i \notin ag(a)$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \langle\mathcal{O}\rangle_i\phi\}$
Consequently, property $P(\pi_{m+1})$ holds.

22. Suppose, rule (**TR 23**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash [\mathcal{O}]_i\phi\}$ with $i \notin ag(a)$.
$\Longrightarrow$ (* by definition of $[\mathcal{O}]_i\phi$ *)
ℓ_m is a satisfying $\mathcal{L}$-embedding for
$\Gamma_{\mathcal{T}} \cup \{(\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \bigwedge_{o\in\mathcal{O}_i}[o]_i\phi$ for $i \notin ag(a)\}$
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F},\mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F},\mathfrak{c}a))$ and $\ell_m((\mathfrak{F},\mathfrak{c})) \models_{\mathcal{M}} \bigwedge_{o\in\mathcal{O}_i}[o]_i\phi$
$\Longrightarrow$ (* because $\ell_m((\mathfrak{F},\mathfrak{c})) \equiv_i \ell_m((\mathfrak{F},\mathfrak{c}a))$ for all $j \notin ag(a)$ *)
$\ell_m((\mathfrak{F},\mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F},\mathfrak{c}a))$ and $\ell_m((\mathfrak{F},\mathfrak{c}a)) \models_{\mathcal{M}} \bigwedge_{o\in\mathcal{O}_i}[o]_i\phi$
$\Longrightarrow$ $\ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma_{\mathcal{T}} \cup \{(\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \bigwedge_{o\in\mathcal{O}_i}[o]_i\phi\}$ in $\mathcal{M}$.
$\Longrightarrow$ (* by definition of $[\mathcal{O}]_i\phi$ *)
ℓ_{m+1} is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash [\mathcal{O}]_i\phi\}$
Consequently, property $P(\pi_{m+1})$ holds.

23. Suppose, rule (**TR 24**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a)\}$.

$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$
$\Longrightarrow$ (* by definition 3.3.2 and $\ell_m((\mathfrak{F}, \mathfrak{c})) \equiv_i \ell_m((\mathfrak{F}, \mathfrak{c}a))$ for all $i \notin ag(a)$ *)
$(\ell_m((\mathfrak{F}, \mathfrak{c})), \ell_m((\mathfrak{F}, \mathfrak{c}a))) \in R_i^K$ for all $i \notin ag(a)$
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_{m+1}) := \ell_m$ is a satisfying $\mathcal{L}$-embedding for
$\Gamma(\nu_{m+1}) = \Gamma(\nu_m) \cup \{(\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}, \mathfrak{c}a)$ for all $i \notin ag(a)\}$
Consequently, property $P(\pi_{m+1})$ holds.

24. Suppose, rule (**TR 25**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}')$ for all $i \in ag(a)\}$.
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and $(\ell_m((\mathfrak{F}, \mathfrak{c})), \ell_m((\mathfrak{F}', \mathfrak{c}'))) \in R_i^K$ for all $i \in ag(a)$
$\Longrightarrow$ (* by definition 3.3.2 condition 4 *)
there exists $(F', c'') \in |\mathcal{M}|$ such that $\ell_m((\mathfrak{F}', \mathfrak{c}')) \xrightarrow{a} (F', c'')$
$\Longrightarrow$ (* because $(F', c'') \models_{\mathcal{M}} \top$ is a tautology *)
$\ell_m((\mathfrak{F}', \mathfrak{c}')) \models_{\mathcal{M}} \langle a \rangle_i \top$ for each $i \in ag(a)$
$\Longrightarrow \ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}-$embedding for
$\Gamma(\nu_{m+1}) = \Gamma(\nu_m) \cup \{(\mathfrak{F}', \mathfrak{c}') \vdash \langle a \rangle_i \top$ for all $i \in ag(a)\}$
Consequently, property $P(\pi_{m+1})$ holds.

25. Suppose, rule (**TR 26**) has been applied to node ν_m.
Suppose, $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a), (\mathfrak{F}', \mathfrak{c}') \xrightarrow{a} (\mathfrak{F}', \mathfrak{c}'a), (\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \mathfrak{c}')$
for all $i \in ag(a)\}$.
$\Longrightarrow$ (* by definition 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
$\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ and $\ell_m((\mathfrak{F}', \mathfrak{c}')) \xrightarrow{a} \ell_m((\mathfrak{F}', \mathfrak{c}'a))$ and
$(\ell_m((\mathfrak{F}, \mathfrak{c})), \ell_m((\mathfrak{F}', \mathfrak{c}'))) \in R_i^K$ for all $i \in ag(a)$
$\Longrightarrow$ (* by definition 3.3.2 cond. 4 and E_i is totally ordered *)
$(\ell_m((\mathfrak{F}, \mathfrak{c}a)), \ell_m((\mathfrak{F}', \mathfrak{c}'a))) \in R_i^K$
$\Longrightarrow \ell_{m+1} := \ell_m$ is a satisfying $\mathcal{L}-$embedding for
$\Gamma(\nu_{m+1}) = \Gamma(\nu_m) \cup \{(\mathfrak{F}, \mathfrak{c}a)R_i(\mathfrak{F}', \mathfrak{c}'a)$ for all $i \in ag(a)\}$
Consequently, property $P(\pi_{m+1})$ holds.

□

We have shown that if the root $\Gamma_0^{\mathcal{T}}$ of regular tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ is $\mathcal{L}-$satisfiable with some $\mathcal{L}$-embedding $\ell : \mathfrak{label}(\Gamma_0^{\mathcal{T}}) \longrightarrow |\mathcal{M}|$ then there exists a path π in $\mathcal{T}(\Gamma_0^{\mathcal{T}})$, such that for each prefix π_m of π there exists a satisfying $\mathcal{L}$-embedding $\ell_m : \mathfrak{label}(\Gamma_m) \longrightarrow |\mathcal{M}|$ and it holds that for all $m \in N : \ell_m(\mathfrak{l}) = \ell_{m+1}(\mathfrak{l})$ for all $\mathfrak{l} \in \mathfrak{label}(\Gamma_m)$.

We will now show how to construct a satisfying $\mathcal{L}$-embedding ℓ_π for the set of formulae Γ_π. Note, that π may be an infinite path and thus Γ_π may also be infinite.

Let us now consider an infinite tableau path $\pi = \nu_0, \nu_1, \nu_2, \ldots$. Suppose, for this path holds that each node ν_m is $\mathcal{L}-$satisfiable with some $\mathcal{L}$-embedding ℓ_m, where $\mathcal{L}$-embedding ℓ_m is an extension of $\mathcal{L}$-embedding ℓ_{m-1} in the sense of lemma 6.1.2.

Lemma 6.1.3 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be a regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let $\ell_0 : \mathfrak{label}(\Gamma_0^{\mathcal{T}}) \longrightarrow |\mathcal{M}|$ be a satisfying $\mathcal{L}$-embedding for $\Gamma_0^{\mathcal{T}}$.*

There exists a tableau path π in $\mathcal{T}$, such that there exists a satisfying $\mathcal{L}$-embedding $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ for Γ_π.

Proof:
Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be a regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let $\ell_0 : \mathfrak{label}(\Gamma_0^{\mathcal{T}}) \longrightarrow |\mathcal{M}|$ be a satisfying $\mathcal{L}$-embedding for $\Gamma_0^{\mathcal{T}}$. According to lemma 6.1.2 there exists a path π in tableau $\mathcal{T}$, such that for all nodes ν_m on π there exists a satisfying $\mathcal{L}$-embedding $\ell_m : \mathfrak{label}(\Gamma(\nu_m)) \longrightarrow |\mathcal{M}|$ such that for all tableau labels $(\mathfrak{F}, \mathfrak{c}) \in \mathfrak{label}(\Gamma_m)$ it holds that $\ell_m((\mathfrak{F}, \mathfrak{c})) = \ell_{m+1}((\mathfrak{F}, \mathfrak{c}))$.

We now construct a mapping $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ in such a way that $\ell_\pi((\mathfrak{F}, \mathfrak{c})) := \ell_k((\mathfrak{F}, \mathfrak{c}))$ with $(\mathfrak{F}, \mathfrak{c}) \in \mathfrak{label}(\Gamma(\nu_k))$ and for all $n < k : (\mathfrak{F}, \mathfrak{c}) \notin \mathfrak{label}(\Gamma(\nu_n))$, in other words, we construct $\ell_\pi := \bigcup\limits_{k \in \mathbb{N}} \ell_k$ to be the union of all $\mathcal{L}$-embeddings ℓ_k.

Suppose, ℓ_π is not a satisfying $\mathcal{L}$-embedding for Γ_π. There may be three possible reasons:

1. There exists a structure formula $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a) \in \Gamma_\pi$ but not $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_\pi((\mathfrak{F}, \mathfrak{c}a))$.
   However, then there exists a smallest $m \in \mathbb{N}$ with $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}a) \in \Gamma(\nu_m)$ and not $\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}a))$ which is a contradiction to the assumption that ℓ_m is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m)$.

2. There exists a structure formula $(\mathfrak{F}, \mathfrak{c}) R_i (\mathfrak{F}', \mathfrak{c}') \in \Gamma_\pi$ but $(\ell_\pi((\mathfrak{F}, \mathfrak{c})), \ell_\pi((\mathfrak{F}', \mathfrak{c}'))) \notin R_i^K$.
   However, then there exists a smallest m with $(\mathfrak{F}, \mathfrak{c}) R_i (\mathfrak{F}', \mathfrak{c}') \in \Gamma(\nu_m)$ and $(\ell_m((\mathfrak{F}, \mathfrak{c})), \ell_m((\mathfrak{F}', \mathfrak{c}'))) \notin R_i^K$ which is a contradiction to the assumption that ℓ_m is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m)$.

3. There exists a labelled formula $(\mathfrak{F}, \mathfrak{c}) \vdash \phi \in \Gamma_\pi$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \not\models_{\mathcal{M}} \phi$.
   However, then there exists a smallest m with $(\mathfrak{F}, \mathfrak{c}) \vdash \phi \in \Gamma(\nu_m)$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \not\models_{\mathcal{M}} \phi$ which is a contradiction to the assumption that ℓ_m is a satisfying $\mathcal{L}$-embedding for $\Gamma(\nu_m)$.

From this we can conclude that ℓ_π is a satisfying $\mathcal{L}$-embedding for Γ_π and thus ℓ_π is a satisfying $\mathcal{L}$-embedding for all $\Gamma(\nu_m)$ with $\nu_m \in \pi$.

□

As we have stated before, there are several possible reasons for infinite paths in a tableau. One is the existence of so-called unfair tableau paths. Remember, we call a tableau-path $\pi = \nu_0, \nu_1, \nu_2, \ldots$ unfair, if Γ_π contains a tableau-formula $(\mathfrak{F}, \mathfrak{c}) \vdash \langle\cdot\rangle_i\phi$ but does not contain a chain $\chi = (\mathfrak{F}, \mathfrak{c}) \xrightarrow{o_0} (\mathfrak{F}, \mathfrak{c}_1) \xrightarrow{o_1} \ldots \xrightarrow{o_n} (\mathfrak{F}, \mathfrak{c}_{n+1})$ with $o_n \in \mathcal{O}_i$.

We show that we may ignore unfair tableau paths: Whenever there is an unfair path $\pi = \nu_0, \nu_1, \nu_2 \ldots$ such that Γ_π is $\mathcal{L}-$satisfiable, there also exists a fair path $\pi' = \nu_0, \nu_1', \nu_2', \ldots$ such that $\Gamma_{\pi'}$ is $\mathcal{L}-$satisfiable.

Lemma 6.1.4 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be a regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let $\ell_0 : \mathfrak{label}(\Gamma_0^{\mathcal{T}}) \longrightarrow |\mathcal{M}|$ be a satisfying $\mathcal{L}$-embedding for $\Gamma_0^{\mathcal{T}}$.*

The tableau $\mathcal{T}$ contains a fair tableau path π, such that there exists a satisfying $\mathcal{L}$-embedding $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$.

Each maximal chain χ of tableau labels of a tableau path π can be seen as the counterpart of an interleaving of a partially ordered run F of the model $\mathcal{M}$. We call an interleaving of a partially ordered run *fair*, if for every agent i and for every situation (F, c) it holds that starting from situation (F, c), if i performs another operation in the partial order view, then i performs the next operation after a finite number of steps in the interleaved view.

We prove lemma 6.1.4 by step by step identifying a fair tableau path π such that each maximal chain of Γ_π corresponds to a fair interleaving of a run of model $\mathcal{M}$. While identifying π, we at the same time construct a satisfying $\mathcal{L}$-embedding $\ell_\pi : \mathfrak{label} \longrightarrow |\mathcal{M}|$ for Γ_π.

Before we give the proof of lemma 6.1.4, we introduce fair interleavings formally:

Definition 6.1.5 (interleaving) *Let $\mathcal{M}$ be a model and let $(F, c_0) \in |\mathcal{M}|$ be a situation of $\mathcal{M}$ with $F = (E, \leq, \lambda)$.*

An interleaving $\overrightarrow{(F, c_0)} = ((F, c_0), (F, c_1), \ldots)$ starting from situation (F, c_0), is a sequence of situations, such that

1. *for all situations $(F, c) \in \overrightarrow{(F, c_0)}$, with $(F, c) \neq (F, c_0)$ it holds that $c_0 \subset c$, i.e. (F, c_0) is the least situation of $\overrightarrow{(F, c_0)}$;*

2. *each situation* $(F, c_i) \in \overrightarrow{(F, c_0)}$ *either has exactly one direct successor* $(F, c_{i+1}) \in \overrightarrow{(F, c_0)}$*, such that there exists an event* $e \in E$ *with* $c_i \cup \{e\} = c_{i+1}$ *or* (F, c_i) *is he last element of the sequence.*

We call an infinite interleaving $\overrightarrow{(F, c_0)}$ fair *iff for all agents* $i \in Ag$ *and for all situations* $(F, c) \in \overrightarrow{(F, c_0)}$ *the following holds:*

If $E_i \setminus c \neq \emptyset$*, then there exists a finite subsequence* $((F, c), (F, c'), \ldots, (F, c^*))$ *of* $\overrightarrow{(F, c_0)}$ *such that* $(c^* \setminus c) \cap E_i \neq \emptyset$*.*

Informally, we call an interleaving fair, if for every agent i and for every situation (F, c) it holds that starting from situation (F, c), if i performs another operation in the partial order view, then i performs the next operation after a finite number of steps in the interleaved view.

Proof of lemma 6.1.4:
Suppose, $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ is a regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and $\ell : \mathfrak{label}(\Gamma_0^{\mathcal{T}}) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma_0^{\mathcal{T}}$.

In the following we will stepwise identify a fair tableau path π with a satisfying $\mathcal{L}$-embedding $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$. We further fix a set I of fair interleavings , such that for every maximal chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{a_0} (\mathfrak{F}, \mathfrak{c}_1) \xrightarrow{a_1} \ldots$ there exists exactly one fair interleaving $\overrightarrow{(F, c_0)} \in I$ with $\ell_\pi((\mathfrak{F}, \mathfrak{c}_o)) = (F, c_0)$.

The following invariant will hold for each prefix $\pi_m = v_0 v_1 v_2, \ldots v_m$ of π:

Invariant:

1. ℓ_m is a satisfying $\mathcal{L}$-embedding for Γ_m and
2. each maximal chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{a_0} (\mathfrak{F}, \mathfrak{c}_1) \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} (\mathfrak{F}, \mathfrak{c}_n)$ of Γ_m corresponds to a prefix of a fair interleaving $\overrightarrow{(F, c_0)}$ such that
   a) $\ell_m((\mathfrak{F}, \mathfrak{c}_o)) = (F, c_0)$ and
   b) if $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{a} (\mathfrak{F}, \mathfrak{c}')$ in χ, then $\ell_m((\mathfrak{F}, \mathfrak{c})) \in \overrightarrow{(F, c_0)}$ and $\ell_m((\mathfrak{F}, \mathfrak{c}')) \in \overrightarrow{(F, c_0)}$ and $\ell_m((\mathfrak{F}, \mathfrak{c})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}'))$.

Induction begin:
Let $\pi_0 := v_0$ and for $\ell_0((\mathfrak{F}, \mathfrak{c}_o)) = (F, c_)$ we fix a fair interleaving $\overrightarrow{(F, c_0)}$ and define $I_0 := \{\overrightarrow{(F, c_0)}\}$.

According to the assumption, $\ell_0 : \mathfrak{label}(\Gamma_0^{\mathcal{T}}) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for $\Gamma_0^{\mathcal{T}}$. Further, $\Gamma_0^{\mathcal{T}}$ has exactly one maximal chain, namely $\chi = (\mathfrak{F}, \mathfrak{c}_o)$. Obviously,

$\ell_0((\mathfrak{F}, \mathfrak{c}_o)) = (F, c_0)$ is a prefix of the fair interleaving $\overrightarrow{(F, c_0)}$. Consequently, the invariant holds for π_0 and I_0.

Induction hypothesis:
Suppose, we have already identified a prefix π_m of π, an $\mathcal{L}$-embedding ℓ_m : $\mathfrak{label}(\Gamma_m) \longrightarrow |\mathcal{M}|$ and a set I_m of fair interleavings, such that the invariant holds for π_m and I_m.

Induction step:
We construct π_{m+1} from π_m, ℓ_{m+1} from ℓ_m and I_{m+1} from I_m such that the invariant holds for π_{m+1},ℓ_{m+1} and I_{m+1}.

We divide the tableau rules into five categories:

1. Suppose, one of the following tableau rules is applied in node ν_m: (**TR 1**), (**TR 2**), (**TR 17**).
   Since ℓ_m is a satisfying $\mathcal{L}$-embedding, none of these rules can have been applied.

2. Suppose, one of the following tableau rules is applied in node ν_m:
   (**TR 3**), (**TR 4**), (**TR 6**), (**TR 10**), (**TR 11**), (**TR 12**), (**TR 13**), (**TR 15**), (**TR 16**), (**TR 18**), (**TR 19**), (**TR 20**), (**TR 21**), (**TR 22**), (**TR 23**), (**TR 24**), (**TR 25**), (**TR 26**).
   Each of these rules has only one denominator such that there is no choice for the extension of π_m to π_{m+1}. According to the proof of lemma 6.1.2, we define $\ell_{m+1} := \ell_m$. Since $\Gamma_m \simeq \Gamma_{m+1}$, we also define $I_{m+1} := I_m$. Then the invariant holds for π_{m+1}.

3. Suppose, rule (**TR 5**) is applied in node ν_m.

   As none of the denominators of this rule adds new labels to the tableau set, we have that $\Gamma_m \simeq \Gamma_{m+1}$. We keep the $\mathcal{L}$-embedding and define $\ell_{m+1} := \ell_m$. According to induction step 5 on page 108 in the proof of lemma 6.1.2, ℓ_{m+1} is a satisfying $\mathcal{L}$-embedding for Γ_{m+1}. Thus, the first item of the invariant holds for π_{m+1}. Since $\Gamma_m \simeq \Gamma_{m+1}$, we do not need to consider any new interleavings and thus we define $I_{m+1} := I_m$. Then, obviously, the second item of the invariant is also fulfilled, such that the invariant holds for π_{m+1}, ℓ_{m+1} and I_{m+1}.

4. Suppose, rule (**TR 7**) or rule (**TR 8**) are applied in node ν_m.

   Again, as none of the denominators of rule (**TR 7**) or rule (**TR 8**), respectively, adds a new tableau label to the tableau set, we hat that $\Gamma_m \simeq \Gamma_{m+1}$. We define $\ell_{m+1} := \ell_m$. According to induction step 7 on page 109 and induction step 8 on page 109 in the proof of lemma 6.1.2 respectively, ℓ_{m+1} is a satisfying $\mathcal{L}$-embedding for either the left or the right denominator of the rule. We choose node ν_{n+1} accordingly, so that ℓ_{m+1} is a satisfying $\mathcal{L}$-embedding for

Γ_{m+1}. Thus, the first item of the invariant holds for π_{m+1}. Since we have that $\Gamma_m \simeq \Gamma_{m+1}$ we do not need to consider any new interleavings and thus we define $I_{m+1} := I_m$. Obviously, the second item of the invariant is also fulfilled, such that the invariant holds for π_{m+1}, ℓ_{m+1} and I_{m+1}.

5. Suppose, rule (**TR 9**) is applied in node ν_m.

   This rule has only one denominator, so there is only one possible node ν_{m+1} for π_{m+1} to extend π_m. We define $\ell_{m+1}(\mathfrak{l}) := \begin{cases} \ell_m(\mathfrak{l}) & \text{for } \mathfrak{l} \in \mathfrak{label}(\Gamma_m) \\ (F', c') & \text{for } \mathfrak{l} \in (\mathfrak{F}', \mathfrak{c}') \end{cases}$
   According to induction step 9 on page 110 in the proof of lemma 6.1.2, ℓ_{m+1} is a satisfying $\mathcal{L}$-embedding for Γ_{m+1}. Thus, the first item of the invariant holds for π_{m+1}.

   The tableau label $(\mathfrak{F}', \mathfrak{c}')$ is a new label in Γ_{m+1}. Since rule (**TR 9**) does not add a new structure formula of the type $\mathfrak{l} \xrightarrow{o} \mathfrak{l}'$ to Γ_m for any tableau labels $\mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_{m+1})$, we have a new maximal chain $\chi = (\mathfrak{F}', \mathfrak{c}')$ consisting of only one tableau label.

   Let $\ell_{m+1}((\mathfrak{F}', \mathfrak{c}')) = (F', c')$, then we fix a fair interleaving $\overrightarrow{(F', c')}$ and define

   $$I_{m+1} := I_m \cup \{\overrightarrow{(F', c')}\}.$$

   For all maximal chains $\chi' \neq \chi$ in Γ_{m+1} the second item of the invariant is trivially fulfilled. For the newly added chain $\chi = (\mathfrak{F}', \mathfrak{c}')$ we have fixed the interleaving $\overrightarrow{(F', c')}$ in such a way that the second item of the invariant is trivially fulfilled.

   Thus, the invariant holds for π_{m+1}, ℓ_{m+1} and I_{m+1}.

   Therefore, the invariant is fulfilled for π_{m+1} and ℓ_{m+1}.

6. Suppose, rule (**TR 14**) is applied in node ν_m.

   This is the crucial rule for identifying the fair tableau path π. This rule has a choice of denominators (one denominator for each operation $o \in \mathcal{O}$) as well as it introduces a new tableau label and extends a maximal chain χ in each denominator.

   We choose the node ν_{m+1} which extends path π_m according to the already fixed set of interleavings in the following way.

   Let $\Gamma(\nu_m) = \Gamma_{\mathcal{T}} \cup \{(\mathfrak{F}, \mathfrak{c}) \vdash \langle\cdot\rangle_i \phi, (\mathfrak{F}, \mathfrak{c})\}$ with $(\mathfrak{F}, \mathfrak{c})$ unmarked and either $\langle\cdot\rangle_i \phi = \langle a \rangle_i \phi$ for some $i \in Ag, a \in \mathcal{O}_i$ and some $\phi \in \mathcal{L}$ or $\langle\cdot\rangle_i \phi = \langle \mathcal{O} \rangle_i \phi$ for some $i \in Ag$ and some $\phi \in \mathcal{L}$.

$\Longrightarrow$ (* by def. 5.3.2 of a satisfying $\mathcal{L}$-embedding *)
there exists an operation $o \in \mathcal{O}_i$ with $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \langle o \rangle_i \phi$

$\Longrightarrow$ (* by lemma 5.5.4 and because $(\mathfrak{F}, \mathfrak{c})$ is unmarked *)
there exists an operation $o \in \mathcal{O}_i$ with $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \langle o \rangle_i \phi$ and there exists exactly one maximal chain of the form $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{a_0} \ldots \xrightarrow{a_{n-1}} (\mathfrak{F}, \mathfrak{c})$ in $\Gamma(\nu_m)$

$\Longrightarrow$ (* by induction hypothesis *)
there exists a fair interleaving $\overrightarrow{(F, c_0)} \in I_m$ that corresponds to χ in the following way: $\ell_m((\mathfrak{F}, \mathfrak{c}_o)) = (F, c_0)$ and for all $(\mathfrak{F}, \mathfrak{c}_\mathfrak{x}) \xrightarrow{a_x} (\mathfrak{F}, \mathfrak{c}_{\mathfrak{x}+1})$ in χ with $x \leq n-1$ it holds that $\ell_m((\mathfrak{F}, \mathfrak{c}_\mathfrak{x})) \in \overrightarrow{(F, c_0)}$ and $\ell_m((\mathfrak{F}, \mathfrak{c}_{\mathfrak{x}+1})) \in \overrightarrow{(F, c_0)}$ and $\ell_m((\mathfrak{F}, \mathfrak{c}_\mathfrak{x})) \xrightarrow{a} \ell_m((\mathfrak{F}, \mathfrak{c}_{\mathfrak{x}+1}))$
and there exists an operation $o \in \mathcal{O}_i$ with $\ell_m((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \langle o \rangle_i \phi$

Let $\ell_m((\mathfrak{F}, \mathfrak{c})) = (F, c)$. There exists a situation $(F, c') \in \overrightarrow{(F, c_0)}$ such that $(F, c) \xrightarrow{o'} (F, c')$ for some $o' \in \mathcal{O}$. (Note, that not necessarily $o = o'$. It might also be that $o' \in \mathcal{O}_j$ for some $j \notin ag(o)$.)

We choose the extension of π_m according to the fair interleaving $\overrightarrow{(F, c_0)}$ in such a way that $(\mathfrak{F}, \mathfrak{c}) \xrightarrow{o'} (\mathfrak{F}, \mathfrak{c}') \in \Gamma(\nu_{m+1})$ for some new tableau label $(\mathfrak{F}, \mathfrak{c}')$. (This extension is possible because rule (**TR 14**) generates a denominator for each operation $o \in \mathcal{O}$.

Next, we modify the $\mathcal{L}$-embedding as we have done in the proof of lemma 6.1.2:

$$\ell_{m+1}(\mathfrak{l}) = \begin{cases} \ell_m(\mathfrak{l}) & \text{for } \mathfrak{l} \neq (\mathfrak{F}, \mathfrak{c}') \\ (F, c') & \text{for } \mathfrak{l} = (\mathfrak{F}, \mathfrak{c}') \end{cases}$$

Because of the definition of ℓ_{m+1} according to the fair interleaving, ℓ_{m+1} is a satisfying $\mathcal{L}$-embedding for $\Gamma_{m+1} = \Gamma_m \cup \{(\mathfrak{F}, \mathfrak{c}) \xrightarrow{o'} (\mathfrak{F}, \mathfrak{c}')\}$.

Since we have chosen the extension of π_m according to the fair interleaving $\overrightarrow{\ell_m((\mathfrak{F}, \mathfrak{c}_o))}$, the invariant is fulfilled for π_{m+1}.

Up to now, we have identified a path $\pi = \nu_0, \nu_1, \nu_2, \ldots$. According to the proof of lemma 6.1.3, $\ell_\pi := \bigcup_{m \in \mathbf{N}} \ell_m$ is a satisfying $\mathcal{L}$-embedding for Γ_π. We define the set of fair interleavings $I_\pi := \bigcup_{m \in \mathbf{N}} I_m$.

Now suppose, $\pi = \nu_0, \nu_1, \nu_2, \ldots$ is not fair.

By definition 5.5.8 of fairness of a tableau path, there exists a tableau label $(\mathfrak{F}, \mathfrak{c}) \in \mathfrak{label}(\Gamma_\pi)$, such that $(\mathfrak{F}, \mathfrak{c}) \vdash \langle a \rangle_i \phi \in \Gamma_\pi$ or $(\mathfrak{F}, \mathfrak{c}) \vdash \langle \mathcal{O} \rangle_i \phi \in \Gamma_\pi$ and for the maximal chain $\chi = \mathfrak{l}_o \xrightarrow{a_0} \mathfrak{l}_1 \xrightarrow{a_1} \mathfrak{l}_2 \ldots \xrightarrow{a_k} (\mathfrak{F}, \mathfrak{c}) \xrightarrow{a_{k+1}} \ldots$ it holds that for $k' > k$, $a_{k'} \notin \mathcal{O}_i$.

Suppose, $\ell_m((\mathfrak{F},\mathfrak{c})) = (F, c)$. Since ℓ_π is a satisfying $\mathcal{L}$-embedding for Γ_π, there exists an operation $o \in \mathcal{O}_i$, such that $(F, c) \models_\mathcal{M} \langle o \rangle_i \phi$.

By construction of the set of fair interleavings I_π, there exists a fair interleaving $\overrightarrow{(F, c_o)} \in I_\pi$ with $(F, c_0) = ((F, c_0), (F, c_1), (F, c_2), \ldots)$ and with $\ell_\pi(\mathfrak{l}_o) = (F, c_0)$ and for all $\mathfrak{l}_m \xrightarrow{a_m} \mathfrak{l}_{m+1}$ it holds that $\ell_\pi(\mathfrak{l}_m) \in \overrightarrow{(F, c'_0)}$ and $\ell_\pi(\mathfrak{l}_{m+1}) \in \overrightarrow{(F, c'_0)}$ and $\ell_\pi(\mathfrak{l}_m) \xrightarrow{a_m} \ell_\pi(\mathfrak{l}_{m+1})$.

Thus, there exists a $k \in \mathbb{N}$ such that $\ell_\pi((\mathfrak{F},\mathfrak{c})) = (F, c_k)$ and for all $k' > k$ it holds that if $(F, c_{k'}), (F, c_{k'+1}) \in \overrightarrow{(F, c_o)}$ and $(F, c_{k'}) \xrightarrow{a_{ak'}} (F, c_{k'+1})$ then $a_{k'} \notin \mathcal{O}_i$.

Since $(F, c_k) \models_\mathcal{M} \langle o \rangle_i \phi$ with $o \in \mathcal{O}_i$ this is a contradiction to the definition of a fair interleaving.

Consequently, the identified path π must be a fair path.

We have shown that if the root $\Gamma_0^\mathcal{T}$ of a regular tableau $\mathcal{T}(\Gamma_0^\mathcal{T})$ is $\mathcal{L}-$satisfiable, then there exists a fair tableau path π in $\mathcal{T}$, such that Γ_π is $\mathcal{L}-$ satisfiable. This means that we do not have to consider unfair paths any further but can simply reject unfair paths in a tableau: If we find an unfair tableau path π such that Γ_π is $\mathcal{L}-$satisfiable, then there also exists a fair tableau path π', such that $\Gamma_{\pi'}$ is $\mathcal{L}-$ satisfiable.

Next we have a closer look at paths that contain $\mathcal{U}-$ traces. We will show that there does not exist a satisfying $\mathcal{L}$-embedding for any path π that contains an $\mathcal{U}$-trace.

Consider an until formula $\phi \mathcal{U}_i \psi$ that is true in a situation (F, c) of a model $\mathcal{M}$. The semantics of the until-formula requires that there exists a situation (F, c') with $\downarrow^i (F, c) \subseteq \downarrow^i (F, c')$ such that $\downarrow^i (F, c')$ "resolves" the until formula, i.e. makes ψ true ($\downarrow^i (F, c') \models_\mathcal{M} \psi$).

Now suppose, π is a path of $\mathcal{T}$ and Γ_π contains a $\mathcal{U}$-trace, i.e. it contains a tableau formula $(\mathfrak{F},\mathfrak{c}) \vdash \phi\, \mathcal{U}_i \psi$ and for all $(\mathfrak{F},\mathfrak{c}')$ that occur later in the same maximal chain as $(\mathfrak{F},\mathfrak{c})$, Γ_π contains a formula $(\mathfrak{F},\mathfrak{c}') \vdash \neg\psi$. A chain corresponds to an interleaving of a run in a model, so this means that in such an interleaving each situation into which one of the labels $(\mathfrak{F},\mathfrak{c}')$ is embedded satisfies the formula $\neg\psi$. This, however, contradicts the requirement that ψ must eventually be satisfied.

Lemma 6.1.6 *Let $\mathcal{T}$ be a tableau and let π be a fair infinite path of $\mathcal{T}$, such that π contains an $\mathcal{U}$-trace. Then Γ_π is not $\mathcal{L}-$ satisfiable.*

Proof:

Suppose, π is a fair tableau path that contains an $\mathcal{U}$-trace and Γ_π is $\mathcal{L}-$satisfiable.

By definition 5.5.7 of an $\mathcal{U}$-trace this supposition implies

Γ_π contains a (not necessarily maximal) infinite chain

$$\chi = \mathfrak{l}_k \xrightarrow{o_k} \mathfrak{l}_{k+1} \xrightarrow{o_{k+1}} \ldots \text{ for some } k \in \mathbb{N}, \tag{1}$$

$$\text{such that } \mathfrak{l}_k \vdash \phi\ \mathcal{U}_i\psi \in \Gamma_\pi \tag{2}$$

$$\text{and for all } \mathfrak{l}_m \in \chi \text{ with } m \geq k \text{ and } \mathfrak{l}_m \vdash \phi\ \mathcal{U}_i\psi \in \Gamma_\pi \tag{3}$$

$$\text{there exists a subchain } \chi' = \mathfrak{l}_m \xrightarrow{o_m} \ldots \xrightarrow{o_n} \mathfrak{l}_{n+1} \text{ of } \chi, \text{ such that} \tag{4}$$

$$\mathfrak{l}_m \vdash \neg\psi \in \Gamma_\pi, \tag{5}$$

$$\text{for all } o_x, \text{ with } m \leq x < n \text{ we have that } o_x \notin \mathcal{O}_i, \text{ and} \tag{6}$$

$$\mathfrak{l}_{n+1} \vdash \phi\mathcal{U}_i\psi \in \Gamma_\pi \tag{7}$$

Remember, the assumption was that Γ_π is $\mathcal{L}$—satisfiable, i.e. there a exists a satisfying $\mathcal{L}$-embedding $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$. By definition 5.3.2 of a satisfying $\mathcal{L}$-embedding and by the lines above we can conclude that the following holds:

for all $x \geq k$ it holds that $\ell_\pi(\mathfrak{l}_x) \xrightarrow{o_x} \ell_\pi(\mathfrak{l}_{x+1})$ (* cf. line (1) *)
and $\ell_\pi(\mathfrak{l}_k) \models_\mathcal{M} \phi\mathcal{U}_i\psi$. (* cf. line (2) *)

Further, from lines (3) to (7) it follows that for all tableau labels $\mathfrak{l}_m \in \chi$ with $m \geq k$ and $\mathfrak{l}_m \vdash \phi\ \mathcal{U}_i\psi \in \Gamma_\pi$ there exists a subchain
$\chi' = \mathfrak{l}_m \xrightarrow{o_m} \ldots \xrightarrow{o_n} \mathfrak{l}_{n+1}$ of χ, such that

$\ell_\pi(\mathfrak{l}_m) \xrightarrow{o_m} \ell_\pi(\mathfrak{l}_{m+1}) \xrightarrow{o_{m+2}} \ldots \xrightarrow{o_n} \ell_\pi(\mathfrak{l}_{n+1})$,
$\ell_\pi(\mathfrak{l}_m) \models_\mathcal{M} \phi\ \mathcal{U}_i\psi \wedge \neg\psi$, (* cf. lines (3), (5) *)
$\ell_\pi(\mathfrak{l}_{n+1}) \models_\mathcal{M} \phi\ \mathcal{U}_i\psi$ (* cf. line (6) *)
and $\ell_\pi(\mathfrak{l}_m) \equiv_i \ell_\pi(\mathfrak{l}_{m+1}) \equiv_i \ldots \equiv_i \ell_\pi(\mathfrak{l}_n)$ (* cf. line (7) *)

By lemma 4.3.2 this means that $\ell_\pi(\mathfrak{l}_k) \models_\mathcal{M} \phi\ \mathcal{U}_i\psi$ and for all tableau labels $\mathfrak{l}_m \in \chi$ with $\mathfrak{l}_m \vdash \phi\ \mathcal{U}_i\psi \in \Gamma_\pi$ there exists a subchain $\chi' = \mathfrak{l}_m \xrightarrow{o_m} \ldots \xrightarrow{o_n} \mathfrak{l}_{n+1}$ of χ,
such that

$$\text{for all } m \leq x \leq n \text{ it holds that } \ell_\pi(\mathfrak{l}_x) \models_\mathcal{M} \phi\ \mathcal{U}_i\psi \wedge \neg\psi, \tag{8}$$

$$\text{and } \ell_\pi(\mathfrak{l}_{n+1}) \models_\mathcal{M} \phi\ \mathcal{U}_i\psi \wedge \neg\psi. \tag{9}$$

This implies that for all tableau labels $\mathfrak{l} \in \chi$ it holds that

$$\ell_\pi(\mathfrak{l}) \models_\mathcal{M} \phi\ \mathcal{U}_i\psi \wedge \neg\psi. \tag{10}$$

Again by lemma 4.3.2 we have that for all situations $(F, c) \in |\mathcal{M}|$, such that there exists a tableau label $\mathfrak{l} \in \chi$ with $\ell_\pi(\mathfrak{l}) \equiv_i (F, c)$, it holds that $(F, c) \models_\mathcal{M} \phi\ \mathcal{U}_i\psi \wedge \neg\psi$.

According to definition 3.2.2 the events local to each agent are totally ordered. Thus, for $\ell_\pi(\mathfrak{l}_o) = (\mathsf{F}, \mathsf{c}_o)$ there does not exist a situation $(\mathsf{F}, \mathsf{d}) \in |\mathcal{M}|$ with $\mathsf{c}_0 \subseteq \mathsf{d}$ and $(\mathsf{F}, \mathsf{d}) \not\models_{\mathcal{M}} \phi\ \mathcal{U}_i\psi \wedge \neg\psi$.

This is a contradiction to the semantics of $\phi\ \mathcal{U}_i\psi$ which demands that at some situation $(\mathsf{F}, \mathsf{c}) \supseteq \ell_\pi((\mathfrak{F}, \mathfrak{c}_o))$ the formula ψ must hold.

Thus, if π is a fair tableau path and there exists a satisfying $\mathcal{L}$-embedding ℓ_π for Γ_π, then π does not contain an $\mathcal{U}$-trace.

□

With these preliminaries we can now prove the soundness of the tableau system as stated in theorem 6.1.1.

Let us recall theorem 6.1.1:
The given tableau system is sound: If a regular tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ of this system rejects a finite set of formulae $\Gamma_0^{\mathcal{T}}$, then $\Gamma_0^{\mathcal{T}}$ is not $\mathcal{L}-$satisfiable.

Proof of theorem 6.1.1:

Suppose,

$$\Gamma_0^{\mathcal{T}} \text{ is a finite } \mathcal{L}-\text{satisfiable set of tableau formulae and} \tag{11}$$

$$\text{there exists a rejecting regular tableau } \mathcal{T}(\Gamma_0^{\mathcal{T}}) \tag{12}$$

By definition 5.3.2 on page 55 of a satisfying $\mathcal{L}$-embedding there exists a model $\mathcal{M} = (\mathcal{A}, \mathcal{R}^{\mathsf{K}}, \mathcal{I})$ and a satisfying $\mathcal{L}$-embedding $\ell : \mathfrak{label}(\Gamma_0^{\mathcal{T}}) \longrightarrow |\mathcal{M}|$ for $\Gamma_0^{\mathcal{T}}$ from the set of tableau labels of $\Gamma_0^{\mathcal{T}}$ (i.e. $\{\mathfrak{l}_o\}$) into the set of situations of $\mathcal{M}$.

According to lemma 6.1.3 on page 118, there exists a tableau path π in $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ such that there exists a satisfying $\mathcal{L}$-embedding $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ for Γ_π.

Now we distinguish between two cases:

1. π is a finite path.
   $\Longrightarrow$ (* by assumption 12 that $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ is a rejecting tableau and by the rejection conditions in definition 5.5.9 *)
   The leaf of π is labelled with an axiom.
   $\Longrightarrow$ Path π either contains a tableau formula $(\mathfrak{F}, \mathfrak{c}) \vdash \bot$ or two tableau formulae $(\mathfrak{F}, \mathfrak{c}) \vdash \phi$ and $(\mathfrak{F}, \mathfrak{c}) \vdash \neg\phi$.
   $\Longrightarrow$ There does not exist a satisfying $\mathcal{L}$-embedding for Γ_π.
   $\Longrightarrow$ Path π is not a finite.
2. π is infinite.

As $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ is a rejecting tableau, we again distinguish between two cases:

a) π is fair and contains an $\mathcal{U}$-trace.
$\implies$ (* with lemma 6.1.6 and proposition 6.1.6 *)
Γ_π is not $\mathcal{L}-$satisfiable.
$\implies$ this is a contradiction to the conclusion drawn above from lemma 6.1.3
$\implies$ π is either unfair or does not contain an $\mathcal{U}$-trace.

b) π is unfair.
$\implies$ (* by lemma 6.1.4 on page 119 *)
There exists a *fair* tableau path π', such that $\Gamma_{\pi'}$ is $\mathcal{L}-$satisfiable.
$\implies$ (* according to the assumption (12) that $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ is a rejecting tableau *)
Path π' must either be finite finite path the leaf of which is labelled with an axiom or it must be an infinite and fair path that contains an $\mathcal{U}$-trace.
$\implies$ This is a contradiction to the proof steps above.
$\implies$ π is not unfair.

These three cases lead to the conclusion that path π is neither a finite path the leaf of which is labelled with an axiom, nor is it an unfair path nor does it contain an $\mathcal{U}$-trace.

Thus, according to definition 5.5.9, π is not a rejected path and consequently, $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ is not a rejecting tableau.

This is a contradiction to assumption (12).

Thus, we have proven theorem 6.1.1:
If $\Gamma_0^{\mathcal{T}}$ is $\mathcal{L}-$satisfiable, then there does not exist a rejecting regular tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ for $\Gamma_0^{\mathcal{T}}$.

$\square$

6.2 Completeness of the Tableau System

We call the tableau method *complete* if there exists a rejected tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ for each tableau set $\Gamma_0^{\mathcal{T}}$ that is not $\mathcal{L}-$satisfiable.

As already mentioned in the introduction to this chapter on page 105, the tableau method is not complete for the full language $\mathcal{L}$ without the belief operator $\mathcal{B}$.

The following example is a counterexample for completeness.

Let $Ag := \{1, 2\}, \mathcal{O}_1 := \{a\}, \mathcal{O}_2 := \{b\}$ and $\mathcal{P}_1 = \emptyset, \mathcal{P}_2 := \{p\}$. Now consider tableau set $\Gamma_0^{\mathcal{T}} := \{\iota_o \vdash K_1 p, \iota_o \vdash \langle a\rangle_1 \top, \iota_o \vdash \langle b\rangle_2(\neg p)\}$. $\Gamma_0^{\mathcal{T}}$ is not $\mathcal{L}-$satisfiable.

However, each tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ will contain an accepted tableau path like the following:

$\iota_o \vdash K_1 p, \iota_o \vdash \langle a\rangle_1 \top, \iota_o \vdash \langle b\rangle_2(\neg p), \iota_o$

(TR 6)

$\iota_o \vdash K_1 p, \iota_o \vdash p, \iota_o \vdash \langle a\rangle_1 \top, \iota_o \vdash \langle b\rangle_2(\neg p), \iota_o$

(TR 14)

$\iota_o \vdash K_1 p, \iota_o \vdash p, \iota_o \vdash \langle a\rangle_1 \top, \iota_o \xrightarrow{a} \iota_a, \iota_o \vdash \langle b\rangle_2(\neg p), \iota_o, \iota_a$

(TR 15), **(TR 18)**

$\iota_o \vdash K_1 p, \iota_o \vdash p, \iota_a \vdash \top, \iota_o \xrightarrow{a} \iota_a, \iota_a \vdash \langle b\rangle_2(\neg p), \iota_o, \iota_a$

(TR 3)

$\iota_o \vdash K_1 p, \iota_o \vdash p, \iota_o \xrightarrow{a} \iota_a, \iota_a \vdash \langle b\rangle_2(\neg p), \iota_o, \iota_a$

(TR 24)

$\iota_o \vdash K_1 p, \iota_o \vdash p, \iota_o \xrightarrow{a} \iota_a, \iota_o R_2 \iota_a, \iota_a \vdash \langle b\rangle_2(\neg p), \iota_o, \iota_a$

(TR 14)

$\iota_o \vdash K_1 p, \iota_o \vdash p, \iota_o \xrightarrow{a} \iota_a, \iota_a \xrightarrow{b} \iota_{ab}, \iota_o R_2 \iota_a, \iota_a \vdash \langle b\rangle_2(\neg p), \iota_o, \iota_a, \iota_{ab}$

(TR 15)

$\iota_o \vdash K_1 p, \iota_o \vdash p, \iota_o \xrightarrow{a} \iota_a, \iota_a \xrightarrow{b} \iota_{ab}, \iota_o R_2 \iota_a, \iota_{ab} \vdash \neg p, \iota_o, \iota_a, \iota_{ab}$

(TR 12)

$\iota_o \vdash K_1 p, \iota_o \vdash p, \iota_o \xrightarrow{a} \iota_a, \iota_a \xrightarrow{b} \iota_{ab}, \iota_o R_2 \iota_a, \iota_a \vdash p, \iota_{ab} \vdash \neg p, \iota_o, \iota_a, \iota_{ab}$

(TR 24)

$\iota_o \vdash K_1 p, \iota_o \vdash p, \iota_o \xrightarrow{a} \iota_a, \iota_a \xrightarrow{b} \iota_{ab}, \iota_a R_1 \iota_{ab}, \iota_o R_2 \iota_a, \iota_{ab} \vdash \neg p, \iota_o, \iota_a, \iota_{ab}$

(TR 10)

$\iota_o \vdash K_1 p, \iota_o \vdash p, \iota_o \xrightarrow{a} \iota_a, \iota_a \xrightarrow{b} \iota_{ab}, \iota_a R_1 \iota_{ab}, \iota_o R_2 \iota_a, \iota_{ab} R_1 \iota_a, \iota_a R_2 \iota_o, \iota_{ab} \vdash \neg p, \iota_o, \iota_a, \iota_{ab}$

No further tableau rules can be applied to the last line of above tableau path. On first sight it seems as if rule (**TR 12**) would be applicable to formulae $l_{ab}R_1 l_a$ and $l_{ab} \vdash \neg p$. However, the side condition of rule (**TR 12**) requiring p to be element of $\mathcal{P}_1$ forbids the application of the rule. Since the path is finite and does not end with an axiom, it is accepted. Hence, the tableau containing such a path is accepted as well. However, the tableau set $\Gamma_0^{\mathcal{T}}$ is not $\mathcal{L}-$satisfiable.

In the formula $K_1 p$, the proposition p is not local to agent 1 who is supposed to know p. Agent 1 cannot notice, when agent 2 changes the value of p. Figure 6.1 illustrates the problem. The situation (F, cb), shown by a red line in the figure, does not 'occur' in the tableau in the sense that if we embed the tableau labels into a model, then the considered tableau path does not contain a label that is embedded into situation (F, cb). This fact in itself is not a problem. In lemma 4.3.2 we have shown that a formulae of type A is satisfied in a situation (F, c) if and only if it is satisfied in all situations (F, c') that are i-equivalent to (F, c) for *all* $i \in A$. In this example this means that $\ell_\pi(l_o)$ satisfies formula $K_1 p$ iff all situations that are R_1^K-indistinguishable from $\ell_\pi(l_o)$, satisfy p. This means in particular that all situations that are 1-equivalent to $\ell_\pi(l_o)$ must satisfy p. However, since $p \in \mathcal{P}_2$ is a proposition local to agent 2, the tableau rules only ensure that all R_2^K-indistinguishable situations satisfy p. They do not ensure that all R_1^K-indistinguishable situations satisfy p.

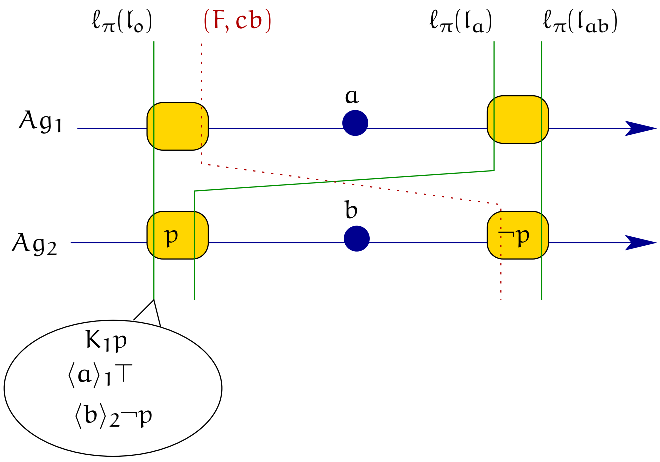


Figure 6.1: Counterexample for completeness

However, the tableau system is complete for a reasonable subset of the language $\mathcal{L}$ that contains only those knowledge formulae $K_i\phi$ in which ϕ is of type i, i.e. $\phi \in \Phi_{\{i\}}$. We claim that this subset is a reasonable subset for the following reason: In the unrestricted version of $\mathcal{L}$, it is possible to state that an agent i knows a fact about another agent j without being able to notice the change of the fact. In the restricted version of $\mathcal{L}$, it is only possible to formulate knowledge of agent i about facts, the change in truth-value of which agent i is able to observe.

However, the example in the next chapter, chapter 7 shows, that sometimes it is desirable to not have the restricted language: An agent may for example know constraints about the behavior of other agents and of the system or about the effects of operations performed by other agents.

The subset, for which the tableau method is complete, will be called $\mathcal{L}^c$. The language $\mathcal{L}^c$ is basically defined as $\mathcal{L}$ with two differences: Firstly, it does not contain belief operators $\mathcal{B}_i$ (for which the tableau was not constructed anyway) and secondly, the knowledge operators K_i are defined with the restriction described above.

Definition 6.2.1 (Language $\mathcal{L}^c$) *Let Ag be a set of agents. $\phi \in \mathcal{L}$ is a formula of $\mathcal{L}^c$ iff*

1. *ϕ does not contain any belief operator $\mathcal{B}_i$ for $i \in Ag$ and*
2. *for each agent $i \in Ag$ and for each sub formula of ϕ of the form $K_i\psi$ it holds that $\psi \in \Phi_{\{i\}}$ is of type i.*

Let $\Gamma_0 \subset \mathcal{L}^c$ be a finite set of formulae and let $\Gamma_0^{\mathcal{T}}$ be the tableau set constructed from Γ_0 according to definition 5.3.3. We will show that the existence of an accepted tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ for tableau set $\Gamma_0^{\mathcal{T}}$ implies that the tableau set is $\mathcal{L}-$satisfiable.

This means that for a set of tableau formulae $\Gamma_0^{\mathcal{T}}$ that is not $\mathcal{L}-$satisfiable, there does not exist an accepted tableau and thus, all tableaux constructed from $\Gamma_0^{\mathcal{T}}$ must be rejected.

If we only had to deal with finite tableaux, i.e. tableaux with only finite paths, we could prove completeness as follows: Given an accepted tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$. According to definition 5.5.9, there must exist an accepted path π in the tableau. Take the leaf of the accepted path π and construct a model that satisfies the leaf. Then show by induction that the first node of π, i.e. the root of the tableau is $\mathcal{L}-$satisfiable. This means, we have to show that whenever a denominator of a rule is $\mathcal{L}-$satisfiable, so is the numerator.

In our case the problem is that the tableaux are potentially infinite, and not only finite but also infinite paths can be accepted. There is no leaf that can be used as first node for the induction. Thus, we have to modify the method.

A tableau that accepts a set of formulae $\Gamma_0^{\mathcal{T}}$ must have an accepted path. This means, the tableau must contain a path π that is either

- finite and does not end with an axiom or
- infinite, fair and does not contain an $\mathcal{U}$-trace.

We prove completeness by

1. defining a method for constructing a model $\mathcal{M}$ for the set of tableau formulae Γ_π occurring on an accepted path π and an $\mathcal{L}$-embedding $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ and
2. showing,
   - that the structure constructed according to this method is a model in the sense of definition 3.4.1, and
   - that the mapping ℓ_π constructed according to this method is a satisfying $\mathcal{L}$-embedding for Γ_π and thus for $\Gamma_0^{\mathcal{T}}$ which is a subset of Γ_π.

Since $\Gamma_0^{\mathcal{T}} \subseteq \Gamma_\pi$, it obviously holds that if Γ_π is $\mathcal{L}-$satisfiable, then so is $\Gamma_0^{\mathcal{T}}$.

6.2.1 Construction of a Model $\mathcal{M}$ and a satisfying $\mathcal{L}$-embedding

As mentioned above, we first present a method to construct a model $\mathcal{M}$ and a satisfying $\mathcal{L}$-embedding $\ell : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ for a set of tableau formulae Γ_π occurring on an accepted path π in a tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$.

Requiring an accepted path as input, the method consists of three parts: Part 1 constructs the temporal part of the model, i.e. it constructs the set of runs $\mathcal{A}$. Further, part 1 also constructs the $\mathcal{L}$-embedding ℓ_π that embeds all tableau labels into situations of the runs. The second part of the construction constructs the epistemic part of the model, i.e. the indistinguishability relations R_i^K. Finally, the third part of the method constructs the interpretation function of the model $\mathcal{M}$ under construction.

In the following, we briefly describe the method informally:

We can assume that the set of agents Ag, the distributed set of operations $\tilde{\mathcal{O}}$ and the distributed set of propositions $\tilde{\mathcal{P}}$ is fixed.

We modify the set of operations and extend it by an init-operation, that is jointly performed by all agents. The first operation of each constructed run will be this init-operation. There is only a syntactic reason for adding this operation: In the construction method we will construct a number of runs and extend them successively. To be able to distinguish among the various runs, we initially add one event to each run in which every agent participates and that is labelled by operation init.

However, we could also use any other operation instead of init, as long as the event is unique to the run.

To construct a model, we start with an accepted path $\pi = \nu_0\nu_1\nu_2\nu_3, \ldots$. We successively consider for every node $\nu_j, j \geq 1$ the difference between Γ_j and Γ_{j-1}.

If ν_j is derived from ν_{j-1} by application of rule (**TR 9**), then $\Gamma_j \setminus \Gamma_{j-1}$ contains a new tableau label l that is (and will remain) the first label of a (new) maximal chain. For this tableau label we construct a new run $F = (E, \leq, \lambda)$ such that E contains only one event. This event is labelled by init and is used to distinguish F from other runs. We embed the tableau label into the situation (F, E).

If ν_j is derived from ν_{j-1} by the application of rule (**TR 14**), then $\Gamma_j \setminus \Gamma_{j-1}$ contains a new tableau label l and a new structure formula of the form $l' \xrightarrow{a} l$ for some tableau label l' and for some operation a. We consider the situation (F, c) into which tableau label l' is embedded and extend run F by a new event e. For $i \in ag(a)$ we define e to be greater (with respect to relation $\leq$ defined in definition 3.2.2) than all other i-events that already belong to run F. We then embed l into the configuration that results from configuration c by performing operation a, i.e. we embed tableau label l in configuration $c \cup \{e\}$.

If ν_j is derived from ν_{j-1} by any other tableau rule, we don't do anything as then the set $\Gamma_j \setminus \Gamma_{j-1}$ does not contain any new tableau labels.

Now every tableau label is embedded into a situation of a run. However, each run may also contain situations that are not images of tableau labels, i.e. the $\mathcal{L}$-embedding ℓ is not surjective. We can view each tableau path as representing one interleaving of a partially ordered run in the sense that those situations of a run that are images of tableau labels form one interleaving of this run. However, there may exist other interleavings of the same run which consist of other situations. The next two parts of the construction method are divided into two steps each: In the first step, the indistinguishability relation for knowledge and the interpretation function are constructed for situations that are images of tableau labels, respectively. In the second step, the indistinguishability relation and the interpretation function are extended for all situations.

After having constructed the temporal part of the model, i.e. the set of runs $\mathcal{A}$ and after having constructed the $\mathcal{L}$-embedding we construct the indistinguishability relations R_i^K for knowledge in two steps: First we define the indistinguishability relations restricted to those situations that are images of tableau labels: For all tableau formulae of type lR_il', we define the images of the tableau labels to be in the relation $R_i^{K,pre}$: $(\ell_\pi(l), \ell_\pi(l')) \in R_i^{K,pre}$. In the second step we extend the relation $R_i^{K,pre}$ to the complete relation R_i^K. We add all pairs of situations of $|\mathcal{M}|$ to the relation that are i-equivalent. Since the indistinguishability relation is an equivalence relation, we finally construct the transitive, reflexive and symmetric closure of the relation $R_i^{K,pre}$ and the pairs of i-equivalent situations.

In the third part of the construction method, we construct the interpretation function. Again we proceed in two steps: First we define the interpretation function $\mathcal{I}^{pre}$ for all pairs of situations (F, c) and propositions, such that (F, c) is an image of a tableau label $\mathfrak{l}$. We define the interpretation function to be true for all such pairs $(\mathfrak{l}, p)$, for which there exists a formula $\mathfrak{l} \vdash p \in \Gamma_\pi$. For all other pairs we define the interpretation function to be false. In the second step we extend the interpretation function for all situations that are not an image of a tableau label. We define those pairs of situations (F, c) and propositions p to be true, for which there exists an image $\ell_\pi(\mathfrak{l})$ of a tableau label $\mathfrak{l}$, such that $((F, c), \ell_\pi(\mathfrak{l})) \in R_i^K$ and $\mathcal{I}^{pre}(\ell_\pi(\mathfrak{l}), p) = \top$.

Construction method for constructing a model $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ and a satisfying $\mathcal{L}$-embedding ℓ for $\Gamma_0^{\mathcal{T}}$ from an accepted tableau path π.

We assume that the signature $(Ag, \tilde{\mathcal{O}}, \tilde{\mathcal{P}})$ is known. We first add an auxiliary operation 'init' to the set of operations that will be executed once in the beginning of each run. It will only be used to distinguish among various runs that have only the initial event.

For all $i \in Ag$ we define $\mathcal{O}_i^* := \mathcal{O}_i \cup \{\text{init}\}$, such that $\mathcal{O}^* = \mathcal{O} \cup \{\text{init}\}$.

Let $\pi = v_0 v_1 v_2 v_3 \ldots$ be an accepted path of a regular tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ for a tableau set $\Gamma_0^{\mathcal{T}}$.

Part 1: Construction of the set of runs $\mathcal{A}$ and an $\mathcal{L}$-embedding ℓ_π.

Start with $\mathcal{A}_0 := \{F_0^0\}$ such that $F_0^0 = (E, \leq, \lambda)_0^0$ where

$$E = \{e\} \tag{13}$$
$$\leq = \{(e, e)\} \tag{14}$$
$$\lambda : E \longrightarrow \mathcal{O}^*, \lambda(e) = \text{init} \tag{15}$$

Define $\ell_0 : \mathfrak{label}(\Gamma_0) \longrightarrow |\mathcal{A}|$ with

$$\ell_0(\mathfrak{l}_o) := (F_0^0, E) \text{ for } \mathfrak{l}_o \in \mathfrak{label}(\Gamma_0). \tag{16}$$

Set $x := 1$.

For $j \in N^+$ **do begin**

- If v_j is derived from v_{j-1} by application of one of the rules (**TR 3**), (**TR 4**), (**TR 5**), (**TR 6**), (**TR 7**), (**TR 8**), (**TR 10**), (**TR 11**), (**TR 12**), (**TR 13**), (**TR 15**), (**TR 16**), (**TR 18**), (**TR 19**), (**TR 20**), (**TR 21**), (**TR 22**), (**TR 23**), (**TR 24**), (**TR 25**), (**TR 26**)[1], then

set

$$\mathcal{A}_j := \mathcal{A}_{j-1} \text{ and} \tag{17}$$

$$\ell_j : \mathfrak{label}(\Gamma_j) \longrightarrow |\mathcal{A}|, \text{ with}$$

$$\ell_j(\mathfrak{l}) := \ell_{j-1}(\mathfrak{l}). \tag{18}$$

- If ν_j is derived from ν_{j-1} by application of rule (**TR 9**), then

  derive $\mathcal{A}_j$ from $\mathcal{A}_{j-1}$ as follows:
  $\mathcal{A}_j := \mathcal{A}_{j-1} \cup \{F_0^x\}$ where $F_0^x = (E, \leq, \lambda)$ with

  $$E := \{e\} \text{ for a } \textit{new} \text{ event } e, \tag{19}$$

  $$\leq := \{(e, e)\}, \tag{20}$$

  $$\lambda : E \longrightarrow \mathcal{O}^*, \lambda(e) = \text{init} \tag{21}$$

  Derive $\ell_j : \mathfrak{label}(\Gamma_j) \longrightarrow |\mathcal{A}|$ from ℓ_{j-1} as follows:

  $$\ell_j(\mathfrak{l}) = \begin{cases} \ell_{j-1}(\mathfrak{l}) & \text{for } \mathfrak{l} \in \mathfrak{label}(\Gamma_{j-1}) \\ (F_0^x, E) & \text{for } \mathfrak{l} \in \mathfrak{label}(\Gamma_j) \setminus \mathfrak{label}(\Gamma_{j-1}) \end{cases} \tag{22}$$

  Increment x.

- If ν_j is derived from ν_{j-1} by application of rule (**TR 14**), then

  there exists a tableau label $\mathfrak{la} \in \mathfrak{label}(\Gamma_j) \setminus \mathfrak{label}(\Gamma_{j-1})$ and a tableau formula $\mathfrak{l} \xrightarrow{a} \mathfrak{la} \in \Gamma_j \setminus \Gamma_{j-1}$.

  Further, there exists a situation $(F_y^z, c) \in |\mathcal{A}|$ such that $\ell_{j-1}(\mathfrak{l}) = (F_y^z, c)$, where $F_y^z = (E, \leq, \lambda)_y^z$ is a run and $c \subseteq E$ is a set of events.

  Derive $\mathcal{A}_j$ from $\mathcal{A}_{j-1}$ as follows:

  Replace $F_y^z = (E, \leq, \lambda)_y^z$ in $\mathcal{A}_{j-1}$ by $F_{y+1}^z = (E', \leq', \lambda')_{y+1}^z$ with

[1]Note, that rules (**TR 1**), (**TR 2**) and (**TR 17**) cannot be applied on accepted paths.

$$E' := E \cup \{e\}, \text{ for a } \textit{new} \text{ event } e. \tag{23}$$

$$\leq' := (\leq \cup \{\ (e', e) \mid e' \in E_i \text{ for } i \in ag(a)\} \cup \{(e, e)\}\ )^t \tag{24}$$

where X^t is the transitive closure of X.

$\lambda' : E' \longrightarrow \mathcal{O}^*$,

$$\lambda'(e') := \begin{cases} a & \text{for } e' = e \\ \lambda(e) & \text{otherwise} \end{cases} \tag{25}$$

Derive $\ell_j : \mathfrak{label}(\Gamma_j) \longrightarrow |\mathcal{A}|$ from ℓ_{j-1} as follows:

$$\ell_j(\mathfrak{l}') := \begin{cases} (F^z_{y+1}, E') & \text{for } \mathfrak{l}' = \mathfrak{la} \\ (F^z_{y+1}, d) & \text{for } \ell_{j-1}(\mathfrak{l}') = (F^z_y, d), \mathfrak{l}' \neq \mathfrak{la} \\ (F^n_m, d) & \text{for } \ell_{j-1}(\mathfrak{l}') = (F^n_m, d), \mathfrak{l}' \neq \mathfrak{la} \text{ and } n \neq z \end{cases} \tag{26}$$

end do

Note, that the definitions of the ℓ_j's are such that there exist unique values $x_\mathfrak{l}$ and $d_\mathfrak{l}$ for each tableau label $\mathfrak{l} \in \mathfrak{label}(\Gamma_\pi)$ with the following property:
for all $j \in N$ it holds that if $\ell_j(\mathfrak{l}) = (F^x_y, d)$ then $x = x_\mathfrak{l}$ and $d = d_\mathfrak{l}$.

"After the completion" of the loop, which has infinitely many iterations (for all $j \in N$), we construct each run F^x as follows:

$$F^x := \bigcup_{y \in N} F^x_y = (\bigcup_{y \in N} E^x_y, \bigcup_{y \in N} \leq^x_y, \bigcup_{y \in N} \lambda^x_y) \tag{27}$$

Now we construct the (possibly infinite) set $\mathcal{A}$ of all (possibly infinite) runs F^x and the $\mathcal{L}$-embedding ℓ_π as follows:

$$\mathcal{A} := \{F^x \mid x \in N\} \tag{28}$$

$$\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{A}|, \text{ with } \ell_\pi(\mathfrak{l}) = (F^{x_\mathfrak{l}}, d_\mathfrak{l}) \tag{29}$$

Part 2: In the second part of the construction method we construct the indistinguishability relation $\mathcal{R}^K \subseteq |\mathcal{A}| \times |\mathcal{A}|$ as follows:
For all $i \in Ag$ start with $R^{K,0}_i := \emptyset$.
For all $j \in N^+$ **do begin**

- If ν_j is derived from ν_{j-1} by application of one of the rules (**TR 3**), (**TR 4**), (**TR 5**), (**TR 6**), (**TR 7**), (**TR 8**), (**TR 12**), (**TR 13**), (**TR 14**), (**TR 15**), (**TR 16**), (**TR 18**), (**TR 19**), (**TR 20**), (**TR 21**), (**TR 22**), (**TR 23**), (**TR 25**):

  $$\text{Construct } R_i^{K,j} := R_i^{K,j-1}. \tag{30}$$

- If ν_j is derived from ν_{j-1} by application of one of the rules (**TR 9**), (**TR 10**), (**TR 11**), (**TR 24**), (**TR 26**):
  For $\mathfrak{l}R_i\mathfrak{l}' \in \Gamma_j \setminus \Gamma_{j-1}$ construct $R_i^{K,j}$ from $R_i^{K,j-1}$ as follows:

  $$R_i^{K,j} := R_i^{K,j-1} \cup \{((F,c),(F',c')) \mid \ell_\pi(\mathfrak{l}) = (F,c), \ell_\pi(\mathfrak{l}') = (F',c')\} \tag{31}$$

end do

"After the completion" of the loop, which has again infinitely many iterations (for all $j \in \mathbb{N}^+$), we construct $R_i^{K,pre}$ as the (infinite) union of all $R_i^{K,j}$, that is,

$$R_i^{K,pre} := \bigcup_{j\in\mathbb{N}^+} R_i^{K,j}. \tag{32}$$

We then extend $R_i^{K,pre}$ to R_i^K as follows:

$$R_i^K := (R_i^{K,pre} \cup \{((F,c),(F,c')) \mid (F,c),(F,c') \in |\mathcal{A}|, (F,c) \equiv_i (F,c')\})^{trs} \tag{33}$$

where $(X)^{trs}$ is the reflexive, transitive and symmetric closure of X.

Part 3: In the third part of the construction method we construct the interpretation function $\mathcal{I} : \{(F,c) \mid F \in \mathcal{A}, c \subseteq E \text{ for } F = (E,\leq,\lambda)\} \times \mathcal{P} \longrightarrow \{\top,\bot\}$ in two steps:

First we define the interpretation function for all situations that are images of tableau labels:

$$\mathcal{I}^{pre}(\ell_\pi(\mathfrak{l}),p) := \begin{cases} \top & \text{if } \mathfrak{l} \vdash p \in \Gamma_\pi \\ \bot & \text{otherwise} \end{cases} \tag{34}$$

Then we extend the interpretation function for all situations in $\mathcal{A}$:

$$\mathcal{I}((F,c),p) := \begin{cases} \bot & \text{if } c = \emptyset \\ \mathcal{I}^{pre}((F,c),p) & \text{if } (F,c) \text{ is an image of a tableau label.} \\ \mathcal{I}^{pre}((F',c'),p) & \text{if } (F,c) \text{ is no image of a tableau label} \\ & \text{ and } ((F,c),(F',c')) \in R_i^K \\ & \text{ and } (F',c') \text{ is an image of a tableau label.}^2 \end{cases} \tag{35}$$

Finally, we construct the desired model $\mathcal{M} := (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$.

In the following we will show that the model $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ constructed according to the construction method above does actually fulfill the constraints of definition 3.4.1. Further, we have to show that the mapping ℓ is actually a satisfying $\mathcal{L}$-embedding in the sense of definition 5.3.2.

Let us briefly recall the requirements for $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ to be a model:

- $\mathcal{A} = \{F_1, F_2, F_3, \ldots\}$ is a set of runs if for every run $F_i = (E, \leq, \lambda)_i$ the following holds:
  - E is a denumerable set.
  - $\leq \subseteq E \times E$ is a partial order on the set of events, $\leq \cap (E_i \times E_i)$ is a total order on the set of i-local events.
  - $\lambda : E \longrightarrow \mathcal{O} \cup \{\text{init}\}$ is a mapping from the set of events into the set of operations.
- $\mathcal{R}^K$ is the set of indistinguishability relations for each agent, if the following holds:
  - For every agent $i \in Ag$ the relation R_i^K is an equivalence relation.
  - If $((F,c),(F,c')) \in R_i^K$ for all $i \in Ag$ then $c = c'$.
  - If for two situations $(F,c),(F,c') \in |\mathcal{A}|$ we have that $(F,c) \equiv_i (F,c')$ then $((F,c),(F,c')) \in R_i^K$.
  - If for two situations $(F,c_1),(F,c_2) \in |\mathcal{A}|$ holds, that $(F,c_1) \xrightarrow{op} (F,c_2)$ and there exists $(F',c_1') \in |\mathcal{A}|$ such that for every $i \in ag(op)$ it holds that $((F,c_1),(F',c_1')) \in R_i^K$ then there exists $(F',c_2') \in \mathcal{A}$ such that $(F',c_1') \xrightarrow{op} (F',c_2')$ and $((F,c_2),(F',c_2')) \in R_i^K$ for every $i \in ag(op)$.

[2] From lemmas 6.2.8 and 6.2.16 it follows that this is a partition on the set of situations.

- $\mathcal{I} : |\mathcal{M}| \times \mathcal{P} \longrightarrow \{\top, \bot\}$ is an interpretation function if the following holds: For all $(F, c), (F', c') \in |\mathcal{M}|$ with $((F, c), (F', c')) \in R_i^K$ and for all $p \in \mathcal{P}_i$ it holds that $\mathcal{I}((F, c), p) = \mathcal{I}((F', c'), p)$.

Let us also briefly recall the requirement for a satisfying $\mathcal{L}$-embedding ℓ_π : $\mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{A}|$:

- ℓ_π must be a total mapping.
- If $\mathfrak{l} \vdash \phi \in \Gamma_\pi$ then $\ell_\pi(\mathfrak{l}) \models_{\mathcal{M}} \phi$.
- If $\mathfrak{l} \xrightarrow{a} \mathfrak{l}' \in \Gamma_\pi$ then $\ell_\pi(\mathfrak{l}) \xrightarrow{a} \ell_\pi(\mathfrak{l})$.
- If $\mathfrak{l} R_i \mathfrak{l}' \in \Gamma_\pi$ then $(\ell_\pi(\mathfrak{l}), \ell_\pi(\mathfrak{l}')) \in R_i^K$.

Further we must also show that for every tableau label $\mathfrak{l} \in \mathfrak{label}(\Gamma_\pi)$ it holds that if $\ell_\pi(\mathfrak{l}) = (F, c)$, then c is actually a configurations of run F, i.e. c is a downward closed set of events.

If all these requirements hold for the constructed model $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ and for the $\mathcal{L}$-embedding ℓ_π, then ℓ_π is a satisfying $\mathcal{L}$-embedding for the set of formulae Γ_π and thus also for $\Gamma_0^{\mathcal{T}}$ which is a subset of Γ_π.

First we show that $\mathcal{A}$ is a set of runs according to definition 3.2.2.

Proposition 6.2.2 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure constructed from path π by the construction method on page 133.

For each $F \in \mathcal{A}$ it holds that $F = (E, \leq, \lambda)$ is a run in the sense of definition 3.2.2.

Proof:

1. **Claim:** E is a denumerable set of events.
   This follows directly from construction steps (13), (19), and (23).

2. **Claim:** $\leq$ is a partial order, i.e. $\leq$ is reflexive, transitive and antisymmetric.

   a) Let $e \in E$ be events of run F.
      $\Longrightarrow$ There exists a smallest $j \in \mathbf{N}$, such that $e \in E_j \setminus E_{j-1}$
      $\Longrightarrow$ (* according to construction steps (14), (14) and (24) *)
      $(e, e) \in \leq_j \setminus \leq_{j-1}$
      $\Longrightarrow$ (* by construction of $\leq = \bigcup_{k \in \mathbf{N}} \leq_k$ *)
      it holds that $(e, e) \in \leq$

b) Let $e_1, e_2, e_3 \in E$ be events of run F.

$\Longrightarrow$ there exists a smallest $j \in N$, such that $e_1, e_2, e_3 \in E_j$ and $(e_1, e_2) \in \leq_j$,and $(e_2, e_3) \in \leq_j$

$\Longrightarrow$ (* according to construction steps (14), (14) and (24) *) $(e_1, e_3) \in \leq_j$

$\Longrightarrow$ (* by construction of $\leq = \bigcup_{k \in N} \leq_k$ *) $(e_1, e_3) \in \leq$

c) Let us first prove the following sub-lemma:

Lemma 6.2.3 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*
Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure constructed from path π by the construction method on page 133.
For all $j \in N$ it holds that if $e \in E^j \setminus E^{j-1}$ then for all pairs $(e_1, e_2) \in \leq_j \setminus \leq_{j-1}$ we have that $e_2 = e$.

Proof: According to step (24) of the construction method, $\leq_k$ is constructed from $\leq_{k-1}$ such that $\leq_k = (\leq_{k-1} \cup \{(e', e) \mid e' \in E_i \text{ for } i \in ag(a)\} \cup \{(e, e)\})^t$. Since $e \in E_k \setminus E_{k-1}$, we know that for all pairs $(e_1, e_2) \in \leq_{k-1}$ it holds that $e_1 \neq e$ and $e_2 \neq e$. Obviously, for all pairs $(e_1, e_2) \in \{(e', e) \mid e' \in E_i \text{ for } i \in ag(a)\} \cup \{(e, e)\}$ it holds that $e_2 = e$. According to construction steps (14), (20) and (24), $\leq_{k-1}$ is already closed under transitivity. Consequently, for all pairs $(e_1, e_2) \in \leq_k$ it holds that $e_2 = e$. □

Now we prove by contradiction that $\leq$ is antisymmetric.
Suppose, $\leq$ is not antisymmetric, i.e. there exists a pair of events $e_1, e_2 \in E$ with $e_1 \neq e_2$ such that $(e_1, e_2) \in \leq$ and $(e_2, e_1) \in \leq$.

$\Longrightarrow$ w.l.o.g there exist $j, k, m \in N$ with $j < k \leq m$, s.t. $e_1 \in E_j \setminus E_{j-1}$ and $e_2 \in E_k \setminus E_{k-1}$ and $\{(e_1, e_2), (e_2, e_1)\} \subseteq \leq_m$ and since steps (13), (19), (23) each time use a *new* event, it holds for all $j' > j$ that $e_1 \notin E_{j'} \setminus E_{j'-1}$ and it holds for all $k' > k$ that $e_2 \notin E_{k'} \setminus E_{k'-1}$

$\Longrightarrow$ (* with sub lemma 6.2.3 *) $(e_2, e_1) \in \leq_j \setminus \leq_{j-1}$ and $(e_1, e_2) \in \leq_k \setminus \leq_{k-1}$

$\Longrightarrow$ $(e_2, e_1) \in \leq_j \setminus \leq_{j-1}$ is a contradiction to the fact that $e_2 \in E_k \setminus E_{k-1}$ and $j < k$.

$\Longrightarrow$ there cannot exist events $e_1, e_2 \in E$ with $e_1 \neq e_2$ and $(e_1, e_2) \in \leq$ and $(e_2, e_1) \in \leq$.

$\Longrightarrow$ $\leq$ is antisymmetric

3. **Claim:** For all $i \in Ag$ the relation $\leq \cap (E_i \times E_i)$ is a total order.
We prove this claim by contradiction.
Suppose, $\leq \cap (E_i \times E_i)$ is not a total order.
$\Longrightarrow$ There exists a pair of events $e_1, e_2 \in E$ with $i \in ag(\lambda(e_1))$ and $i \in ag(\lambda(e_2))$ and $(e_1, e_2) \notin \leq$ and $(e_2, e_1) \notin \leq$.
$\Longrightarrow$ w.l.o.g. there exist $j, k \in \mathbb{N}$ with $j < k$ and $e_1 \in E_j \setminus E_{j-1}$ and $e_2 \in E_k \setminus E_{k-1}$.
$\Longrightarrow$ (* by construction step (23) *)
$e_1 \in E_k$ and $e_2 \in E_k \setminus E_{k-1}$ and $i \in ag(\lambda_k(e_1))$ and $i \in ag(\lambda_k(e_2))$.
$\Longrightarrow$ (* by construction step (24) *)
$(e_1, e_2) \in \leq_k$
$\Longrightarrow$ (* by construction step (27) *)
$(e_1, e_2) \in \leq$
$\Longrightarrow$ contradiction to the assumption that $(e_1, e_2) \notin \leq$ and $(e_2, e_1) \notin \leq$.

4. **Claim:** λ is a total mapping on the set of events.
This follows directly from steps (15), (21), (25), (27) of the construction.

$\square$

We now investigate some correlations between the constructed runs and the tableau set Γ_π.

Lemma 6.2.4 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau and let π be an accepted path in $\mathcal{T}$. Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure constructed from path π by the construction method and let $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ be the constructed mapping.*

For all tableau labels $\mathfrak{l}$ it holds that if $\ell_\pi(\mathfrak{l}) = (F^z, c)$ then c is a downward closed set of events.

Proof: Let $\ell_\pi(\mathfrak{l}) = (F^z, c)$ with $F^z = (E, \leq, \lambda)$. Then there exists a $j \in \mathbb{N}$ such that $\mathfrak{l} \in \Gamma_j \setminus \Gamma_{j-1}$.

By part 1 of the construction there exists a $y \in \mathbb{N}$ such that $\ell_j(\mathfrak{l}) = (F_y^z, E')$ with $F_y^z = (E', \leq', \lambda')$. Obviously E' is a downward closed set of events with respect to $\leq'$.

From construction step (24) it follows that for all events $e \in E \setminus E'$ there does not exist an event $e' \in E'$ such that $e \leq e'$.

Hence, from step (27) it follows that $c \subset E$ is a downward closed set of events with respect to $\leq$.

$\square$

In lemma 6.2.5 we will now show that for each tableau formula $\mathfrak{l} \xrightarrow{a} \mathfrak{la}$ the following holds: If by $\mathcal{L}$-embedding ℓ_π tableau label $\mathfrak{l}$ is embedded into situation $\ell_\pi(\mathfrak{l})$ and tableau label $\mathfrak{la}$ is embedded into situation $\ell_\pi(\mathfrak{la})$ then in the model $\mathcal{M}$ we have that $\ell_\pi(\mathfrak{l}) \xrightarrow{a} \ell_\pi(\mathfrak{la})$.

Lemma 6.2.5 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau and let π be an accepted path in $\mathcal{T}$. Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure constructed from path π by the construction method and let $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ be the constructed mapping.*

For tableau labels $\mathfrak{l}, \mathfrak{la}$ it holds that $\mathfrak{l} \xrightarrow{a} \mathfrak{la} \in \Gamma_\pi$ iff $\ell_\pi(\mathfrak{l}) \xrightarrow{a} \ell_\pi(\mathfrak{la})$.

Proof:

"$\Longrightarrow$" Let $\mathfrak{l} \xrightarrow{a} \mathfrak{la} \in \Gamma_\pi$ and let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the model constructed by the construction method.
According to part 1 of the construction, there exists a $j \in \mathbb{N}$, such that for $\ell_j(\mathfrak{l}) = (F, c)$ and $\ell_j(\mathfrak{la}) = (F, d)$ there exists an event $e \in E$ with $d = c \cup \{e\}$ and $\lambda(e) = a$.
By definition 3.2.3 this means, that $(F, c) \xrightarrow{a} (F, d)$.
Steps (26) and (29) imply that this holds for all $\ell_k, k > j$ and in particular for ℓ_π, i.e. $\ell_\pi(\mathfrak{l}) \xrightarrow{a} \ell_\pi(\mathfrak{la})$.

"$\Longleftarrow$" Let $\ell_\pi(\mathfrak{l}) \xrightarrow{a} \ell_\pi(\mathfrak{la})$.

By definition 3.2.3 there exists an event $e \in E$ such that for $\ell_\pi(\mathfrak{l}) = (F^z, c)$ and $\ell_\pi(\mathfrak{la}) = (F^z, ca)$ and $F^z = (E, \leq, \lambda)$ it holds that $ca = c \cup \{e\}$ and $\lambda(e) = a$.

By construction of F^z and ℓ_π in steps (27) and (29) there exist $j, y \in \mathbb{N}$ such that $\mathfrak{la} \in \mathfrak{label}(\Gamma_j) \setminus \mathfrak{label}(\Gamma_{j-1})$ and $\ell_j(\mathfrak{la}) = (F_y^z, ca)$.

The only case in the construction that extends an already existing run is, if the considered node is derived by application of rule (**TR 14**).
Thus, $F_y^z = (E', \leq, \lambda')$ must have been constructed from F_{y-1}^z by steps (23) to (26). By step (26), it holds that $ca = E'$ and there must exist a tableau formula $\mathfrak{l}' \xrightarrow{a'} \mathfrak{la}$.

By lemma 5.5.4 it holds that $\mathfrak{l}' = \mathfrak{l}$ and $a' = a$ and hence $\mathfrak{l} \xrightarrow{a} \mathfrak{la} \in \Gamma_\pi$.

□

In the following proposition we show that for each $i \in Ag$ the relation R_i^K is an indistinguishability relation for knowledge according to definition 3.3.2.

Proposition 6.2.6 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau and let π be an accepted path in $\mathcal{T}$. Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure constructed from* *path* π *by the construction method and let* $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ *be the constructed* $\mathcal{L}$*-embedding.*

The relation R_i^K *is an indistinguishability relation for knowledge for each agent* $i \in Ag$.

Before we can prove this proposition, we need to investigate a number of properties of the constructed relations R_i^K. These properties will be used in the proof of the proposition.

In the following we will distinguish between those situations $(F, c) \in |\mathcal{M}|$ of the model that are *images* of tableau labels and those situations that are not images. Whenever we say, a situation (F, c) is an *image* of a tableau label, we mean that there exists a tableau label $\mathfrak{l} \in \Gamma_\pi$ such that $\ell_\pi(\mathfrak{l}) = (F, c)$. Note, that the relation $R_i^{K,pre}$, which is constructed in step (32), is only defined on images of tableau labels.

The first observation we make, is that for each pair $(F, c), (F', c')$ of images of tableau labels it holds that if $F = F'$, i.e. if the images belong to the same run, then this pair of labels is i-equivalent for some agent $i \in Ag$, $(F, c) \equiv_i (F', c')$ iff the pair of these labels is an element of the relation $R_i^{K,pre}$, i.e. $((F, c), (F', c')) \in R_i^{K,pre}$:

Lemma 6.2.7 *Let* $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ *be an accepted regular tableau for a tableau set* $\Gamma_0^{\mathcal{T}}$ *and let* π *be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ *be the structure and* ℓ_π *be the* $\mathcal{L}$*-embedding constructed from path* π *by the construction method on page 133.*

Let $(F, c), (F', c') \in |\mathcal{M}|$ *be images of tableau labels. Then for all* $i \in Ag$ *the following holds:*

1. *If* $((F, c), (F', c')) \in R_i^{K,pre}$ *and* $F = F'$, *then* $(F, c) \equiv_i (F', c')$.
2. *If* $(F, c) \equiv_i (F', c')$ *then* $((F, c), (F', c')) \in R_i^{K,pre}$.

Proof:

1. Let $((F, c), (F', c')) \in R_i^{K,pre}$ and $F = F'$.

$\Longrightarrow$ there exist tableau labels $\mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_\pi)$ such that $\ell_\pi(\mathfrak{l}) = (F, c)$ and $\ell_\pi(\mathfrak{l}') = (F', c')$

$\Longrightarrow$ (* by construction of $R_i^{K,pre}$, steps (30), (31), (32) *)
$\mathfrak{l} R_i \mathfrak{l}' \in \Gamma_\pi$

$\Longrightarrow$ (* by part 1 of the construction and the fact that $\mathfrak{l}$ and $\mathfrak{l}'$ are embedded into the same run *)
w.l.o.g. there exists a chain $\mathfrak{l} \xrightarrow{a_0} \cdots \xrightarrow{a_n} \mathfrak{l}'$ and $\mathfrak{l} R_i \mathfrak{l}' \in \Gamma_\pi$

$\Longrightarrow$ (* by lemma 6.2.5 *)
there exists a chain $\mathfrak{l} \xrightarrow{a_0} \cdots \xrightarrow{a_n} \mathfrak{l}'$ and $(F, c) = \ell_\pi(\mathfrak{l}) \xrightarrow{a_0} \cdots \xrightarrow{a_n} \ell_\pi(\mathfrak{l}') = (F, c')$ and $\mathfrak{l} R_i \mathfrak{l}' \in \Gamma_\pi$

$\Longrightarrow$ (* by definition 3.2.3 *)
there exists a chain $\mathfrak{l} \xrightarrow{a_0} \cdots \xrightarrow{a_n} \mathfrak{l}'$ and $\mathfrak{l} R_i \mathfrak{l}' \in \Gamma_\pi$ and for $\ell_\pi(\mathfrak{l}) = (F, c)$ and $\ell_\pi(\mathfrak{l}') = (F, c')$ we have that $c \subset c'$

$\Longrightarrow$ (* by lemma 5.6.16 *)
for all $j \leq n$ it holds that $a_j \notin \mathcal{O}_i$

$\Longrightarrow$ (* by part 1 of the construction *)
for all $e \in c' \setminus c$ it holds that $e \notin E_i$

$\Longrightarrow$ (* by definition 3.2.3 *)
$(F, c) \equiv_i (F', c')$

2. Let $(F, c), (F', c')$ be images of tableau labels and let $(F, c) \equiv_i (F', c')$.

$\Longrightarrow$ $F = F'$ and there exist tableau labels $\mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_\pi)$ such that $\ell_\pi(\mathfrak{l}) = (F, c)$ and $\ell_\pi(\mathfrak{l}') = (F', c')$

$\Longrightarrow$ (* by part 1 of the construction *)
there exists a chain $\mathfrak{l} \xrightarrow{a_0} \cdots \xrightarrow{a_n} \mathfrak{l}'$ in Γ_π, such that for all $j \leq n$ it holds that $o_j \notin \mathcal{O}_i$.

$\Longrightarrow$ (* by rules (**TR 11**) and (**TR 24**) and proposition 5.6.14 *)
$\mathfrak{l} R_i \mathfrak{l}' \in \Gamma_\pi$

$\Longrightarrow$ (* by construction step (31) *)
$((F, c), (F', c')) \in R_i^{K,pre}$

□

The next observation we make, is that for each situation $(F, c) \in |\mathcal{M}|$ and for each agent $i \in Ag$ there exists an image of a tableau label $(F, c') \in |\mathcal{M}|$ such that (F, c) and (F, c') are i-equivalent ($(F, c) \equiv_i (F, c')$).

Lemma 6.2.8 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure constructed from path π by the construction method on page 133 and let $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ be the constructed mapping.

Let $(F, c) \in |\mathcal{M}|$ *with* $c \neq \emptyset$ *(i.e.* c *contains at least the initial operation init) be a situation of the constructed model. For each* $i \in Ag$ *there exists a situation* $(F, c') \in |\mathcal{M}|$ *for which there exists a tableau label* $l \in \mathfrak{label}(\Gamma_\pi)$ *such that* $\ell_\pi(l) = (F, c')$ *and* $(F, c) \equiv_i (F, c')$.

Proof:

Let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be a model constructed by the construction method above and let $F = (E, \leq, \lambda) \in \mathcal{A}$ be a run of $\mathcal{M}$.

According to the construction of $\mathcal{M}$ each event $e \in E$ with $\lambda(e) \neq$ init is inserted into the set of events E by step (23) of the construction.

By steps (24), (25) and (26) and by the construction of ℓ_π in step (29) of the construction method the following holds:
For each agent $i \in Ag$ and for each event $e \in E_i$ there exists a tableau label l such that $\ell_\pi(l) = (F, d)$ with $d \in E$ and the following property holds:
$e = max_{\leq}(d \cap E_i)$. [3]

This means, that for each situation $(F, c) \in |\mathcal{M}|$ and for each $i \in Ag$ there exists a tableau label l with $\ell_\pi(l) = (F, d)$, such that $\max_{\leq}(d \cap E_i) = \max_{\leq}(c \cap E_i)$.

Hence, according to definition 3.2.3 of i-view it holds that for each situation $(F, c) \in |\mathcal{M}|$ there exists a tableau label l such that $\ell_\pi(l) \equiv_i (F, c)$.

□

In the next two lemmas we show the following two properties:

1. For all images of tableau labels $(F, c), (F', c')$ that if $((F, c), (F', c')) \in R_i^K$ then already $((F, c), (F', c')) \in R_i^{K,pre}$. This means that the construction of R_i^K does lead to any new related pairs of images of tableau labels.

2. For every pair of situations $(F, c), (F, c')$ belonging to the same run of a model constructed by the method defined above it holds that (F, c) and (F, c') are R_i^K-indistinguishable, $((F, c), (F, c')) \in R_i^K$, iff they are i-equivalent, $(F, c) \equiv_i (F, c')$. This means that the construction of the reflexive, transitive and symmetric closure does add only pairs of situations to the R_i^K-indistinguishability relation that belong to different chains.

In the rest of this section let $R_{\equiv_i} := \{((F_1, c_1), (F_1, c_2)) \mid (F_1, c_1), (F_1, c_2) \in |\mathcal{A}|, (F_1, c_1) \equiv_i (F_1, c_2)\}$. In the proofs of the following two lemmas, we shall see the set of situations $|\mathcal{M}|$ as nodes and the relation $R_i^{K,pre}$ and $R_{\equiv_i}$ as the edges of a directed graph.

[3] Note, that according to lemma 6.2.4, $\ell_\pi(l)$ is a configuration, i.e., $\ell_\pi(l)$ is downward closed.

Note, that $R_{\equiv_i}$ is an equivalence relation since $\equiv_i$ is an equivalence relation. Further, by tableau rules (**TR 10**) and (**TR 11**), by proposition 5.6.14 and by construction steps (31) and (32), the relation $R_i^{K,pre}$ is closed under symmetry and transitivity.

We define an $\equiv_i$-maximal sub-path of a path p as follows:

Definition 6.2.9 *Let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be a model constructed by the construction method above. Let $G = (|\mathcal{M}|, R_i^{K,pre} \cup R_{\equiv_i})$ be a graph, such that the situations of $\mathcal{M}$ are the nodes and the relations $R_i^{K,pre}$ and $R_{\equiv_i}$ are the edges of the graph.*

Let a path p' between nodes s and s' be a sub-path of a path p. We call sub-path p' $\equiv_i$-maximal *if the following holds:*

- *either s is the first node in p or the edge in p leading to s is in $R_i^{K,pre}$, and*
- *either s' is the last node in p or the edge in p leading from node s' is in $R_i^{K,pre}$ and*
- *all edges in sub-path p' are in $R_{\equiv_i}$*

Lemma 6.2.10 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure constructed from path π by the construction method on page 133.

Let $(F, c), (F', c') \in |\mathcal{M}|$ be images of tableau labels. For each agent $i \in Ag$ it holds that if $((F, c), (F', c')) \in R_i^K$, then $((F, c), (F', c')) \in R_i^{K,pre}$.

Proof: Let $((F, c), (F', c')) \in R_i^K$ and let $(F, c), (F', c')$ be images of tableau labels.

Then $((F, c), (F', c')) \in R_i^K$ iff there exists a path p from (F, c) to (F', c') in the graph. (Note, that p may contain edges of $R_i^{K,pre}$ as well as of $R_{\equiv_i}$.)

For each $\equiv_i$-maximal sub-path p' of p, where p' leads from a node s to a node s' it holds that s and s' are images of tableau labels belonging to the same run F.
Since $R_{\equiv_i}$ is an equivalence relation, it holds that $(s, s') \in R_{\equiv_i}$ and thus $s \equiv_i s'$.
By lemma 6.2.7 we have that $(s, s') \in R_i^{K,pre}$.

We have shown that each $\equiv_i$-maximal sub-path of p can be substituted by an edge in $R_i^{K,pre}$. Hence, there exists a path p'' from (F, c) to (F', c'), such that all edges of p'' belong to $R_i^{K,pre}$.

Since $R_i^{K,pre}$ is closed under transitivity, there exists an edge $((F, c), (F', c')) \in R_i^{K,pre}$.

□

Lemma 6.2.11 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^{K}, \mathcal{I})$ be the structure constructed from path π by the construction method on page 133.

Let $(F, c), (F, c') \in |\mathcal{M}|$ be situations belonging to the same run F. For all agents $i \in Ag$ it holds that $((F, c), (F, c')) \in R_i^K$ iff $(F, c) \equiv_i (F, c')$.

Proof:

"$\Longrightarrow$": Let $((F, c), (F, c')) \in R_i^K$.
This means, there exists a path p in $R_i^{K,pre} \cup R_{\equiv_i}$ from (F, c) leading to (F, c'). If all edges of p are in $R_{\equiv_i}$, then we are already done. Otherwise, by lemma 6.2.8 there exists a prefix p' of path p from (F, c) to some node s such that s is an image of a tableau label and p' contains only edges of $R_{\equiv_i}$. Hence, since $R_{\equiv_i}$ is an equivalence relation, there exists an edge of $R_{\equiv_i}$ between (F, c) and s. Similarly, there exists a sub-path p'' of p leading from some node s' to (F, c') such that all edges of p'' are of $R_{\equiv_i}$ and hence there also exists an edge between s' and (F, c') in $R_{\equiv_i}$.

Since there exists a path between s and s' in R_i^K and s, s' are images of tableau labels, it holds that by lemma 6.2.10 there exists an edge between node s and node s' in $R_i^{K,pre}$.

Note, that the nodes s and s' must both belong to run F. By lemma 6.2.7 there exists an edge between node s and node s' in $R_{\equiv_i}$.

This means that there exists an edge in $R_{\equiv_i}$ from (F, c) to s, from s to s' and also from s' to (F, c'). Since $R_{\equiv_i}$ is an equivalence relation, it finally holds that $((F, c), (F, c')) \in R_{\equiv_i}$.

"$\Longleftarrow$": This direction follows directly from construction step (33).

□

Lemma 6.2.12 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^{K}, \mathcal{I})$ be the structure constructed from path π by the construction method on page 133.

Let $(F, c), (F, c') \in |\mathcal{M}|$ be situations with $(F, c) \neq (F, c')$. There exists an agent $i \in Ag$ such that $((F, c), (F, c')) \in R_i^K$.

Proof: $(F, c), (F, c') \in |\mathcal{M}|$ with $(F, c) \neq (F, c')$
$\Longrightarrow$ there exists an agent $i \in Ag$ such that $(F, c) \not\equiv_i (F, c')$
$\Longrightarrow$ $((F, c), (F, c')) \notin R_{\equiv_i}$
$\Longrightarrow$ (* by lemma 6.2.11 and since (F, c) and (F, c') belong to the same run *)
$((F, c), (F, c')) \notin R_i^K$

□

The relation R_i^K constructed according to the method described in the beginning of this section has a property that is not necessary for an indistinguishability relation for knowledge, but that will help us for proving lemma 6.2.14:

Lemma 6.2.13 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure and ℓ_π be the mapping constructed from path π by the construction method on page 133.

Let $(F, c), (F', c') \in |\mathcal{M}|$ be situations of to different runs: $F \neq F'$. There exists at most one agent $i \in Ag$ with $((F, c), (F, c')) \in R_i^K$.

Proof: Let $i \in Ag$ be an agent and let $((F, c), (F', c')) \in R_i^K$ and $F \neq F'$.

$\Longrightarrow$ (* by lemma 6.2.8 *)
$((F, c), (F', c')) \in R_i^K$ and there exist images of tableau labels $(F, \bar{c}), (F', \bar{c}') \in |\mathcal{M}|$, such that $(F, c) \equiv_i (F, \bar{c})$ and $(F', c') \equiv_i (F', \bar{c}')$
$\Longrightarrow$ (* by step (33) and by lemma 6.2.10 *)
$((F, \bar{c}), (F, \bar{c}')) \in R_i^K$ and $((F, \bar{c}), (F, \bar{c}')) \in R_i^{K,pre}$
$\Longrightarrow$ (* by part 2 of the model construction *)
there exist tableau labels $\mathfrak{l}, \mathfrak{l}' \in \mathfrak{label}(\Gamma_\pi)$ with $\ell_\pi(\mathfrak{l}) = (F, \bar{c})$ and $\ell_\pi(\mathfrak{l}') = (F, \bar{c'})$ and $\mathfrak{l} R_i \mathfrak{l}' \in \Gamma_\pi$
$\Longrightarrow$ (* since $F \neq F'$ and by part 1 of the construction *)
$\mathfrak{l}, \mathfrak{l}'$ belong to different maximal chains in Γ_π
$\Longrightarrow$ (* by lemma 5.6.18 and corollary 5.6.19 *)
for all agents $j \neq i$ and for all tableau labels $\bar{\mathfrak{l}}, \bar{\mathfrak{l}}'$, such that $\mathfrak{l}$ belongs to the same chain as $\bar{\mathfrak{l}}$ and $\mathfrak{l}'$ belongs to the same chain as $\bar{\mathfrak{l}}'$, it holds that $\bar{\mathfrak{l}} R_j \bar{\mathfrak{l}}' \notin \Gamma_\pi$
$\Longrightarrow$ (* by construction steps (31) and (32) *)
for all $j \neq i$ and for all images $((F, \bar{\bar{c}}), (F', \bar{\bar{c'}})) \notin R_i^{K,pre}$
$\Longrightarrow$ (* by lemma 6.2.10 *)
for all agents $j \neq i$ it holds that $((F, c), (F', c')) \notin R_i^K$.

□

In the next lemma we will show that each relation R_i^K satisfies the condition of an indistinguishability relation that says that the enabling and the effect of operations is only dependent on those agents that participate in the operation.

Lemma 6.2.14 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure and $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ constructed from path π by the construction method on page 133.

If for two situations $(F, c_1), (F, c_2) \in |\mathcal{M}|$ holds that $(F, c_1) \xrightarrow{op} (F, c_2)$ and there exists $(F', c_1') \in |\mathcal{M}|$ such that for every $i \in ag(op)$ it holds that $((F, c_1), (F', c_1')) \in R_i^K$ then there exists $(F', c_2') \in |\mathcal{M}|$ such that $(F', c_1') \xrightarrow{op} (F', c_2')$ and $((F, c_2), (F', c_2')) \in R_i^K$ for every $i \in ag(op)$.

Proof: Suppose, for two situations $(F, c), (F, ca) \in |\mathcal{M}|$ it holds that $(F, c) \xrightarrow{a} (F, ca)$ and there exists $(F', c') \in |\mathcal{M}|$ with $((F, c), (F', c')) \in R_i^K$ for every $i \in ag(a)$:

$$\begin{tikzcd} (F, c) \arrow[r, "a"] \arrow[d, "R_i^K", "\text{forall } i \in ag(a)"'] & (F, ca) \\ (F', c') & \end{tikzcd}$$

Figure 6.2: starting position

We then distinguish between two cases:

1. $F = F'$:

$\Longrightarrow$ (* by lemma 6.2.11 on page 146 *)
$(F, c) \equiv_i (F', c')$:

$$\begin{tikzcd} (F, c) \arrow[r, "a"] \arrow[d, "\equiv_i", "\text{forall } i \in ag(a)"'] & (F, ca) \\ (F', c') & \end{tikzcd}$$

$\Longrightarrow$ (* by definition 3.2.3 of i-equivalence *)
there exists a situation $(F', c'a) \in |\mathcal{M}|$ with $(F', c') \xrightarrow{a} (F', c'a)$:

$$\begin{tikzcd}
(F, c) \arrow[r, "a"] \arrow[d, "\text{forall } i \in ag(a)"', "\equiv_i"] & (F, ca) \\
(F', c') \arrow[r, "a"] & (F', c'a)
\end{tikzcd}$$

$\Longrightarrow$ (* by lemma 3.2.4 on page 25 *)
there exists a situation $(F', c'a) \in |\mathcal{M}|$ with $(F', c') \xrightarrow{a} (F', c'a)$ and $(F, ca) \equiv_i (F', c'a)$:

$$\begin{tikzcd}
(F, c) \arrow[r, "a"] \arrow[d, "\text{forall } i \in ag(a)"', "\equiv_i"] & (F, ca) \arrow[d, "\text{forall } i \in ag(a)"', "\equiv_i"] \\
(F', c') \arrow[r, "a"] & (F', c'a)
\end{tikzcd}$$

$\Longrightarrow$ (* by lemma 6.2.11 on page 146 *)
for all $i \in ag(a)$ it holds that $((F, ca), (F', c'a)) \in R_i^K$:

$$\begin{tikzcd}
(F, c) \arrow[r, "a"] \arrow[d, "\text{forall } i \in ag(a)"', "R_i^K"] & (F, ca) \arrow[d, "\text{forall } i \in ag(a)"', "R_i^K"] \\
(F', c') \arrow[r, "a"] & (F', c'a)
\end{tikzcd}$$

2. $F \neq F'$:

$\Longrightarrow$ (* by lemma 6.2.13 on page 147 *)
there exists exactly one $i \in Ag$ with $((F, c), (F', c')) \in R_i^K$

$\Longrightarrow$ (* by lemma 6.2.8 on page 143 *)
there exist situations $(F, \bar{c}), (F, \bar{c}a), (F', \bar{c}') \in |\mathcal{M}|$ with
$(F, c) \equiv_i (F, \bar{c}), (F, ca) \equiv_i (F, \bar{c}a)$ and $(F', c') \equiv_i (F', \bar{c}')$
and for which there exist tableau labels
$(\mathfrak{F}, \bar{\mathfrak{c}}), (\mathfrak{F}', \bar{\mathfrak{c}}'_0), (\mathfrak{F}, \bar{\mathfrak{c}}\mathfrak{a}) \in \mathfrak{label}(\Gamma_\pi)$ such that
$\ell_\pi((\mathfrak{F}, \bar{\mathfrak{c}})) = (F, \bar{c})$, $\ell_\pi((\mathfrak{F}', \bar{\mathfrak{c}}'_0)) = (F', \bar{c}')$ and $\ell_\pi((\mathfrak{F}, \bar{\mathfrak{c}}\mathfrak{a})) = (F, \bar{c}a)$

$$\begin{tikzcd}
(\mathfrak{F},\bar{\mathfrak{c}}) \arrow[r, bend right, "\ell_\pi"'] & (F,\bar{c}) \arrow[d, phantom, "\equiv_i"] & (F,\bar{c}a) \arrow[d, phantom, "\equiv_i"] & (\mathfrak{F},\bar{\mathfrak{c}}\mathfrak{a}) \arrow[l, bend left, "\ell_\pi"] \\
& (F,c) \arrow[r, "a"] \arrow[d, "R_i^K"] & (F,ca) & \\
& (F',c') \arrow[d, phantom, "\equiv_i"] & & \\
(\mathfrak{F}',\bar{\mathfrak{c}}'_0) \arrow[r, bend left, "\ell_\pi"] & (F',\bar{c}'_0) & &
\end{tikzcd}$$

$\Longrightarrow$ (* by definition 3.2.3 *)
$(F, \bar{c}) \xrightarrow{a} (F, \bar{c}a)$

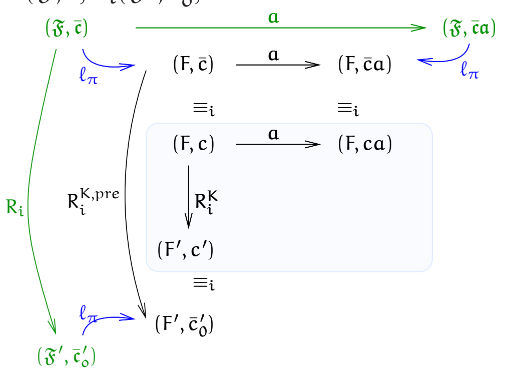
$\Longrightarrow$ (* by lemma 6.2.5 *)
$(\mathfrak{F}, \bar{\mathfrak{c}}) \xrightarrow{\mathfrak{a}} (\mathfrak{F}, \bar{\mathfrak{c}}\mathfrak{a}) \in \Gamma_\pi$

$\Longrightarrow$ (* by lemma 6.2.11 and since R_i^K is transitive and symmetric *)
$((F, \bar{c}), (F', \bar{c}')) \in R_i^K$

$\Longrightarrow$ (* by lemma 6.2.10 *)
$((F, \bar{c}), (F', \bar{c}')) \in R_i^{K,pre}$

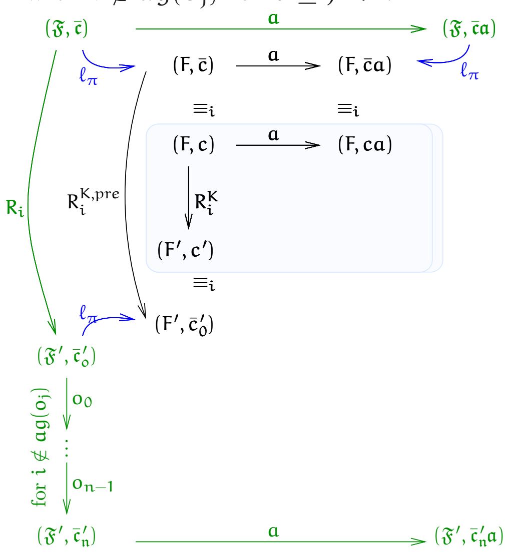
$$\begin{tikzcd}
(\mathfrak{F},\bar{\mathfrak{c}}) \arrow[rrr, bend left, "\mathfrak{a}"] \arrow[r, bend right, "\ell_\pi"'] & (F,\bar{c}) \arrow[r, "a"] \arrow[d, phantom, "\equiv_i"] \arrow[ddd, bend right=40, "R_i^{K,pre}"'] & (F,\bar{c}a) \arrow[d, phantom, "\equiv_i"] & (\mathfrak{F},\bar{\mathfrak{c}}\mathfrak{a}) \arrow[l, bend left, "\ell_\pi"] \\
& (F,c) \arrow[r, "a"] \arrow[d, "R_i^K"] & (F,ca) & \\
& (F',c') \arrow[d, phantom, "\equiv_i"] & & \\
(\mathfrak{F}',\bar{\mathfrak{c}}'_0) \arrow[r, bend left, "\ell_\pi"] & (F',\bar{c}'_0) & &
\end{tikzcd}$$

$\Longrightarrow$ (* by construction of $R_i^{K,pre}$ in steps (30), (31) and (32) *)
$(\mathfrak{F}, \bar{\mathfrak{c}})R_i(\mathfrak{F}', \bar{\mathfrak{c}}'_o)$

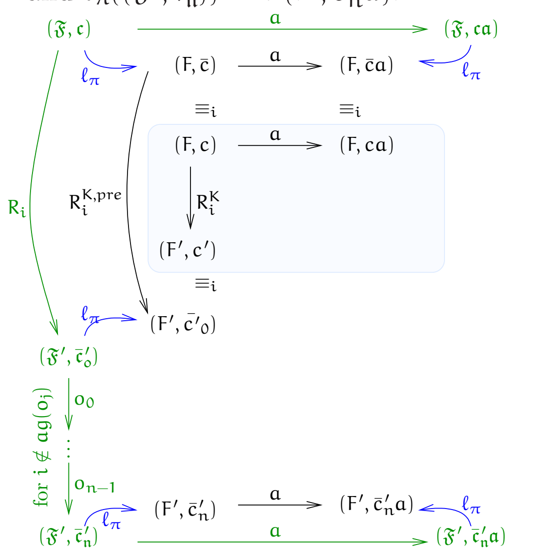


$\Longrightarrow$ (* by rule **TR 25** applied to tableau rules $(\mathfrak{F}, \bar{\mathfrak{c}}) \xrightarrow{a} (\mathfrak{F}, \bar{\mathfrak{c}}a)$ and $(\mathfrak{F}, \bar{\mathfrak{c}})R_i(\mathfrak{F}', \bar{\mathfrak{c}}'_o)$ and by proposition 5.6.14 *)
$(\mathfrak{F}', \bar{\mathfrak{c}}'_o) \vdash \langle a \rangle_i \top \in \Gamma_\pi$

$\Longrightarrow$ (* π is an accepted path (fair, no axiom) *)
Γ_π contains a finite chain $(\mathfrak{F}', \bar{\mathfrak{c}}'_o) \xrightarrow{o_0} \ldots \xrightarrow{o_{n-1}} (\mathfrak{F}', \bar{\mathfrak{c}}'_n) \xrightarrow{a} (\mathfrak{F}', \bar{\mathfrak{c}}'_n a)$ with $i \notin ag(o_j)$ for $0 \leq j < n$:

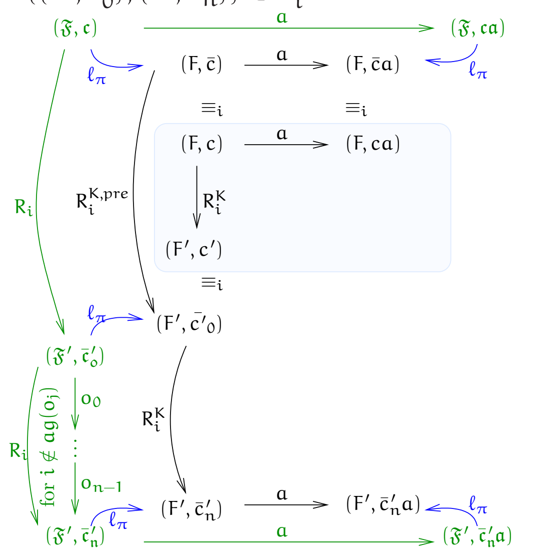


$\Longrightarrow$ (* by part 1 of the construction and lemma 6.2.5 *)
there exists a situation $(F', \bar{c}'_n a) \in |\mathcal{M}|$ with $\ell_\pi((\mathfrak{F}', \bar{\mathfrak{c}}'_n \mathfrak{a})) = (F', \bar{c}'_n a)$ and $\ell_\pi((\mathfrak{F}', \bar{\mathfrak{c}}'_n)) \xrightarrow{a} (F', \bar{c}'_n a)$:



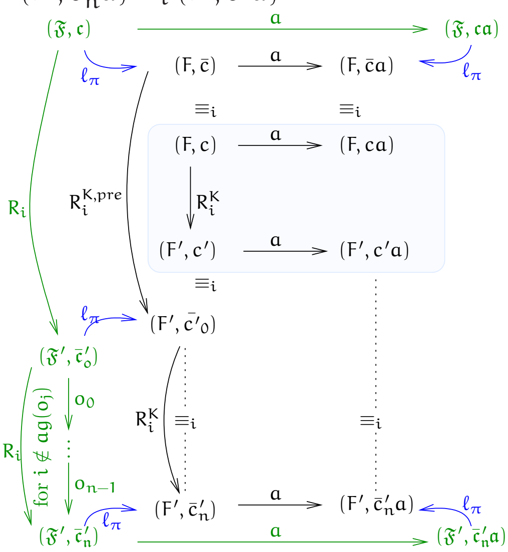
$\Longrightarrow$ (* by tableau rules (**TR 24**) and (**TR 11**) and proposition 5.6.14 *)
$(\mathfrak{F}', \bar{\mathfrak{c}}'_0) R_i (\mathfrak{F}', \bar{\mathfrak{c}}'_n) \in \Gamma_\pi$

$\Longrightarrow$ (* by construction steps (31), (32) and (33) *)
$((F', \bar{c}'_0), (F', \bar{c}'_n)) \in R_i^K$



$\Longrightarrow$ (* by lemma 6.2.11 *)
$(F', c') \equiv_i (F', \bar{c}'_n)$

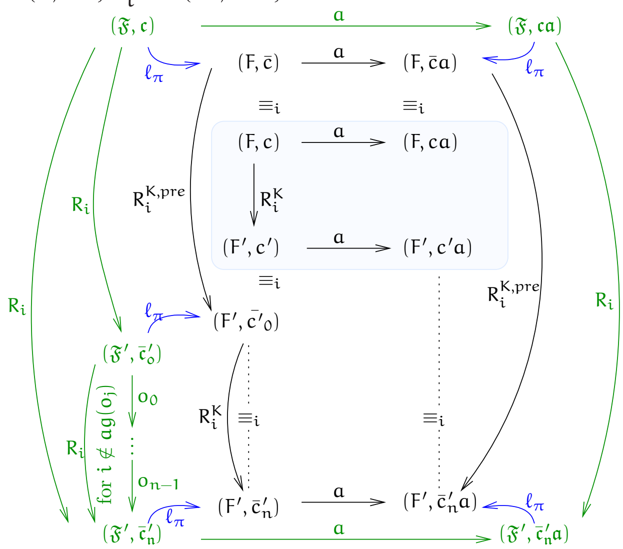
$\Longrightarrow$ (* by definition 3.2.3 and lemma 3.2.4 *)
there exists a situation $(F', c'a)$ such that $(F', c') \xrightarrow{a} (F', c'a)$ and $(F', \bar{c}'_n a) \equiv_i (F', c'a)$



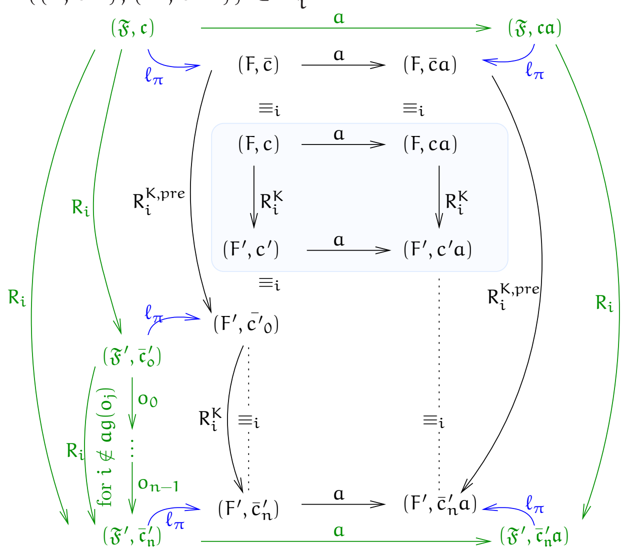
$\Longrightarrow$ (* by rule (**TR 11**) and proposition 5.6.14 *)
$(\mathfrak{F}, \mathfrak{c})R_i(\mathfrak{F}', \bar{\mathfrak{c}}'_n) \in \Gamma_\pi$

$\Longrightarrow$ (* by rule **TR 26** and proposition 5.6.14 *)
$(\mathfrak{F}, \bar{\mathfrak{c}}\mathfrak{a})R_i(\mathfrak{F}', \bar{\mathfrak{c}}'_n\mathfrak{a}) \in \Gamma_\pi$

$\Longrightarrow$ (* by construction steps (31), (32) *)
$(F, ca)R_i^{K,pre}(F', c'a)$



$\Longrightarrow$ (* by construction step (33) *)
$((F, ca), (F', c'a)) \in R_i^K$



$\square$

Using the lemmas shown above, we can now easily prove proposition 6.2.6, which says that the relation R_i^K is an indistinguishability relation according to definition 3.3.2:

Proof of proposition 6.2.6:

1. **Claim:** For every agent $i \in Ag$ the relation R_i^K is an equivalence relation.

   This follows directly from construction step (33).

2. **Claim:** If $((F, c), (F, c')) \in R_i^K$ for all $i \in Ag$ then $c = c'$.

   This follows directly from lemma 6.2.12.

3. **Claim:** If for two situations $(F, c), (F, c') \in |\mathcal{A}|$ we have that $(F, c) \equiv_i (F, c')$ then $((F, c), (F, c')) \in R_i^K$.

   This follows directly from construction step (33).

4. **Claim:** If for two situations $(F, c_1), (F, c_2) \in |\mathcal{A}|$ holds, that $(F, c_1) \xrightarrow{op} (F, c_2)$ and there exists $(F', c_1') \in |\mathcal{A}|$ such that for every $i \in ag(op)$ it holds that $((F, c_1), (F', c_1')) \in R_i^K$ then there exists $(F', c_2') \in \mathcal{A}$ such that $(F', c_1') \xrightarrow{op} (F', c_2')$ and $((F, c_2), (F', c_2')) \in R_i^K$ for every $i \in ag(op)$.

   This has been shown in the proof of lemma 6.2.14.

$\square$

Next, we have to show that the mapping $\mathcal{I}$ constructed in steps (34) and (35) of the model construction is actually an interpretation according to definition 3.3.4.

Proposition 6.2.15 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure and ℓ_π be the mapping constructed from path π by the construction method on page 133.

$\mathcal{I} : |\mathcal{M}| \times \mathcal{P} \longrightarrow \{\top, \bot\}$ is an interpretation function, i.e. the following holds: For all $(F, c), (F', c') \in |\mathcal{M}|$ with $((F, c), (F', c')) \in R_i^K$ and for all $p \in \mathcal{P}_i$ it holds that $\mathcal{I}((F, c), p) = \mathcal{I}((F', c'), p)$.

Recall, that the mapping $\mathcal{I}$ is constructed in two steps: In the first step a mapping $\mathcal{I}^{pre}$ is constructed and defined for all pairs of propositions and those situations that are images of tableau labels. In the second step the mapping is extended for all situations of the constructed runs.

We thus prove above proposition 6.2.15 in two parts: We first restrict to only those situations that are images of tableau labels. We prove a lemma which says that for all propositions p and for tableau labels $\mathfrak{l}, \mathfrak{l}'$ it holds that if $\mathfrak{l}R_i\mathfrak{l}' \in \Gamma_\pi$ then $\mathcal{I}^{pre}(\ell_\pi(\mathfrak{l})) = \mathcal{I}^{pre}(\ell_\pi(\mathfrak{l}'))$.

Using this lemma we can prove the proposition for all situations that occur in the structure. We then show, that for all agents i and for all situations $(F, c), (F', c') \in |\mathcal{M}|$ it holds that if (F, c) and (F', c') are i-indistinguishable, then for all propositions $p \in \mathcal{P}_i$ the situations (F, c) and (F', c') have the same interpretation.

Lemma 6.2.16 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*
Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure and $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ be the mapping constructed from path π by the construction method on page 133.
For all agents $i \in Ag$, for all propositions $p \in \mathcal{P}_i$ and for all images $(F, c), (F', c')$ of tableau labels the following holds:
If $((F, c), (F', c')) \in R_i^{K,pre}$ then $\mathcal{I}^{pre}((F, c), p) = \mathcal{I}^{pre}((F', c'), p)$

Proof:

1. Let $i \in Ag$. Let $((F, c), (F', c')) \in R_i^{K,pre}$ and let $\mathcal{I}^{pre}((F, c), p) = \top$.
   $\Longrightarrow$ (* by construction steps (31) and (32) *)
   there exist tableau labels $\mathfrak{l}, \mathfrak{l}'$ with $\ell_\pi(\mathfrak{l}) = (F, c)$ and $\ell_\pi(\mathfrak{l}') = (F', c')$ and $\mathfrak{l}R_i\mathfrak{l}' \in \Gamma_\pi$
   $\Longrightarrow$ (* by construction step (34) *)
   $\mathfrak{l} \vdash p \in \Gamma_\pi$ and $\mathfrak{l}R_i\mathfrak{l}' \in \Gamma_\pi$
   $\Longrightarrow$ (* by tableau rule (**TR 12**) and proposition 5.6.14 *)
   $\mathfrak{l}' \vdash p \in \Gamma_\pi$
   $\Longrightarrow$ (* by construction step (34) *)
   $\mathcal{I}^{pre}((F', c'), p) = \top$

2. Let $i \in Ag$. Let $((F, c), (F', c')) \in R_i^{K,pre}$ and let $\mathcal{I}^{pre}((F', c'), p) = \top$.
   $\Longrightarrow$ (* by construction steps (31) and (32) *)
   there exist tableau labels $\mathfrak{l}, \mathfrak{l}'$ with $\ell_\pi(\mathfrak{l}) = (F, c)$ and $\ell_\pi(\mathfrak{l}') = (F', c')$ and $\mathfrak{l}R_i\mathfrak{l}' \in \Gamma_\pi$
   $\Longrightarrow$ (* by tableau rule (**TR 10**) and proposition 5.6.14 *)
   $\mathfrak{l}R_i\mathfrak{l}' \in \Gamma_\pi$ and $\mathfrak{l}'R_i\mathfrak{l} \in \Gamma_\pi$
   $\Longrightarrow$ (* by construction step (34) *)
   $\mathfrak{l}' \vdash p \in \Gamma_\pi$ and $\mathfrak{l}'R_i\mathfrak{l} \in \Gamma_\pi$
   $\Longrightarrow$ (* by tableau rule (**TR 12**) and proposition 5.6.14 *)
   $\mathfrak{l} \vdash p \in \Gamma_\pi$
   $\Longrightarrow$ (* by construction step (34) *)
   $\mathcal{I}^{pre}((F, c), p) = \top$

$\square$

Next, we show that proposition 6.2.15 holds:

Proof of proposition 6.2.15:
We prove this lemma by contradiction.
Let $i \in Ag$ be an agent. Let $(F, c), (F', c') \in |\mathcal{M}|$ be situations with $((F, c), (F', c')) \in R_i^K$. Further, let $p \in \mathcal{P}_i$ be an i-local proposition.

Suppose, $\mathcal{I}((F, c), p) = \top$ and $\mathcal{I}((F', c'), p) = \bot$.

$\Longrightarrow$ (* by lemma 6.2.8 and construction step (35) *)
there exist situations $(F, \bar{c}), (F', \bar{c}') \in |\mathcal{M}|$ that are images of tableau labels and for which holds $(F, c) \equiv_i (F, \bar{c})$ and $(F', c') \equiv_i (F', \bar{c}')$ and $\mathcal{I}^{pre}((F, \bar{c}), p) = \top$ and $\mathcal{I}^{pre}((F', \bar{c}'), p) = \bot$

$\Longrightarrow$ (* by construction step (33) *)
$((F, \bar{c}), (F', \bar{c}')) \in R_i^K$ and $\mathcal{I}^{pre}((F, \bar{c}), p) = \top$ and $\mathcal{I}^{pre}((F', \bar{c}'), p) = \bot$

$\Longrightarrow$ (* by lemma 6.2.10 *)
$((F, \bar{c}), (F', \bar{c}')) \in R_i^{K,pre}$ and $\mathcal{I}^{pre}((F, \bar{c}), p) = \top$ and $\mathcal{I}^{pre}((F', \bar{c}'), p) = \bot$

$\Longrightarrow$ contradiction to lemma 6.2.16!

Since the relation R_i^K is an equivalence relation, the above proof holds also in the case that $\mathcal{I}((F, c), p) = \bot$ and $\mathcal{I}((F', c'), p) = \top$. Hence, for $((F, c), (F', c')) \in R_i^K$ and for $p \in \mathcal{P}_i$ it holds that $\mathcal{I}((F, c), p) = \mathcal{I}((F', c'), p)$.

□

From propositions 6.2.2, 6.2.6 and 6.2.15 we can now immediately infer, that the constructed structure $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ is a model in the sense of definition 3.4.1:

Theorem 6.2.17 *Let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted regular tableau for a tableau set $\Gamma_0^{\mathcal{T}}$ and let π be an accepted path in the tableau.*
Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure constructed from path π by the construction method on page 133.

Then $\mathcal{M}$ is a model in the sense of definition 3.4.1.

Proof:

1. **Claim:** $\mathcal{A}$ is a set of runs.
   This has been shown in proposition 6.2.2.

2. **Claim:** $R^K = (R_1^K, \ldots, R_n^K)$ is a family containing an indistinguishability relation R_i^K for each agent $i \in Ag$.
   This has been shown for each agent in proposition 6.2.6.

3. **Claim:** $\mathcal{I}$ is an interpretation function.
   This has been shown in proposition 6.2.15.

□

We finally have to prove that the constructed mapping ℓ_π is a satisfying $\mathcal{L}$-embedding in the sense of definition 5.3.2. There are three conditions for ℓ_π being a satisfying $\mathcal{L}$-embedding:

1. For each tableau formula $\mathfrak{l} \xrightarrow{a} \mathfrak{l}a \in \Gamma_\pi$ it must hold that $\ell_\pi(\mathfrak{l}) \xrightarrow{a} \ell_\pi(\mathfrak{l}a)$. This has already been shown in lemma 6.2.5.
2. For each tableau formula $\mathfrak{l}R_i\mathfrak{l}' \in \Gamma_\pi$ it must hold that $(\ell_\pi(\mathfrak{l}), \ell_\pi(\mathfrak{l}')) \in R_i^K$. This follows directly from the construction of $R_i^{K,pre}$ in steps (31) and (32) and from the construction of R_i^K in step (33).
3. For each tableau formula $\mathfrak{l} \vdash \phi \in \Gamma_\pi$ it must hold that $\ell_\pi(\mathfrak{l}) \models_\mathcal{M} \phi$. The proof of this condition requires more effort. We will prove this condition in proposition 6.2.18.

Proposition 6.2.18 *Let $\mathcal{T}(\Gamma_0^\mathcal{T})$ be an accepted regular tableau for a tableau set $\Gamma_0^\mathcal{T}$ and let π be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure and $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ be the mapping constructed from path π by the construction method on page 133.

Then for every labelled formula $\mathfrak{l} \vdash \phi \in \Gamma_\pi$ it holds that $\ell_\pi(\mathfrak{l}) \models_\mathcal{M} \phi$.

Consider the set of labelled tableau formulae $\Gamma_\pi \cap \mathfrak{L}_\mathfrak{l}$. From proposition 5.6.14 and the tableau rules it follows that for every formula all sub formulae are contained in Γ_π as well: If for example Γ_π contains a formula $\mathfrak{l} \vdash \phi \wedge \psi$ then by 5.6.14 and rule (**TR 4**) Γ_π also contains tableau formulae $\mathfrak{l} \vdash \phi$ and $\mathfrak{l} \vdash \psi$.

We define the size of a labelled tableau formula roughly as the number of propositions and operators occurring in the formula and prove the above proposition by induction over the size of the formulae.

Before we prove the hypothesis for a formula, we must have proven it for all its sub formulae.

The size of a formula is not directly dependent on the number of operators in the formula: Negation signs in tableau formulae are allowed only in front of propositions and in front of the knowledge operator. However, when processing a tableau formula $\mathfrak{l} \vdash \neg K_i\phi$ with tableau rule (**TR 9**), we create a tableau formula $\mathfrak{l}' \vdash \neg\phi$. Before this formula is entered into the tableau, it gets transformed according to table on page 52. This means, instead of counting the number of logical symbols of a negative tableau formula $\neg\phi$ we first have to transform $\neg\phi$ into a positive tableau formula and then count the number of logical symbols of this positive tableau formula.

One might wonder why we need a new metric for the size of formulae and do not reuse the one defined in 5.6.4 and 5.6.5. However, these definitions need a finite set of tableau formulae as context. Γ_π is a potentially infinite set with potentially infinite chains. Thus for the following proof we need a metric for the size of formulae that is independent of the context in which the formula occurs.

Definition 6.2.19 (size of a formula) *We define the size $|\phi|$ of a formula $\phi \in \mathcal{L}$ as follows:*

$$
\begin{array}{lll}
|\top| & & = 1 \\
|\bot| & & = 1 \\
|p| & & = 1 \textit{ for all } p \in \mathcal{P} \\
|\neg p| & & = 2 \textit{ for all } p \in \mathcal{P} \\
|\phi_1 \vee \phi_2| & & = |\phi_1| + 1 + |\phi_2| \\
|\neg(\phi_1 \wedge \phi_2)| & = |(\neg\phi_1) \vee (\neg\phi_2)| & = |\neg\phi_1| + 1 + |\neg\phi_2| \\
|\phi_1 \wedge \phi_2| & & = |\phi_1| + 1 + |\phi_2| \\
|\neg(\phi_1 \wedge \phi_2)| & = |(\neg\phi_1) \wedge (\neg\phi_2)| & = |\neg\phi_1| + 1 + |\neg\phi_2| \\
|\phi_1\ \mathcal{U}_i \phi_2| & & = |\phi_1| + 1 + |\phi_2| \\
|\neg(\phi_1\ \mathcal{U}_i \phi_2)| & = |(\neg\phi_2)\mathcal{W}_i(\neg\phi_1 \wedge \neg\phi_2)| & = 2|\neg\phi_2| + |\neg\phi_1| + 2 \\
|\phi_1\ \mathcal{W}_i \phi_2| & & = |\phi_1| + 1 + |\phi_2| \\
|\neg(\phi_1\ \mathcal{W}_i \phi_2)| & = |(\neg\phi_2)\mathcal{W}_i(\neg\phi_1 \wedge \neg\phi_2)| & = 2|\neg\phi_1| + |\neg\phi_1| + 2 \\
|K_i \phi| & & = |\phi| + 1 \\
|\neg K_i \phi| & & = |\neg\phi| + 1 \\
|\langle a \rangle_i \phi| & & = |\phi| + 1 \\
|\neg\langle a \rangle_i \phi| & = |[a]_i(\neg\phi)| & = |\neg\phi| + 1 \\
|[a]_i \phi| & & = |\phi| + 1 \\
|\neg[a]_i \phi| & = |\langle a \rangle_i(\neg\phi)| & = |\neg\phi| + 1
\end{array}
$$

Definition 6.2.20 (size of a tableau-formula) *We define the size of a labelled tableau formula $|(\mathfrak{F}, \mathfrak{c}) \vdash \phi| \in \mathfrak{L}_l$ as equal to the size $|\phi|$ of ϕ :*

$$|(\mathfrak{F}, \mathfrak{c}) \vdash \phi| := |\phi|$$

As explained above, we will prove proposition 6.2.18 by induction over the size of the formulae. For formulae of size 1 the induction hypothesis can easily be shown on the basis of the interpretation function defined in the model construction algorithm.

For formulae of a size greater than 1 we have to make a case differentiation depending on the outer most logical symbol. Remember, that Γ_π contains only positive formulae and formulae with negation signs only in front of a proposition or a knowledge operator. This means that we do not have to look at the cases of negative formulae.

Proof of proposition 6.2.18:

We show for every labelled formula $\mathfrak{l} \vdash \phi \in \Gamma_\pi$ that it holds that $\ell_\pi(\mathfrak{l}) \models_{\mathcal{M}} \phi$.

Induction begin:
Let $|(\mathfrak{F}, \mathfrak{c}) \vdash \phi| = 1$, then $(\mathfrak{F}, \mathfrak{c}) \vdash \phi$ is either

1. of the form $(\mathfrak{F}, \mathfrak{c}) \vdash p$ for any $p \in \mathcal{P}$, or
2. of the form $(\mathfrak{F}, \mathfrak{c}) \vdash \top$ for
3. of the form $(\mathfrak{F}, \mathfrak{c}) \vdash \bot$.

For the first case, the construction of the interpretation function $\mathcal{I}$ is such that $(\mathfrak{F}, \mathfrak{c}) \vdash p \in \Gamma_\pi$ implies that $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} p$, the second case is trivial since all situations satisfy $\top$, thus also $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \top$. The third case cannot occur in Γ_π: The assumption is that π is an accepted path that cannot contain a formula $(\mathfrak{F}, \mathfrak{c}) \vdash \bot$.

Induction hypothesis:
For all formula $(\mathfrak{F}, \mathfrak{c}) \vdash \phi$ with $|(\mathfrak{F}, \mathfrak{c}) \vdash \phi| \leq n$ it holds that

$$(\mathfrak{F}, \mathfrak{c}) \vdash \phi \in \Gamma_\pi \text{ implies } \ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi.$$

Induction step:
Suppose, the induction hypothesis holds and $|(\mathfrak{F}, \mathfrak{c}) \vdash \phi| = n + 1$. Then we have to differentiate between the following cases:

1. **Claim:** $(\mathfrak{F}, \mathfrak{c}) \vdash \phi = (\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \wedge \phi_2 \in \Gamma_\pi$ implies $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_1 \wedge \phi_2$:

   $(\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \wedge \phi_2 \in \Gamma_\pi$
   $\Longrightarrow$ (* by definition 5.5.1, rule (**TR 4**) and proposition 5.6.14 *)
   $(\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \in \Gamma_\pi$ and $(\mathfrak{F}, \mathfrak{c}) \vdash \phi_2 \in \Gamma_\pi$
   $\Longrightarrow$ (* because of the induction hypothesis *)
   $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_1$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_2$
   $\Longrightarrow$ (* by definition 4.2.1 *)
   $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_1 \wedge \phi_2$

2. **Claim:** $(\mathfrak{F}, \mathfrak{c}) \vdash \phi = (\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \vee \phi_2 \in \Gamma_\pi$ implies $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_1 \vee \phi_2$:

$(\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \vee \phi_2 \in \Gamma_\pi$
$\Longrightarrow$ (* by definition 5.5.1, rule (**TR 5**) and proposition 5.6.14 *)
$(\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \in \Gamma_\pi$ or $(\mathfrak{F}, \mathfrak{c}) \vdash \phi_2 \in \Gamma_\pi$
$\Longrightarrow$ (* by induction hypothesis *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \phi_1$ or $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \phi_2$
$\Longrightarrow$ (* by definition 4.2.1 *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \phi_1 \vee \phi_2$

3. **Claim:** $(\mathfrak{F}, \mathfrak{c}) \vdash \phi = (\mathfrak{F}, \mathfrak{c}) \vdash K_i\phi_1 \in \Gamma_\pi$ implies $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} K_i\phi_1$:

   $(\mathfrak{F}, \mathfrak{c}) \vdash K_i\phi_1 \in \Gamma_\pi$
   $\Longrightarrow$ (* by definition 5.5.1, rules (**TR 6**) and (**TR 12**) and by proposition 5.6.14 *)
   $(\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \in \Gamma_\pi$ and for all $(\mathfrak{F}', \mathfrak{c}') \in \mathfrak{label}(\Gamma_\pi)$ with $(\mathfrak{F}, \mathfrak{c})\ R_i\ (\mathfrak{F}', \mathfrak{c}') \in \Gamma_\pi$ holds that $(\mathfrak{F}', \mathfrak{c}') \vdash \phi_1$
   $\Longrightarrow$ (* by induction hypothesis *)
   $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \phi_1$ and for all $(\mathfrak{F}', \mathfrak{c}')$ with $(\mathfrak{F}, \mathfrak{c}) R_i (\mathfrak{F}', \mathfrak{c}') \in \Gamma_\pi$ holds that $\ell_\pi((\mathfrak{F}', \mathfrak{c}')) \models_\mathcal{M} \phi_1$
   $\Longrightarrow$ (* by lemma 4.3.2 and the restriction that $\phi_1 \in \Phi_{\{i\}}$[4] *)
   for all $(F', c') \in |\mathcal{M}|$ for which $(\ell_\pi((\mathfrak{F}, \mathfrak{c})), (F', c')) \in R_i^K$ it holds that $(F', c') \models_\mathcal{M} \phi_1$
   $\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\mathcal{L}$) *)
   $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} K_i\phi_1$

4. **Claim:** $(\mathfrak{F}, \mathfrak{c}) \vdash \phi = (\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i\phi_1 \in \Gamma_\pi$ implies $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \neg K_i\phi_1$:

   $(\mathfrak{F}, \mathfrak{c}) \vdash \neg K_i\phi_1 \in \Gamma_\pi$
   $\Longrightarrow$ (* by rule (**TR 9**) and proposition 5.6.14 *)
   there exists $(\mathfrak{F}', \mathfrak{c}') \in \mathfrak{label}(\Gamma_\pi)$ for which $(\mathfrak{F}, \mathfrak{c}) R_i (\mathfrak{F}', \mathfrak{c}') \in \Gamma_\pi$ and $(\mathfrak{F}', \mathfrak{c}') \vdash \neg\phi_1 \in \Gamma_\pi$
   $\Longrightarrow$ (* by induction hypothesis *)
   $(\mathfrak{F}, \mathfrak{c}) R_i (\mathfrak{F}', \mathfrak{c}') \in \Gamma_\pi$ and $\ell_\pi((\mathfrak{F}', \mathfrak{c}')) \models_\mathcal{M} \neg\phi_1$
   $\Longrightarrow$ (* by construction steps (31), (32) and (33) *)
   $(\ell_\pi((\mathfrak{F}, \mathfrak{c})), \ell_\pi((\mathfrak{F}', \mathfrak{c}'))) \in R_i^K$ and $(F', c') \models_\mathcal{M} \neg\phi_1$
   $\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\mathcal{L}$ *)
   $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \neg K_i\phi_1$

5. **Claim:** $(\mathfrak{F}, \mathfrak{c}) \vdash \phi = (\mathfrak{F}, \mathfrak{c}) \vdash \phi_1\mathcal{U}_i\phi_2 \in \Gamma_\pi$ implies $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \phi_1\mathcal{U}_i\phi_2$:

   $(\mathfrak{F}, \mathfrak{c}) \vdash \phi_1\mathcal{U}_i\phi_2 \in \Gamma_\pi$
   $\Longrightarrow$ (by rule (**TR 7**) and proposition 5.6.14 *)
   we distinguish between two possible cases:

[4] Recall, that we show completeness of the tableau system only for the subset of $\mathcal{L}$, in which for a formula $K_i\phi_1$ holds that $\phi_1 \in \Phi_{\{i\}}$.

a) $(\mathfrak{F}, \mathfrak{c}) \vdash \phi_2 \in \Gamma_\pi$:

$\Longrightarrow$ (* by induction hypothesis *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_2$

$\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\mathcal{L}$ *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_1 \, \mathcal{U}_i \phi_2$

b) $(\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \wedge \neg\phi_2 \wedge \langle \mathcal{O} \rangle_i \phi_1 \, \mathcal{U}_i \phi_2 \in \Gamma_\pi$:

$\Longrightarrow$ (* Path π is fair and does not contain an until-trace, by repeated applications of rules (**TR 14**), (**TR 20**), (**TR 22**), and (**TR 7**), as well as by proposition 5.6.14 *)
there exists a shortest chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} (\mathfrak{F}, \mathfrak{c}_1) \xrightarrow{o_1} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ of Γ_π with $(\mathfrak{F}, \mathfrak{c}_{m+1}) \vdash \phi_2 \in \Gamma_\pi$ and $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_o)$ and for all $k \leq m$ holds $(\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash \phi_1$:

$\Longrightarrow$ (* by algorithm 6.2.1 (page 133) and by ind. hypothesis *)
for all $k \leq m : \ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \xrightarrow{o_k} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{\mathfrak{k}+1}))$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \models_{\mathcal{M}} \phi_1$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1})) \models_{\mathcal{M}} \phi_2$ and $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_o)$

$\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\mathcal{L}$ *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_1 \, \mathcal{U}_i \phi_2$

6. **Claim:** $(\mathfrak{F}, \mathfrak{c}) \vdash \phi = (\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \mathcal{W}_i \phi_2 \in \Gamma_\pi$ implies $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_1 \mathcal{W}_i \phi_2$:

$(\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \mathcal{W}_i \phi_2 \in \Gamma_\pi$

$\Longrightarrow$ (* by rule (**TR 8**) and proposition 5.6.14 *)
there are two possibilities:

a) $(\mathfrak{F}, \mathfrak{c}) \vdash \phi_2 \in \Gamma_\pi$

$\Longrightarrow$ (* by the induction hypothesis *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_2$

$\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\mathcal{L}$ *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_1 \mathcal{W}_i \phi_2$

b) $(\mathfrak{F}, \mathfrak{c}) \vdash \phi_1 \wedge \neg\phi_2 \wedge [\mathcal{O}_i] \phi_1 \, \mathcal{W}_i \phi_2 \in \Gamma_\pi$

$\Longrightarrow$ (* Path π is fair, by repeated applications of rules (**TR 21**), (**TR 23**), and (**TR 8**), as well as by proposition 5.6.14 *)
we again distinguish between two cases:

i. there exists a shortest finite chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} (\mathfrak{F}, \mathfrak{c}_1) \xrightarrow{o_1} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ of Γ_π with $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_o)$ and $(\mathfrak{F}, \mathfrak{c}_{m+1}) \vdash \phi_2 \in \Gamma_\pi$ and for all $k \leq m$ it holds that $(\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash \phi_1 \in \Gamma_\pi$

$\Longrightarrow$ (* by lemma 6.2.5 and by the induction hypothesis *)
for all $k \leq m : \ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \xrightarrow{o_k} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{\mathfrak{k}+1}))$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \models_{\mathcal{M}} \phi_1$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1})) \models_{\mathcal{M}} \phi_2$ and $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_o)$

$\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\mathcal{L}$ *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_{\mathcal{M}} \phi_1 \mathcal{W}_i \phi_2$

ii. there exists a chain $\chi = (\mathfrak{F}, \mathfrak{c}_0) \xrightarrow{o_0} (\mathfrak{F}, \mathfrak{c}_1) \xrightarrow{o_1} \ldots$ of Γ_π and for all $k \in \mathbf{N} : (\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash \phi_1 \in \Gamma_\pi$ and $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_0)$
$\Longrightarrow$ (* by lemma 6.2.5 and by the induction hypothesis *)
$(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_0)$ and for all $k \in \mathbf{N} : \ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \xrightarrow{o_k} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{\mathfrak{k}+1}))$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \models_\mathcal{M} \phi_1$
$\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\mathcal{L}$ *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \phi_1 \mathcal{W}_i \phi_2$

7. **Claim:** $(\mathfrak{F}, \mathfrak{c}) \vdash \phi = (\mathfrak{F}, \mathfrak{c}) \vdash \langle a \rangle_i \phi_1 \in \Gamma_\pi$ implies $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \langle a \rangle_i \phi_1$:
$(\mathfrak{F}, \mathfrak{c}) \vdash \langle a \rangle_i \phi_1 \in \Gamma_\pi$
$\Longrightarrow$ (* by rule (**TR 14**), by proposition 5.6.14 and by fairness of path π *)
there exists a finite chain $\chi = (\mathfrak{F}, \mathfrak{c}_0) \xrightarrow{o_0} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ in Γ_π such that $(\mathfrak{F}, \mathfrak{c}_0) = (\mathfrak{F}, \mathfrak{c})$ and $o_m = a$ and for all $k < m : o_k \notin \mathcal{O}_i$
$\Longrightarrow$ (* by rules (**TR 15**) and (**TR 18**) and proposition 5.6.14 *)
for all $k \leq m$ holds that $(\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash \langle a \rangle_i \phi_1 \in \Gamma_\pi$ and $(\mathfrak{F}, \mathfrak{c}_{m+1}) \vdash \phi_1 \in \Gamma_\pi$
$\Longrightarrow$ (* by lemma 6.2.5 *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}_0)) \xrightarrow{o_0} \ldots \xrightarrow{o_m} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1}))$ and for all $k \leq m$ holds that $(\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash \langle a \rangle_i \phi_1 \in \Gamma_\pi$ and $(\mathfrak{F}, \mathfrak{c}_{m+1}) \vdash \phi_1 \in \Gamma_\pi$
$\Longrightarrow$ (* by induction hypothesis *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}_0)) \xrightarrow{o_0} \ldots \xrightarrow{o_m} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1}))$ and for all $k \leq m$ holds that $(\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash \langle a \rangle_i \phi_1 \in \Gamma_\pi$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1})) \models_\mathcal{M} \phi_1$
$\Longrightarrow$ (* by definition 3.2.3 *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}_0)) \xrightarrow{o_0} \ldots \xrightarrow{o_m} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1}))$ and for all $k \leq m$ holds that $(\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash \langle a \rangle_i \phi_1 \in \Gamma_\pi$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1})) \models_\mathcal{M} \phi_1$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_0)) \equiv_i \ell_\pi((\mathfrak{F}, \mathfrak{c}_m))$
$\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\mathcal{L}$ *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}_m)) \models_\mathcal{M} \langle a \rangle_i \phi_1$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_0)) \equiv_i \ell_\pi((\mathfrak{F}, \mathfrak{c}_m))$
$\Longrightarrow$ (* by lemma 3.2.4 *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}_0)) \models_\mathcal{M} \langle a \rangle_i \phi$
$\Longrightarrow$ (* since $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_0)$ *)
$(\mathfrak{F}, \mathfrak{c}) \models_\mathcal{M} \langle a \rangle_i \phi$

8. **Claim:** $(\mathfrak{F}, \mathfrak{c}) \vdash \phi = (\mathfrak{F}, \mathfrak{c}) \vdash [a]_i \phi_1 \in \Gamma_\pi$ implies $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} [a]_i \phi_1$:
$(\mathfrak{F}, \mathfrak{c}) \vdash [a]_i \phi_1 \in \Gamma_\pi$
$\Longrightarrow$ We distinguish among three possible cases:

a) there exists a finite chain $\chi = (\mathfrak{F}, \mathfrak{c}_0) \xrightarrow{o_0} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ of Γ_π and $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_0)$ and for all $k < m : o_k \notin \mathcal{O}_i$ and $o_m = a$
$\Longrightarrow$ (* analogous to the proof for $\langle a \rangle_i \phi_1$, but by application of rules (**TR 16**), (**TR 19**) instead of rules (**TR 15**) and (**TR 18**) *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} [a]_i \phi_1$

b) there exists a finite chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ in Γ_π and $(\mathfrak{F}, \mathfrak{c}_o) = (\mathfrak{F}, \mathfrak{c})$ and for all $k < m : o_k \notin \mathcal{O}_i$ and $o_m \in \mathcal{O}_i \setminus \{a\}$
$\Longrightarrow$ (* by rules (**TR 16**)and (**TR 19**) and prop. 5.6.14 *)
$(\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ with $(\mathfrak{F}, \mathfrak{c}_o) = (\mathfrak{F}, \mathfrak{c})$, $o_m \in \mathcal{O}_i \setminus \{a\}$, for all $k < m : o_k \notin \mathcal{O}_i$ and $(\mathfrak{F}, \mathfrak{c}_{m+1}) \vdash \phi_1 \in \Gamma_\pi$
$\Longrightarrow$ (* by lemma 6.2.5 *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}_o)) \xrightarrow{o_0} \ldots \xrightarrow{o_m} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1}))$ with $(\mathfrak{F}, \mathfrak{c}_o) = (\mathfrak{F}, \mathfrak{c})$, $o_m \in \mathcal{O}_i \setminus \{a\}$, for all $k < m : o_k \notin \mathcal{O}_i$ and $(\mathfrak{F}, \mathfrak{c}_{m+1}) \vdash \phi_1 \in \Gamma_\pi$
$\Longrightarrow$ (* by induction hypothesis *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}_o)) \xrightarrow{o_0} \ldots \xrightarrow{o_m} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1}))$ with $(\mathfrak{F}, \mathfrak{c}_o) = (\mathfrak{F}, \mathfrak{c})$, $o_m \in \mathcal{O}_i \setminus \{a\}$, for all $k < m : o_k \notin \mathcal{O}_i$ and for all $k \leq m$ it holds that $(\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash [a]_i\phi_1 \in \Gamma_\pi$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1})) \models_\mathcal{M} \phi_1$
$\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\mathcal{L}$ *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}_o)) \xrightarrow{o_0} \ldots \xrightarrow{o_m} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1}))$ with $(\mathfrak{F}, \mathfrak{c}_o) = (\mathfrak{F}, \mathfrak{c})$, $o_m \in \mathcal{O}_i \setminus \{a\}$, for all $k < m : o_k \notin \mathcal{O}_i$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_m)) \models_\mathcal{M} [a]_i\phi_1$
$\Longrightarrow$ (* by definition 3.2.3 and lemma 3.2.4 *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) = \ell_\pi((\mathfrak{F}, \mathfrak{c}_o)) \models_\mathcal{M} [a]_i\phi_1$

c) there exists an infinite chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} (\mathfrak{F}, \mathfrak{c}_1) \xrightarrow{o_1} \ldots$ in Γ_π with $(\mathfrak{F}, \mathfrak{c}_o) = (\mathfrak{F}, \mathfrak{c})$ and for all $k \in \mathbb{N} : o_k \notin \mathcal{O}_i$
$\Longrightarrow$ (* by lemma 6.2.5 *)
for all $k \in \mathbb{N}$ it holds that $\ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \xrightarrow{o}_k \ell_\pi((\mathfrak{F}, \mathfrak{c}_{\mathfrak{k}+1}))$ and $o_k \notin \mathcal{O}_i$
$\Longrightarrow$ (* by part 1 of the construction *)
Suppose, $\ell_\pi((\mathfrak{F}, \mathfrak{c})) = (F, c)$; for all events $e \in E$ with $e \in E \setminus c$ it holds that $\lambda(e) \notin \mathcal{O}_i$
$\Longrightarrow$ (* by definition 4.2.1 of the semantics of $\mathcal{L}$ *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} [a]_i\phi_1$

9. **Claim:** $(\mathfrak{F}, \mathfrak{c}) \vdash \phi = (\mathfrak{F}, \mathfrak{c}) \vdash \langle\mathcal{O}\rangle_i\phi_1 \in \Gamma_\pi$ implies $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \bigvee_{o \in \mathcal{O}_i} \langle\mathcal{O}\rangle_i\phi_1$:

$(\mathfrak{F}, \mathfrak{c}) \vdash \langle\mathcal{O}\rangle_i\phi_1 \in \Gamma_\pi$
$\Longrightarrow$ (* by rule (**TR 14**), prop. 5.6.14, and assumption that π is fair *)
there exists a chain $(\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ in Γ_π such that $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_o)$ and $o_m \in \mathcal{O}_i$ and for all $k < m : o_k \notin \mathcal{O}_i$
$\Longrightarrow$ (* by rules (**TR 20**) and (**TR 22**) and proposition 5.6.14 *)
there exists a chain $(\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ in Γ_π such that $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_o)$, $o_m \in \mathcal{O}_i$ and for all $k < m : o_k \notin \mathcal{O}_i$; further for all $k \leq m$: $(\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash \langle\mathcal{O}\rangle_i\phi_1 \in \Gamma_\pi$ and $(\mathfrak{F}, \mathfrak{c}_{m+1}) \vdash \phi_1 \in \Gamma_\pi$

$\Longrightarrow$ (* by lemma 6.2.5 *)
for all $k \leq m : \ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \xrightarrow{o_k} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{\mathfrak{k}+1}))$ and $o_m \in \mathcal{O}_i$ and for all $k < m : o_k \notin \mathcal{O}_i$; further for all $k \leq m : (\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash \langle\mathcal{O}\rangle_i\phi_1 \in \Gamma_\pi$ and $(\mathfrak{F}, \mathfrak{c}_{m+1}) \vdash \phi_1 \in \Gamma_\pi$

$\Longrightarrow$ (* by induction hypothesis *)
for all $k \leq m : \ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \xrightarrow{o_k} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{\mathfrak{k}+1}))$ and $o_m \in \mathcal{O}_i$ and for all $k < m : o_k \notin \mathcal{O}_i$; further for all $k \leq m : (\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash \langle\mathcal{O}\rangle_i\phi_1 \in \Gamma_\pi$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1})) \models_\mathcal{M} \phi_1$

$\Longrightarrow$ (* by definition 4.2.1 *)
for all $k \leq m : \ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \xrightarrow{o_k} \ell_\pi((\mathfrak{F}, \mathfrak{c}_{\mathfrak{k}+1}))$ with $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_o)$ and for all $k < m : o_k \notin \mathcal{O}_i$; further $\ell_\pi((\mathfrak{F}, \mathfrak{c}_m)) \models_\mathcal{M} \bigvee\limits_{o \in \mathcal{O}_i} \langle o\rangle_i\phi_1$

$\Longrightarrow$ (* by definition 3.2.3 and lemma 3.2.4 *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \bigvee_{o \in \mathcal{O}_i} \langle o\rangle_i\phi_1$

10. **Claim:** $(\mathfrak{F}, \mathfrak{c}) \vdash \phi = (\mathfrak{F}, \mathfrak{c}) \vdash [\mathcal{O}]_i\phi_1 \in \Gamma_\pi$ implies $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \bigwedge_{o \in \mathcal{O}_i}[o]_i\phi_1$:

$(\mathfrak{F}, \mathfrak{c}) \vdash [\mathcal{O}]_i\phi_1 \in \Gamma_\pi$

$\Longrightarrow$ (* by rule (**TR 14**) and prop. 5.6.14 *)
there are two possibilities:

a) there exists a finite chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ in Γ_π such that $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_o)$, $o_m \in \mathcal{O}_i$ and for all $k < m : o_k \notin \mathcal{O}_i$

$\Longrightarrow$ (* by rules (**TR 21**), (**TR 23**) and prop. 5.6.14 *)
there exists a finite chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ in Γ_π such that $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_o)$, $o_m \in \mathcal{O}_i$ and for all $k < m : o_k \notin \mathcal{O}_i$ and further $(\mathfrak{F}, \mathfrak{c}_{m+1}) \vdash \phi_1 \in \Gamma_\pi$ and for all $k \leq m : (\mathfrak{F}, \mathfrak{c}_\mathfrak{k}) \vdash [\mathcal{O}]_i\phi_1 \in \Gamma_\pi$

$\Longrightarrow$ (* by induction hypothesis *)
there exists a finite chain $\chi = (\mathfrak{F}, \mathfrak{c}_o) \xrightarrow{o_0} \ldots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ in Γ_π such that $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_o)$, $o_m \in \mathcal{O}_i$ and for all $k < m : o_k \notin \mathcal{O}_i$ and further $\ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1})) \models_\mathcal{M} \phi_1$

$\Longrightarrow$ (* by lemma 6.2.5 *)
for all $k \leq m : \ell_\pi((\mathfrak{F}, \mathfrak{c}_\mathfrak{k})) \xrightarrow{o}_k \ell_\pi((\mathfrak{F}, \mathfrak{c}_{\mathfrak{k}+1}))$, for all $k < m : o_k \notin \mathcal{O}_i$ and $o_m \in \mathcal{O}_i$ and $\ell_\pi((\mathfrak{F}, \mathfrak{c}_{m+1})) \models_\mathcal{M} \phi_1$

$\Longrightarrow$ (* by definition 4.2.1 of the semantics of L *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}_m)) \models_\mathcal{M} [\mathcal{O}]_i\phi_1$

$\Longrightarrow$ (* by definition 3.2.3 and lemma 3.2.4 *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}) \models_\mathcal{M} [\mathcal{O}]_i\phi_1$

b) for $(\mathfrak{F}, \mathfrak{c}) = (\mathfrak{F}, \mathfrak{c}_0)$ there does not exist a finite chain $\chi = (\mathfrak{F}, \mathfrak{c}_0) \xrightarrow{o_0} \dots \xrightarrow{o_m} (\mathfrak{F}, \mathfrak{c}_{m+1})$ in Γ_π such that $o_m \in \mathcal{O}_i$
$\Longrightarrow$ (* by part 1 of the construction *)
for all events $e \in E$ with $e \in E \setminus c$ it holds that $\lambda(e) \notin \mathcal{O}_i$
$\Longrightarrow$ (* by definition 4.2.1 of the semantics of L *)
$\ell_\pi((\mathfrak{F}, \mathfrak{c}) \models_\mathcal{M} [o]_i\phi_1$ for all $o \in \mathcal{O}_i$
$\Longrightarrow$ $\ell_\pi((\mathfrak{F}, \mathfrak{c})) \models_\mathcal{M} \bigwedge_{o\in\mathcal{O}_i}[o]_i\phi_1$

□

Now we have collected all requirements that are necessary for mapping ℓ_π constructed by the construction method on pages 133ff. to be a satisfying $\mathcal{L}$-embedding:

Theorem 6.2.21 *Let $\mathcal{T}(\Gamma_0^\mathcal{T})$ be an accepted regular tableau for a tableau set $\Gamma_0^\mathcal{T}$ and let π be an accepted path in the tableau.*

Further, let $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ be the structure and $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ be the mapping constructed from path π by the construction method on page 133.

Then ℓ_π is a satisfying $\mathcal{L}$-embedding.

Proof:

1. **Claim:** For each tableau formula $\mathfrak{l} \xrightarrow{a} \mathfrak{la} \in \Gamma_\pi$ it holds that $\ell_\pi(\mathfrak{l}) \xrightarrow{a} \ell_\pi(\mathfrak{la})$.
   This has been shown in lemma 6.2.5.

2. **Claim:** For each tableau formula $\mathfrak{l}R_i\mathfrak{l}' \in \Gamma_\pi$ it holds that $(\ell_\pi(\mathfrak{l}), \ell_\pi(\mathfrak{l}')) \in R_i^K$.
   This follows directly from the construction of $R_i^{K,pre}$ in steps (31) and (32) and from the construction of R_i^K in step (33).

3. **Claim:** For each tableau formula $\mathfrak{l} \vdash \phi \in \Gamma_\pi$ it holds that $\ell_\pi(\mathfrak{l}) \models_\mathcal{M} \phi$.
   This has been shown in proposition 6.2.18.

□

Finally, with theorems 6.2.17 and 6.2.21 as prerequisites it is easy to show that the tableau method is complete for the language $\mathcal{L}^c$.

From a given tableau set $\Gamma_0^\mathcal{T}$ we can always construct a tableau $\mathcal{T}(\Gamma_0^\mathcal{T})$. We have shown that we can construct a model $\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$ and a satisfying $\mathcal{L}$-embedding ℓ_π from a given accepted tableau path. This means that if $\Gamma_0^\mathcal{T}$ is not $\mathcal{L}-$satisfiable, then there does not exist an accepted tableau for $\Gamma_0^\mathcal{T}$, i.e. all tableaux $\mathcal{T}(\Gamma_0^\mathcal{T})$ are rejected.

Theorem 6.2.22 (Completeness of the tableau system) *The given tableau method is complete for $\mathcal{L}^{\mathsf{c}}$:*
Let $\Gamma_0 \subset \mathcal{L}^{\mathsf{c}}$ be a finite set of formulae and let $\Gamma_0^{\mathcal{T}}$ be the tableau set constructed from Γ_0 according to definition 5.3.3.

If Γ_0 is not satisfiable, and thus $\Gamma_0^{\mathcal{T}}$ is not $\mathcal{L}-$satisfiable, then there exists a rejected tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$.

Proof: Let $\Gamma_0^{\mathcal{T}}$ be a finite tableau set that is not $\mathcal{L}-$satisfiable and let $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ be an accepted tableau for $\Gamma_0^{\mathcal{T}}$. Then by definition 5.5.9, tableau $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ has an accepted path π.

Following the construction method on pages 6.2.1 ff, we now construct a structure $\mathcal{M} = (\mathcal{A}, \mathcal{R}^{\mathsf{K}}, \mathcal{I})$ and a mapping ℓ_π. By theorem 6.2.17, the constructed structure $\mathcal{M}$ is a model in the sense of definition 3.4.1. Further, by theorem 6.2.21, the constructed mapping $\ell_\pi : \mathfrak{label}(\Gamma_\pi) \longrightarrow |\mathcal{M}|$ is a satisfying $\mathcal{L}$-embedding for Γ_π. Since $\Gamma_0^{\mathcal{T}} \subseteq \Gamma_\pi$, the $\mathcal{L}$-embedding ℓ_π is also a satisfying $\mathcal{L}$-embedding for $\Gamma_0^{\mathcal{T}}$. Hence, $\Gamma_0^{\mathcal{T}}$ is $\mathcal{L}-$satisfiable. This is a contradiction to the assumption that $\Gamma_0^{\mathcal{T}}$ is not $\mathcal{L}-$satisfiable.

Consequently, $\mathcal{T}(\Gamma_0^{\mathcal{T}})$ is a rejected tableau.

□

Chapter 7

A Specification Example

In this chapter we give an example of specification of semantic constraints and security constraints for information systems using our logic.

In [BS97a] and [BS97b] Bleumer and Schunter have suggested a procedure for the charging and clearing process in the German health care system that allows effective control of the total remuneration of the health care system on the one side and guarantees the legitimate privacy interests for the participants on the other side. The procedure described there is based on public key cryptography.

In the German health care system, health insurances must bear all the occurring costs of the treatment while physicians are relatively free and uncontrolled in their work. Health insurances have a legitimate interest to control the overall costs while policy holders and physicians have legitimate privacy interests.

The example is divided into three sections:

1. In the first section we informally describe the system proposed by [BS97a], [BS97b] and and its semantic and security constraints.

2. In the second section we specify the proposed system logically. This section is divided into four subsections. In the first subsection, we describe the static part of the system.

   In the second subsection we specify semantic constraints about the temporal behavior of the agents. In the third subsection, we define semantic constraints about the epistemic behavior of agents. In the fourth subsection, we specify a number of security constraints that must be met by the system.

3. In the third section, we discuss advantages and limitations of our view of an information system and of the logic $\mathcal{L}$.

7.1 Privacy Oriented Clearing for the German Health Care System

The focus of the system described in [BS97a], [BS97b] is on the participants and the transaction flows in the German health care system regarding billing and payment of medical services.

Bleumer and Schunter identify four types of health care providers: 1) registered physicians, 2) pharmacies and paramedical professionals, 3) specialists and 4) hospital physicians. We do not model the complete system with all its various types of health care providers but restrict to only physicians and pharmacies:

1. *Registered physicians* (in the following abbreviated by D) are outpatient physicians, registered by compulsory health insurances and have their own independent practices. They may give medical treatment and issue prescriptions for medicaments. Their actual clearing houses are local associations of registered physicians, 'Kassenärztliche Vereinigung' (KV).

   In the described system, registered physicians form groups, such that each physician belongs to exactly one group. Each KV then serves as a group central.

2. *Pharmacists* (in the following abbreviated by F) serve patients according to the prescriptions of registered physicians. Their actual clearing houses are the health insurances.

Apart from the health care providers, we further identify the following participants of the health care system:

1. *Compulsory Health Insurances* (in the following abbreviated by H) are institutions that bear the costs of medical treatment of a policy holder. A policy holder pays an income dependent premium at regular intervals to the health insurance. Independent of the amount of premium, the compulsory health insurance must give each policy holder equal service. Compulsory health insurances issue I-certificates (insurance certificates) to policy holders, reimburse the "Kassenärztliche Vereinigung" (KV) and pharmacies.

2. *Kassenärztliche Vereinigung*, abbreviated by KV, is a local association of registered physicians that reimburses the invoices of registered physicians and gets again reimbursed by the compulsory health insurances. On behalf of the health insurances, the KVs register physicians and assign them to a group.

3. *Policy holders* (in the following abbreviated by P) pay income related premiums to the health insurances. They receive insurance certificates (I-certificates) from their health insurance which enable them to claim services from health care providers.

In our example system, we assume one health insurance H, one KV, one pharmacy F, two (registered) physicians D_1 and D_2 and two policy holders P_1 and P_2. Further, we assume that both physicians, D_1 and D_2 belong to the same group G, the central of which is the KV.

The system defined by Bleumer and Schunter can be divided into two phases, an initialization phase and a treatment and billing phase. The initialization phase includes the following actions:

- The policy holder, the pharmacy and the physicians register at the health insurance.
- The health insurance issues insurance certificates to its registered policy holders.
- As already mentioned above, the system proposed by Bleumer and Schunter is based on public key cryptography. The necessary keys are generated in the initialization phase:
  - The health insurance generates a pair of cryptographic keys the secret key of which it uses for issuing insurance certificates to its policy holders. The corresponding public key is used by physicians to validate insurance certificates of their patients.
  - Each physician generates two pair of cryptographic keys, one for signing invoices, the other one for issuing prescriptions to her patients.
  - To perpetuate anonymity, the physicians do not actually sign invoices and issue prescriptions with their individual keys but form groups and use group keys for signing invoices and issuing prescriptions. Therefor, they send their public keys to the KV which serves as a group central and generates pairs of group keys for signing invoices and issuing prescriptions. The physicians will later use the public group keys together with their individual keys from which the group keys are generated for signing invoices and issuing prescriptions. Using the secret group key, only the group central, the KV, can later determine the identity of the issuer of a prescription or of the signer of an invoice. The health insurance will use the public group key to verify, that the issuer of the invoice belongs to a group of registered physicians.

  - The pharmacy generates a pair of cryptographic keys for signing invoices that it sends to the health insurance.

The main phase is the treatment and billing phase. This phase includes the following actions:

- Policy holders request treatment from physicians and medicaments from pharmacies.
- Physicians issue prescriptions to their patients and send invoices to the health insurance (via the KV).
- The pharmacy delivers medicaments to policy holders and sends invoices to the health insurance.
- The health insurance clears and verifies invoices.
- The KV helps to clear invoices and identifies signers of invoices or issuers of prescriptions in case the health insurance suspects misuse.

In the next two sections we describe these two phases in more detail.

7.1.1 Initialization Phase

Some necessary initialization operations are performed in the beginning of the behavior of the system. These operations are

- registration of pharmacies and physicians,
- generation and publication of cryptographic keys,
- issuing of I-certificates.

Initialization of the system We assume as initial situation that there no connections yet between policy holders, physicians, the health insurance, the KV and the pharmacy, i.e. the policy holders are not registered at the health insurance and don't hold any insurance certificates, yet, the physicians and the pharmacy are not registered yet either, etc.

Registration of physicians and pharmacies In order to participate in the health care billing and payment system, in particular to claim reimbursement from the health insurance for provided treatment or delivered medicaments, the physicians and the pharmacy must be registered at the health insurance. We require that the registration at the health insurance is the first operation of each physician and of the pharmacy. The registration operations are denoted by $\text{register}_{(\mathsf{D}_1,\mathsf{KV},\mathsf{H})}$, $\text{register}_{(\mathsf{D}_2,\mathsf{KV},\mathsf{H})}$, $\text{register}_{(\mathsf{F},\mathsf{H})}$. Note, that we assume that the KV needs to participate in the registration operation of each physician. This is indicated by the indexes of the registration operations.

Generation and publication of cryptographic keys

- *Health insurance* H*:*
  For issuing I-certificates (insurance certificates) to its policy holders, the health insurance H needs a pair of cryptographic keys. We will denote the public key of the health insurance by $\mathtt{pk}_\mathsf{H}$ and the secret key by $\mathtt{sk}_\mathsf{H}$. The operation for generating the keys for health insurance H is gen_H. Using its secret key $\mathtt{sk}_\mathsf{H}$, the health insurance H issues I-certificates to its policy holders. The public key $\mathtt{pk}_\mathsf{H}$ is needed by physicians and pharmacies for validating I-certificates shown to them by policy holders. The health insurance thus publishes its public key $\mathtt{pk}_\mathsf{H}$ to each registered physician D and to each registered pharmacy F by operations $\text{pub}_{(\mathsf{H},\mathsf{D})}(\mathtt{pk}_\mathsf{H})$ and $\text{pub}_{(\mathsf{H},\mathsf{F})}(\mathtt{pk}_\mathsf{H})$ respectively.

- *Pharmacy* F*:*
  The pharmacy F generates a pair $(\mathtt{sk}_\mathsf{F}, \mathtt{pk}_\mathsf{F})$ of cryptographic keys for signing invoices. The operation for generating the keys for the pharmacy is called gen_F. The pharmacy will use its secret key for signing invoices, the health insurance uses the pharmacy's public key for verifying invoices of the pharmacy. Pharmacy F publishes its public key $\mathtt{pk}_\mathsf{F}$ to the health insurance H by operation $\text{pub}_{(\mathsf{F},\mathsf{H})}(\mathtt{pk}_\mathsf{F})$.

- *Physician* D *and "Kassenärztliche Vereinigung"* KV*:*
  By performing operation $\text{gen}^{\mathtt{i}}_{\mathsf{D}}$, each physician D generates a pair of individual cryptographic keys $(\mathtt{pi}_\mathsf{D}, \mathtt{si}_\mathsf{D})$ that are used for signing and verifying signatures on invoices. The physician sends her individual public key $\mathtt{pi}_\mathsf{D}$ to the KV by operation $\text{pub}_{(\mathsf{D},\mathsf{KV})}(\mathtt{pi}_\mathsf{D})$. The KV uses the public individual keys to generate a pair of group keys $(\mathtt{pi}_\mathsf{G}, \mathtt{si}_\mathsf{G})$ by performing operation $\text{gen}^{\mathtt{i}}_{\mathsf{KV}}$. The public group key will be published to each physician D of group G by operation $\text{pub}_{(\mathsf{KV},\mathsf{D})}(\mathtt{pi}_\mathsf{G})$ and the the health insurance by operation $\text{pub}_{(\mathsf{KV},\mathsf{H})}(\mathtt{pi}_\mathsf{G})$.

  D will use her public keys $\mathtt{pi}_\mathsf{G}$ and $\mathtt{pi}_\mathsf{D}$ as well as her secret key $\mathtt{si}_\mathsf{G}$ for signing invoices that she sends to the KV that forwards them to the health insurance. The health insurance H needs the public group key $\mathtt{pi}_\mathsf{G}$ to verify group signatures on invoices. While the health insurance can only determine the group to which the signer of an invoice belongs, the KV uses the public and the secret group keys $\mathtt{pi}_\mathsf{G}$ and $\mathtt{si}_\mathsf{G}$ to determine the identity of the signer.

A similar setting will be used for issuing and verifying prescriptions: By performing operation $\mathrm{gen}^\mathrm{p}_\mathsf{D}$, each physician D generates a pair of cryptographic prescription keys $(\mathtt{pp}_\mathsf{D}, \mathtt{sp}_\mathsf{D})$. She publishes her public key to the KV by performing operation $\mathrm{pub}_{(\mathsf{D},\mathsf{KV})}(\mathtt{pp}_\mathsf{D})$. From the set of public individual prescription keys, the KV generates a pair of group keys for prescriptions $(\mathtt{sp}_\mathsf{G}, \mathtt{pp}_\mathsf{G})$ performing operation $\mathrm{gen}^\mathrm{p}_\mathsf{KV}$. The public group key will be published to all physicians, to the health insurance and to the pharmacy. A physician uses her public and secret individual prescription keys $\mathtt{sp}_\mathsf{D}$ and $\mathtt{pp}_\mathsf{D}$ as well as the public group key $\mathtt{pp}_\mathsf{G}$ for issuing prescriptions. The pharmacy and the health insurance use the public group key to verify the prescriptions. While the pharmacy and the health insurance can only determine the group of the issuer of an prescription, the KV uses the secret group key $\mathtt{sp}_\mathsf{G}$ to determine the identity of the issuer of a prescription.

- *Policy holders:*
  In our setting, policy holders do not need to generate any cryptographic keys.

Issuing of insurance certificates The health insurance H issues a batch of I-certificates to each of its policy holders P using its secret key $\mathtt{sk}_\mathsf{H}$. The issuing operation $\mathrm{issue}^\mathrm{I}_{(\mathsf{H},\mathsf{P})}$ is performed jointly by health insurance H and policy holder P. Each resulting I-certificate $\mathrm{I}(\mathtt{ps})$ contains information about the issuing health insurance and the pseudonym, on which the I-certificate was issued. It ensures the following properties:

- Each I-certificate allows to determine the issuing health insurance H but not the identity of the policy holder P.
- Each I-certificate can be used only once. Using it a second time reveals the policy holder's identity to the health insurance. Thus, obviously each pseudonym must be unique.

7.1.2 Treatment and Billing Phase

The treatment and billing phase is the main phase of the system and starts after the initialization steps are performed. This phase contains the following processes:

Treatment A policy holder P requests treatment from a physician D by showing the physician one of her (unused) I-certificates, e.g. $\mathtt{I}(\mathtt{ps})$ for some pseudonym $\mathtt{ps}$. The showing operation is named $\mathrm{show}_{(\mathsf{P},\mathsf{D})}(\mathtt{I}(\mathtt{ps}))$. Physician D needs the public key $\mathtt{pk}_{\mathsf{H}}$ of the health insurance for verifying the shown I-certificate. After showing the I-certificate, the policy holder is temporarily (i.e. for the time of treatment) known to physician D by pseudonym $\mathtt{ps}$. From the I-certificate, the physician knows that policy holder P holds an I-certificate issued by health insurance H on pseudonym $\mathtt{ps}$. Further, she holds a transcript $\mathtt{t}^{\mathtt{I}}(\mathtt{ps})$ of the insurance certificate. This transcript contains the pseudonym $\mathtt{ps}$ and will be enclosed in the invoice, when the physician claims reimbursement from the health insurance via the KV for treating the policy holder. Eventually, after treatment has finished, the physician may forget the correspondence $(\mathsf{P}, \mathtt{ps})$.

The physician may provide some treatment herself and additionally issue a prescription $\mathsf{M}(\mathsf{D}, \mathtt{ps}')$ to the policy holder[1]. For issuing the prescription, physician D uses her secret prescription key $\mathtt{sp}_{\mathsf{D}}$. The prescription must be issued on a pseudonym $\mathtt{ps}'$ under which the the policy holder will request the prescribed medicaments at a pharmacy. In the issuing process, the policy holder adds the pseudonym of a fresh I-certificate. There are basically two ways of modeling the issuing process: Either we assume that the physician and the policy holder agree on the pseudonym under which the policy holder can later on claim the medicaments or the pseudonym is kept secret from the physician so that only the policy holder knows under which pseudonym she will later on claim the prescribed medicament. In the first case, we could introduce one issue operation for each pseudonym. In our example, we prefer the latter case, in which the physician shall not know the pseudonym for which a prescription is issued. We introduce three operations for the issuing process:

1. By performing operation $\mathrm{s\_issue}_{(\mathsf{D},\mathsf{P})}$ physician D and policy holder P jointly start the issuing process.
2. Then the policy holder chooses one of his fresh I-certificates, e.g. $\mathtt{I}(\mathtt{ps}')$ and adds the pseudonym $\mathtt{ps}'$ to the prescription by operation $\mathrm{add}_{\mathsf{P}}(\mathtt{ps}')$. Since the choice of the pseudonym must be kept secret from the physician, this operation is performed by policy holder P only.
3. Finally physician D and policy holder P together complete the issuing process by jointly performing operation $\mathrm{e\_issue}_{(\mathsf{D},\mathsf{P})}$.

After the issuing process, the policy holder holds the medical prescription $\mathsf{M}(\mathsf{D}, \mathtt{ps}')$ which has as first parameter the issuing physician and as second parameter the

[1] In this example we abstract from the actually prescribed medicaments, they are not modeled in the prescription.

chosen pseudonym. Note, that these three operations always need to be performed in a row. No other operation of physician D and policy holder P may be performed in between.

Showing a medical prescription $\mathsf{M}(\mathsf{D}, \mathtt{ps}')$ and I-certificate $\mathtt{I}(\mathtt{ps})$ to pharmacy F, policy holder P requests the prescribed medicament from the pharmacy. (The pharmacy is supposed to only deliver the requested medicament, if both the I-certificate and the prescription are issued for the same pseudonym and will thus only accept apposite certificates and prescriptions.) The performance of the show operation $\mathrm{gShow}_{(\mathsf{P},\mathsf{F})}(\mathtt{ps}, \mathsf{D}, \mathtt{ps}')$ generates a transcript $\mathtt{t}^{\mathsf{M}}(\mathtt{ps}, \mathsf{D}, \mathtt{ps}')$.

Clearing – Physicians Physicians get reimbursed by the health insurance via the KV.

For billing the health insurance, physician D includes a transcript $\mathtt{t}^{\mathtt{I}}(\mathtt{ps})$ of the appropriate I-certificate in the invoice and signs the invoice with her secret individual invoice key $\mathtt{si}_{\mathsf{D}}$, her individual public invoice key $\mathtt{pi}_{\mathsf{D}}$ and the public invoice key $\mathtt{pi}_{\mathsf{G}}$ of group G. To do so, she performs operation $\mathrm{sign}_{\mathsf{D}}(\mathrm{inv}(\mathtt{ps}))$ by which she gets the signed invoice $\sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathtt{t}^{\mathtt{I}}(\mathtt{ps}))$. The physician then deposits the signed invoice at the KV by operation $\mathrm{dep}_{(\mathsf{D},\mathsf{KV})}(\sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathtt{t}^{\mathtt{I}}(\mathtt{ps})))$. The KV forwards it to the health insurance H by operation $\mathrm{dep}_{(\mathsf{KV},\mathsf{H})}(\sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathtt{t}^{\mathtt{I}}(\mathtt{ps})))$. The signed invoice takes the indirection over the KV instead of being sent directly from the physician D to the health insurance H, because the health insurance should not get to know the signer unless the insurance suspects a misuse. If this is not the case, the KV is used as some kind of anonymizing channel. In case the health insurance suspects misuse, the health insurance requests the KV to determine the identity of the signer with the help the secret group key $\mathtt{si}_{\mathsf{G}}$.

The fact that the invoicing party belongs to group G is denoted as $\mathrm{group}(\mathsf{G}, \sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathtt{t}^{\mathtt{I}}(\mathtt{ps})))$. The fact that physician D has issued an invoice σ is denoted as $\mathrm{id}(\sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathtt{t}^{\mathtt{I}}(\mathtt{ps})))$

If the health insurance does not request the KV to reanonymize the signer of an invoice, it must eventually reimburse the physician D via the KV. This is done by operations $\mathrm{reimb}_{(\mathsf{H},\mathsf{KV})}(\sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathtt{t}^{\mathtt{I}}(\mathtt{ps})))$ and $\mathrm{reimb}_{(\mathsf{KV},\mathsf{D})}(\sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathtt{t}^{\mathtt{I}}(\mathtt{ps})))$.

To ensure that each pseudonym is used only once, the health insurance H remembers pseudonyms, for which she already received an invoices. If a pseudonym is used twice, then the health insurance will know the identity of the policy holder that carried the pseudonym.

Clearing – Pharmacies The pharmacy clears its costs directly with the health insurance. The invoice of a pharmacy must include a transcript of both the prescription and the corresponding I-certificate. The pharmacy signs its invoice by performing operation $\mathrm{sign}_{\mathsf{F}}(\mathrm{inv}(t^{\mathsf{M}}(ps, D, ps')))$. After signing the invoice, the pharmacy holds the signed invoice $\sigma_{\mathsf{F}}^{\mathrm{inv}}(t^{\mathrm{I}}(t^{\mathsf{M}}(ps, D, ps')))$. To get reimbursed, the pharmacy sends the invoice to the health insurance by operation $\mathrm{dep}_{(\mathsf{F},\mathsf{H})}(\sigma_{\mathsf{F}}^{\mathrm{inv}}(t^{\mathsf{M}}(ps, D, ps')))$. The health insurance verifies the signature and "accepts" the invoice if the signature is OK and the pseudonyms of both transcripts match. As before, to ensure that each pseudonym is used only once, the health insurance H checks the pseudonym. If the pseudonym has already been used before the health insurance will automatically know the identity of the policy holder.

After receiving a signed invoice $\sigma_{\mathsf{F}}^{\mathrm{inv}}(t^{\mathsf{M}}(ps, D, ps'))$ from the pharmacy, the health insurance will eventually reimburse the pharmacy by operation $\mathrm{reimb}_{(\mathsf{H},\mathsf{F})}(\sigma_{\mathsf{F}}^{\mathrm{inv}}(t^{\mathsf{M}}(ps, D, ps')))$.

7.2 Logical Specification of the Billing and Clearing Scheme

This section concentrates on the logical specification of semantic constraints and security constraints of the health care clearing system.

In the first subsection we define the static part of the system (refer to section 3.1). We view the participants of the health care system, i.e. physicians, policy holders, health insurances, etc. as a set of agents Ag, and define the distributed set of propositions $\tilde{\mathcal{P}}$ and the distributed set of operations $\tilde{\mathcal{O}}$.

In the second subsection we logically specify the semantic constraints concerning the temporal behavior of the system. The operations do not carry any semantics themselves, they are only syntactic elements. We will define their meaning over the interpretation of the relevant propositions. We do this in the logic itself. Formulae restrict the set of runs that we accept as appropriate.

In the third subsection we will give examples of semantic constraints about the epistemic behavior. We specify examples of how operations effect the knowledge and belief of participating agents.

In the fourth subsection we will specify a number of security constraints that should be met by the system.

To make the presentation (hopefully) easier to understand we make the following abstractions. Note, that the logic $\mathcal{L}$ can be used to specify security constraints without these abstractions.

- We assume that a physician does not accept the same I-certificate of a policy holder twice.

- We do not distinguish between multiple submissions of the same invoice by a physician (via the KV) or a pharmacy.
- We assume that each actor generates only one pair of keys for each purpose. The created keys will then remain stored at the respective actors forever. This ensures that we do not have to deal with outdated keys.

In section 7.3 we will discuss which changes we would have to make if we dropped these abstractions.

7.2.1 Static Part of the System

In the previous section we have listed various participants of the health care system. For our example we assume one health insurance H, two registered physicians D_1 and D_2, one local association of registered physicians KV, two policy holders P_1 and P_2 and one pharmacy F.

We formally define the set of agent as follows:

$$Ag := \{H, D_1, D_2, KV, P_1, P_2, F\}$$

Local Propositions

Each agent is equipped with a set of local propositions. For (hopefully) better reading, we use propositions with a syntactic structure.

In the previous section we have already described a number of facts that have to be modeled as propositions. We first list all necessary facts and briefly describe their (informal) meaning. Then, we will model the propositions as follows: Each proposition 'fact@agent' is constructed in such a way that the first part of the proposition describes a fact and the second part tells us, at which agent this fact is stored, for example $\mathtt{pi}_G$@KV means that the fact $\mathtt{pi}_G$ (meaning "the public invoice key of group G) is stored at the KV.

We now list the facts that can be combined to propositions according to the construct 'fact@agent' by concatenating them with the agent they are stored at.

Pseudonyms: $\{\spadesuit, \diamondsuit, \clubsuit, \heartsuit\}$

Pseudonyms do not occur directly as facts in the specification. However, insurance certificates are issued on pseudonyms, policy holders are known to physicians by their pseudonyms and prescriptions are also issued on pseudonyms. So, we will use pseudonyms in the construction of other facts.

Cryptographic keys: $\mathsf{pk_H}$, $\mathsf{sk_H}$, $\mathsf{pp_G}$, $\mathsf{sp_G}$, $\mathsf{pi_G}$, $\mathsf{si_G}$, $\mathsf{pp_{D_1}}$, $\mathsf{sp_{D_1}}$, $\mathsf{pi_{D_1}}$, $\mathsf{si_{D_1}}$, $\mathsf{pp_{D_2}}$, $\mathsf{sp_{D_2}}$, $\mathsf{pi_{D_2}}$, $\mathsf{si_{D_2}}$, $\mathsf{pk_F}$, $\mathsf{sk_F}$

As described in section 7.1.1, agents generate pairs of cryptographic keys in the initialization phase. We denote the generated keys as follows:

- the health insurance generates a pair of keys which we denote by $\mathsf{pk_H}$ / $\mathsf{sk_H}$ (for the **p**ublic / **s**ecret **k**ey of health insurance **H**).
- each physician $\mathsf{D} \in \{\mathsf{D_1}, \mathsf{D_2}\}$ generates two pairs of cryptographic keys. We denote these individual keys by
  - $\mathsf{pp_D}$ / $\mathsf{sp_D}$ (for the **p**ublic / **s**ecret **p**rescription key of physician **D**) as well as
  - $\mathsf{pi_D}$ / $\mathsf{si_D}$ (for the **p**ublic / **s**ecret **i**ssuing key of physician **D**).
- the KV generates two pairs of group keys from the individual keys of the physicians. We denote these keys by
  - $\mathsf{pp_G}$ / $\mathsf{sp_G}$ (for the **p**ublic / **s**ecret **p**rescription key of group **G**), as well as
  - $\mathsf{pi_G}$ / $\mathsf{si_G}$ (for the **p**ublic / **s**ecret **i**ssuing key of group **G**).
- the pharmacy generates a pair of cryptographic keys for signing its invoices. We denote these keys by
  $\mathsf{pk_F}$ / $\mathsf{sk_F}$ for the **p**ublic / **s**ecret **k**ey of pharmacy **F**.

I-certificates: $\mathrm{I}(\mathsf{ps})$

An I-certificate $\mathrm{I}(\mathsf{ps})$ entitles a policy holder to request treatment from a physician under pseudonym ps and to claim medicaments from a pharmacy according to a prescription that is issued for pseudonym ps.

Used I-certificates: $\mathrm{I}^{\mathsf{u}}(\mathsf{ps})$

An I-certificate issued on pseudonym ps has already been used.

Double-show: $\mathsf{ds}(\mathsf{ps})$

An I-certificate issued on pseudonym ps has been double-shown.

Medical prescriptions: $\mathsf{M}(\mathsf{D}, \mathsf{ps})$

While a policy holder is under treatment at a physician, the physician D may issue a medical prescription $\mathsf{M}(\mathsf{D}, \mathsf{ps})$ to a policy holder. The prescription is issued on a pseudonym and has to the shown to a pharmacy together with the appropriate I-certificate, issued on the same pseudonym.

Transcripts from I-certificates: $\mathsf{t}^{\mathrm{I}}(\mathsf{ps})$

After a policy holder has shown an I- certificate issued on pseudonym ps to a physician D, the physician holds a transcript $\mathsf{t}^{\mathrm{I}}(\mathsf{ps})$ of the corresponding I-certificate.

Patient P has pseudonym ps: $(\mathsf{P}, \mathsf{ps})$
Policy holder P is associated with pseudonym ps and is currently under treatment.

Health insurance H has issued an I-certificate on pseudonym ps : $(\mathsf{H}, \mathsf{ps})$
Health insurance H has issued an I-certificate on pseudonym ps.

Transcripts from medical prescriptions: $\mathsf{t}^{\mathsf{M}}(\mathsf{ps}, \mathsf{D}, \mathsf{ps}')$
After a policy holder has shown an I-certificate $\mathsf{I}(\mathsf{ps})$ and a prescription $\mathsf{M}(\mathsf{D}, \mathsf{ps}')$ to the pharmacist, she holds a transcript $\mathsf{t}^{\mathsf{M}}(\mathsf{ps}, \mathsf{D}, \mathsf{ps}')$ of the shown certificates.

Signed invoices: $\sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathsf{t}^{\mathsf{I}}(\mathsf{ps}))$ and $\sigma_{\mathsf{F}}^{\mathrm{inv}}(\mathsf{t}^{\mathsf{M}}(\mathsf{ps}, \mathsf{D}, \mathsf{ps}'))$
Physicians and the pharmacy enclose a transcript of a shown certificate in an invoice and sign the invoice.

Identity of a policy holder: $\mathsf{id}(\mathsf{P}, \mathsf{ps})$
The policy holder that has double-shown an I-certificate issued on pseudonym ps is policy holder P.

Identity of the signer of an invoice: $\mathsf{id}(\mathsf{D}, \sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathsf{t}^{\mathsf{I}}(\mathsf{ps})))$
The physician that has issued the signed invoice $\sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathsf{t}^{\mathsf{I}}(\mathsf{ps}))$ is physician D.

Group of the signer of an invoice: $\mathrm{group}(\mathsf{G}, \sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathsf{t}^{\mathsf{I}}(\mathsf{ps})))$
The signer of the invoice $\mathrm{group}(\mathsf{G}, \sigma_{\mathsf{D}}^{\mathrm{inv}}(\mathsf{t}^{\mathsf{I}}(\mathsf{ps})))$ belongs to the group G of physicians.

Identity of the issuer of a prescription: $\mathsf{id}(\mathsf{D}, \sigma_{\mathsf{F}}^{\mathrm{inv}}(\mathsf{t}^{\mathsf{M}}(\mathsf{ps}, \mathsf{D}, \mathsf{ps}')))$
Each signed invoice of a pharmacy encloses a transcript of a prescription issued by some physician. The identity of the issuer of this prescription is D.

Group of the issuer of a prescription: $\mathrm{group}(\mathsf{G}, \sigma_{\mathsf{F}}^{\mathrm{inv}}(\mathsf{t}^{\mathsf{M}}(\mathsf{ps}, \mathsf{D}, \mathsf{ps}')))$
Each signed invoice of a pharmacy encloses a transcript of a prescription issued by some physician. The issuer of this prescription belongs to group G.

Formally, we define the set of facts for each agent of the system as follows:

- *Facts for health insurance* H*:*

$$\begin{array}{lll}
\mathbf{fact}(\mathsf{H}) & ::= & \mathbf{key} \mid \mathbf{invoice} \mid \mathbf{identity} \mid \mathbf{group} \\
\mathbf{key} & ::= & sk_H \mid pk_H \mid pk_F \mid pp_G \mid pi_G \mid pp_{D_1} \mid pi_{D_1} \mid pp_{D_2} \mid pi_{D_2} \\
\mathbf{invoice} & ::= & \sigma_D^{inv}(t^I(\mathbf{ps})) \mid \sigma_F^{inv}(t^M(\mathbf{ps}, D_1, \mathbf{ps})) \mid \sigma_F^{inv}(t^M(\mathbf{ps}, D_2, \mathbf{ps})) \\
\mathbf{ps} & ::= & \spadesuit \mid \diamondsuit \mid \heartsuit \mid \clubsuit \\
\mathbf{identity} & ::= & id(P_1, ps) \mid id(P_2, ps) \mid id(D_1, \sigma_{D_1}^{inv}(t^I(\mathbf{ps}))) \mid id(D_2, \sigma_{D_2}^{inv}(t^I(\mathbf{ps}))) \\
 & & \mid id(D_1, \sigma_F^{inv}(t^M(\mathbf{ps}, D_1, \mathbf{ps}))) \mid id(D_2, \sigma_F^{inv}(t^M(\mathbf{ps}, D_2, \mathbf{ps}))) \\
\mathbf{group} & ::= & \mathrm{group}(G, \sigma_D^{inv}(t^I(\mathbf{ps})))
\end{array}$$

- *Facts of policy holder* $P \in \{P_1, P_2\}$*:*

$$\begin{array}{lll}
\mathbf{fact}(P) & ::= & \mathbf{certificate} \mid \mathbf{used} \mid \mathbf{double\text{-}show} \\
\mathbf{certificate} & ::= & I(ps) \mid M(D_1, ps) \mid M(D_2, ps) \\
\mathbf{ps} & ::= & \spadesuit \mid \diamondsuit \mid \heartsuit \mid \clubsuit \\
\mathbf{used} & ::= & I^u(\mathbf{ps}) \\
\mathbf{double\text{-}show} & ::= & ds(\mathbf{ps})
\end{array}$$

- *Facts of the pharmacy* F*:*

$$\begin{array}{lll}
\mathbf{fact}(F) & ::= & \mathbf{key} \mid \mathbf{transcript} \mid \mathbf{invoice} \\
\mathbf{key} & ::= & pk_F \mid sk_F \mid pk_H \mid pi_{D_1} \mid pi_{D_2} \mid pp_{D_1} \mid pp_{D_2} \mid pi_G \mid pp_G \mid \\
\mathbf{transcript} & ::= & t^M(\mathbf{ps}, D_1, \mathbf{ps}) \mid t^M(\mathbf{ps}, D_2, \mathbf{ps}) \\
\mathbf{invoice} & ::= & \sigma_F^{inv}(\mathbf{transcript}) \\
\mathbf{ps} & ::= & \spadesuit \mid \diamondsuit \mid \heartsuit \mid \clubsuit
\end{array}$$

- *Facts for the “Kassenärztliche Vereinigung”:*

$$\begin{array}{lll}
\mathbf{fact}(KV) & ::= & \mathbf{invoice} \mid \mathbf{key} \\
\mathbf{invoice} & ::= & \sigma_{D_1}^{inv}(\mathbf{transcript}) \mid \sigma_{D_2}^{inv}(\mathbf{transcript}) \\
\mathbf{transcript} & ::= & t^I(\mathbf{ps}) \\
\mathbf{ps} & ::= & \spadesuit \mid \diamondsuit \mid \heartsuit \mid \clubsuit \\
\mathbf{key} & ::= & pi_{D_1} \mid pi'_{D_2} \mid pp_{D_1} \mid pp'_{D_2} \mid pi_G \mid si_G \mid pp_G \mid sp_G \mid pk_H \mid pk_F
\end{array}$$

- *Facts of physician* $D \in \{D_1, D_2\}$*:*

$$
\begin{array}{lll}
\textbf{fact}(D) & ::= & \textbf{key} \mid \textbf{transcript} \mid \textbf{invoice} \mid \textbf{treatment} \\
\textbf{key} & ::= & pi_D \mid si_D \mid pp_D \mid sp_D \mid pk_H\ pi_G \mid pp_G \mid pk_F \\
\textbf{transcript} & ::= & t^I(\textbf{ps}) \\
\textbf{ps} & ::= & \spadesuit \mid \diamondsuit \mid \heartsuit \mid \clubsuit \\
\textbf{invoice} & ::= & \sigma_D^{inv}(\textbf{transcript}) \\
\textbf{treatment} & ::= & (P_1, \textbf{ps}) \mid (\mathbf{P_2}, \textbf{ps}) \mid (\mathbf{H}, \textbf{ps})
\end{array}
$$

A proposition consists of a fact plus the name of the agent to whose set of propositions it belongs. This supplement ensures that all sets of propositions are disjoint. Intuitively, the supplement tells us, where (at which location) the fact is stored.

The set of propositions for agent $i \in Ag$ is then defined as follows:

$$\mathcal{P}_i := \{\text{fact(agent)@agent} \mid \text{agent} = i\}$$

The distributed set of propositions is defined as follows:

$$\tilde{\mathcal{P}} := (\mathcal{P}_H, \mathcal{P}_{P_1}, \mathcal{P}_{P_2}, \mathcal{P}_F, \mathcal{P}_{KV}, \mathcal{P}_{D_1}, \mathcal{P}_{D_2})$$

Operations

In the following, we specify the set of operations of each agent. Operations that are in the operation alphabet of more than one agent are to be performed as a joint operation together by all agents that have the operation in their alphabet.

- *Operations of health insurance* H*:*

$$
\begin{array}{lll}
\mathcal{O}_H ::= & \{ & \text{register}_{(F,H)},\ \text{register}_{(\mathbf{D},KV,H)}, \\
& & \text{gen}_H\ ,\ \text{pub}_{(H,\mathbf{ag})}(pk_H) \\
& & \text{issue}^I_{(H,P_1)},\ \text{issue}^I_{(H,P_2)}, \\
& & \text{dep}_{(KV,H)}(\sigma_{\mathbf{D}}^{inv}(t^I(\mathbf{ps}))) \\
& & \text{dep}_{(F,H)}(\sigma_F^{inv}(t^M(\mathbf{ps}, \mathbf{D}, \mathbf{ps'}))), \\
& & \text{req}_{(H,KV)}(\sigma_{\mathbf{D}}^{inv}(t^I(\mathbf{ps}))), \\
& & \text{reimb}_{(H,KV)}(\sigma_{\mathbf{D}}^{inv}(t^I(\mathbf{ps}))) \\
& & \text{reimb}_{(H,F)}(\sigma_F^{inv}(t^M(\mathbf{ps}, \mathbf{D}, \mathbf{ps'}))), \\
& & \mid \mathbf{ag} \in \{P_1, P_2, D_1, D_2, KV, F\}, \mathbf{D} \in \{D_1, D_2\}, \mathbf{ps},\ \mathbf{ps'} \in \{\spadesuit, \diamondsuit, \clubsuit, \heartsuit\} \\
& & \}
\end{array}
$$

- *Operations of the policy holder* P*:*

$$\mathcal{O}_{\mathsf{P}} ::= \{ \ \mathrm{issue}^{\mathrm{I}}_{(\mathsf{H},\mathsf{P})}, \\ \mathrm{s\_issue}_{(\mathbf{D},\mathsf{P})}, \ \mathrm{e\_issue}_{(\mathbf{D},\mathsf{P})}, \ \mathrm{add}_{\mathsf{P}}(\mathbf{ps}) \\ \mathrm{show}_{(\mathsf{P},\mathbf{D})}(\mathbf{ps}), \mathrm{gShow}_{(\mathsf{P},\mathsf{F})}(\mathbf{ps}, \mathbf{D}, \mathbf{ps'}), \ \mathrm{deliver}_{\mathsf{F},\mathsf{P}}(\mathbf{ps}), \\ \mid \ \mathbf{D} \in \{\mathsf{D}_1, \mathsf{D}_2\}, \mathbf{ps}, \ \mathbf{ps'} \in \{\spadesuit, \diamondsuit, \clubsuit, \heartsuit\} \\ \}$$

- *Operations of the pharmacy* F*:*

$$\mathcal{O}_{\mathsf{F}} ::= \{ \ \mathrm{register}_{(\mathsf{F},\mathsf{H})}, \ \mathrm{gen}_{\mathsf{F}} \ , \\ \mathrm{pub}_{(\mathsf{H},\mathsf{F})}(\mathsf{pk}_{\mathsf{H}}), \ \mathrm{pub}_{(\mathsf{F},\mathsf{H})}(\mathsf{pk}_{\mathsf{F}}), \\ \mathrm{gShow}_{(\mathbf{P},\mathsf{F})}(\mathbf{ps}, \mathbf{D}, \mathbf{ps'}) \\ \mathrm{sign}_{\mathsf{F}}(\mathrm{inv}(\mathsf{t}^{\mathsf{M}}(\mathbf{ps}, \mathbf{D}, \mathbf{ps'}))) \\ \mathrm{dep}_{(\mathsf{F},\mathsf{H})}(\sigma^{\mathrm{inv}}_{\mathsf{F}}(\mathsf{t}^{\mathsf{M}}(\mathbf{ps}, \mathbf{D}, \mathbf{ps'}))) \\ \mathrm{reimb}_{(\mathsf{H},\mathsf{F})}(\sigma^{\mathrm{inv}}_{\mathsf{F}}(\mathsf{t}^{\mathsf{M}}(\mathbf{ps}, \mathbf{D}, \mathbf{ps'}))) \\ \mid \ \mathbf{P} \in \{\mathsf{P}_1, \mathsf{P}_2\}, \mathbf{D} \in \{\mathsf{D}_1, \mathsf{D}_2\}, \mathbf{ps}, \mathbf{ps'} \in \{\spadesuit, \diamondsuit, \clubsuit, \heartsuit\} \\ \}$$

- *Operations of the local association of registered physicians* KV*:*

$$\mathcal{O}_{\mathsf{KV}} ::= \{ \ \mathrm{register}_{(\mathbf{D},\mathsf{KV},\mathsf{H})} \\ \mathrm{gen}^{\mathsf{i}}_{\mathsf{KV}}, \ \mathrm{gen}^{\mathsf{p}}_{\mathsf{KV}}, \\ \mathrm{pub}_{(\mathbf{D},\mathsf{KV})}(\mathsf{pi}_{\mathsf{D}}), \ \mathrm{pub}_{(\mathbf{D},\mathsf{KV})}(\mathsf{pi}'_{\mathsf{D}}), \\ \mathrm{pub}_{(\mathsf{KV},\mathsf{H})}(\mathsf{pg}_{\mathsf{G}}), \ \mathrm{pub}_{(\mathsf{KV},\mathsf{H})}(\mathsf{pg}'_{\mathsf{G}}), \\ \mathrm{dep}_{(\mathsf{KV},\mathsf{H})}(\sigma^{\mathrm{inv}}_{\mathbf{D}}(\mathsf{t}^{\mathrm{I}}(\mathbf{ps}))), \\ \mathrm{dep}_{(\mathbf{D},\mathsf{KV})}(\sigma^{\mathrm{inv}}_{\mathbf{D}}(\mathsf{t}^{\mathrm{I}}(\mathbf{ps}))), \\ \mathrm{req}_{(\mathsf{H},\mathsf{KV})}(\sigma^{\mathrm{inv}}_{\mathbf{D}}(\mathsf{t}^{\mathrm{I}}(\mathbf{ps}))), \\ \mathrm{reimb}_{(\mathsf{H},\mathsf{KV})}(\sigma^{\mathrm{inv}}_{\mathbf{D}}(\mathsf{t}^{\mathrm{I}}(\mathbf{ps}))), \\ \mathrm{reimb}_{(\mathsf{KV},\mathbf{D})}(\sigma^{\mathrm{inv}}_{\mathbf{D}}(\mathsf{t}^{\mathrm{I}}(\mathbf{ps}))) \\ \mid \ \mathbf{D} \in \{\mathsf{D}_1, \mathsf{D}_2\}, \mathbf{ps}, \mathbf{ps'} \in \{\spadesuit, \diamondsuit, \clubsuit, \heartsuit\} \\ \}$$

- *Operations of the registered physician* D*:*

$$
\begin{aligned}
\mathcal{O}_\mathsf{D} ::= \quad \{ \quad & \text{register}_{(\mathsf{D},\mathsf{KV},\mathsf{H})},\ \text{gen}^{\mathsf{i}}_{\mathsf{D}},\ \text{gen}^{\mathsf{p}}_{\mathsf{D}} \\
& \text{pub}_{(\mathsf{H},\mathsf{D})}(\mathbf{k}), \\
& \text{show}_{(\mathbf{P},\mathsf{D})}(\mathbf{ps}), \\
& \text{s\_issue}_{(\mathsf{D},\mathbf{P})},\ \text{e\_issue}_{(\mathsf{D},\mathbf{P})} \\
& \text{sign}_{\mathsf{D}}(\text{inv}(\mathbf{ps})) \\
& \text{dep}_{(\mathsf{D},\mathsf{KV})}(\sigma^{\text{inv}}_{\mathsf{D}}(\mathsf{t}^{\mathsf{I}}(\mathbf{ps}))) \\
& \text{reimb}_{(\mathsf{KV},\mathsf{D})}(\sigma^{\text{inv}}_{\mathsf{D}}(\mathsf{t}^{\mathsf{I}}(\mathbf{ps}))) \\
& \mid \quad \mathbf{k} \in \{\mathsf{pk}_\mathsf{H}, \mathsf{pk}_\mathsf{F}, \mathsf{pp}_{\mathsf{D}_1}, \mathsf{pp}_{\mathsf{D}_2}, \mathsf{pp}_\mathsf{G}, \mathsf{pi}_{\mathsf{D}_1}, \mathsf{pi}_{\mathsf{D}_2}, \mathsf{pi}_\mathsf{G}, \}, \\
& \mathbf{P} \in \{\mathsf{P}_1, \mathsf{P}_2\}, \mathbf{ps},\ \mathbf{ps'} \in \{\spadesuit, \diamondsuit, \clubsuit, \heartsuit\} \\
& \}
\end{aligned}
$$

The distributed set of operations is then defined as follows:

$$\tilde{\mathcal{O}} := (\mathcal{O}_\mathsf{H}, \mathcal{O}_{\mathsf{P}_1}, \mathcal{O}_{\mathsf{P}_2}, \mathcal{O}_\mathsf{F}, \mathcal{O}_\mathsf{KV}, \mathcal{O}_{\mathsf{D}_1}, \mathcal{O}_{\mathsf{D}_2})$$

7.2.2 Semantic Constraints – Temporal Behavior

We will now logically specify the semantic constraints about the temporal behavior of the system. All the following formulae are to be satisfied by the initial configuration of a run in an appropriate model.

The operations in our language do not carry any semantics themselves, they are only syntactic elements. We define the meaning of the operations over the interpretation of the relevant propositions. This is done in the logic itself. For each operation we give a number of preconditions that must be met for this operation to be executable, and describe its effect. The interpretation of most propositions is closely related to only one operation. We can specify that their interpretation can be changed only by a certain operation.

We specify the appropriate semantic constraints such as "once a key is generated, it will exist forever". These formulae restrict the set of runs that we accept as appropriate.

Here, we encounter the *frame problem*, a prominent problem in the context of Artificial Intelligence. When specifying the effects of an operation, we somehow have to deal with the non-effects as well. This problem was first mentioned by [MH69] and has since then been well investigated by Reiter, Levesque, Shanahan, etc. among others (see for example [Rei91, SL93, Sha97]). We will further discuss this problem in section 7.3.

Syntactic Abbreviations

For the following logical specification it will be convenient to have some syntactic abbreviations in addition to the ones already defined in chapter 4.

1. It is convenient to use a kind of *meta logic* for conjunctions and disjunctions of sets of agents, operations and propositions. For example, we write $\bigwedge_{p \in \mathcal{P}_i} p$ instead of $p_1 \wedge p_2 \wedge p_3 \wedge p_4$ for $\mathcal{P}_i = \{p_1, p_2, p_3, p_4\}$. This kind of abbreviation is possible because the set of agents, the set of operations and the set of propositions are finite which means, we could always explicitly write down the complete conjunction or disjunction.
2. We will often specify that the value of a proposition can be changed only through a particular operation. In this case, we require that

   the operation is actually performed *before* the change of the value. To specify this, the following abbreviation will be useful:

   $$\begin{aligned} \phi\, \mathcal{W}_i^{+} \psi &\equiv \phi\, \mathcal{W}_i(\phi \wedge \psi) \quad \text{and} \\ \phi\, \mathcal{U}_i^{+} \psi &\equiv \phi\, \mathcal{U}_i(\phi \wedge \psi) \end{aligned}$$

3. We denote the set of all public keys by
   pubkey $:= \{pk_H, pk_F, pp_{D_1}, pp_{D_2}, pp_G, pi_{D_1}, pi_{D_2}, pi_G\}$.
4. We denote the set of public keys generated by agent i as pubkey(i).
5. We denote the set of all pseudonyms by $Ps := \{\spadesuit, \diamondsuit, \clubsuit, \heartsuit\}$.
6. We denote the set of all possible signed invoices of physician D by
   $\Sigma(D) := \{\sigma_D^{\text{inv}}(t^I(ps)) \mid ps \in \{\spadesuit, \diamondsuit, \clubsuit, \heartsuit\}\}$.
7. The pharmacist encloses the transcript of a an I-certificate and of a prescription into her invoice. We denote the set of all possible signed invoices of pharmacist F by
   $\Sigma(F) := \{\sigma_F^{\text{inv}}(t^M(ps, D, ps')) \mid ps, ps' \in Ps, D \in \{D_1, D_2\}\}$

In the following specification we will repeatedly formalize that the execution of a particular operation is not possible. The formalization might seem to be a bit surprising so we explain it in advance: We specify that a particular operation a is *not* executed next by some agent i, by formula $[a]_i\bot$: For all operations a that agent i performs next, it holds that after the performance of operation a the formula $\bot$ holds. Since, according to the definition of the semantics, $\bot$ does not hold in any situation, operation a cannot be performed next.

Initialization of the system

In the initial situation, all propositions are false.

$$\bigwedge_{i \in Ag} \bigwedge_{p \in \mathcal{P}_i} \neg p@i \tag{1}$$

Registration

The first operation performed by a physician and by a pharmacy is their respective registration operation which they perform jointly with the health insurance.

$$\langle \text{register}_{(D_1,KV,H)} \rangle_{D_1} \top \wedge \langle \text{register}_{(D_2,KV,H)} \rangle_{D_2} \top \wedge \langle \text{register}_{(F,H)} \rangle_{F} \top \tag{2}$$

Generation and Publication of Cryptographic Keys

The health insurance H, the physicians D_1 and D_2, the KV and the pharmacy F need cryptographic keys to perform certain operations. In order to not have to deal with outdated keys, we assume that each key, once it is generated, will exist forever.

Each generation operation always generates its respective secret and public cryptographic keys. Once a key is generated, it will exist forever, and no agent will perform the generation operation more than once.

We can read the first line,

$$\Box_H [\text{gen}_H]_H \Box_H (pk_H@H \wedge sk_H@H \wedge [\text{gen}_H]_H \bot),$$

of the following formula (3) as follows:

Always from the point of view of agent H (denoted by $\Box_H$):
If gen_H is the next operation from H's point of view (denoted by $[\text{gen}_H]_H$),
then after the execution of gen_H it holds that always from the point of view of H (denoted by $\Box_H$),
H has the public key pk_H (denoted by the proposition $pk_H@H$)
and the private key sk_H (denoted by the proposition $sk_H@H$)
and will not next perform the generation operation gen_H (denoted by $[\text{gen}_H]_H\bot$).

The other of the formula lines can be read accordingly.

$$\begin{aligned}
&\Box_{H}[\mathrm{gen}_{H}]_{H}\Box_{H}(pk_{H}@H \wedge sk_{H}@H \wedge [\mathrm{gen}_{H}]_{H}\bot)\\
\wedge\ &\Box_{F}[\mathrm{gen}_{F}]_{F}\Box_{F}(pk_{F}@F \wedge sk_{F}@F \wedge [\mathrm{gen}_{F}]_{F}\bot)\\
\wedge\ &\Box_{D_1}[\mathrm{gen}^{p}_{D_1}]_{D_1}\Box_{D_1}(pp_{D_1}@D1 \wedge sp_{D_1}@D1 \wedge [\mathrm{gen}^{p}_{D_1}]_{D_1}\bot)\\
\wedge\ &\Box_{D_1}[\mathrm{gen}^{i}_{D_1}]_{D_1}\Box_{D_1}(pi_{D_1}@D1 \wedge si_{D_1}@D1 \wedge [\mathrm{gen}^{i}_{D_1}]_{D_1}\bot)\\
\wedge\ &\Box_{D_2}[\mathrm{gen}^{p}_{D_2}]_{D_2}\Box_{D_2}(pp_{D_2}@D2 \wedge sp_{D_2}@D2 \wedge [\mathrm{gen}^{p}_{D_1}]_{D_2}\bot)\\
\wedge\ &\Box_{D_2}[\mathrm{gen}^{i}_{D_2}]_{D_2}\Box_{D_2}(pi_{D_2}@D2 \wedge si_{D_2}@D2 \wedge [\mathrm{gen}^{i}_{D_2}]_{D_2}\bot)\\
\wedge\ &\Box_{KV}[\mathrm{gen}^{i}_{KV}]_{KV}\Box_{KV}(pp_{G}@KV \wedge sp_{G}@KV \wedge [\mathrm{gen}^{i}_{KV}]_{KV}\bot)\\
\wedge\ &\Box_{KV}[\mathrm{gen}^{p}_{KV}]_{KV}\Box_{KV}(pi_{G}@KV \wedge si_{G}@KV \wedge [\mathrm{gen}^{p}_{KV}]_{KV}\bot)
\end{aligned} \tag{3}$$

Cryptographic keys can only be generated by the corresponding generation operation.

Again, we explain the first line,

$$\neg(pk_{H}@H \vee sk_{H}@H)\ \mathcal{W}^{+}_{H}\langle \mathrm{gen}_{H}\rangle_{H}\top,$$

in more detail:

Agent H does not have the public and private public keys pk_H and sk_H (denoted by $\neg(pk_H@H \vee sk_H@H)$)
until from H's point of view (denoted by $\mathcal{W}^{+}_{H}$)
an operation $\langle \mathrm{gen}_H\rangle_H$ will occur next. (denoted by $\langle \mathrm{gen}_H\rangle_H\top$).

Note, that this formula contains the abbreviation $\mathcal{W}^{+}_{H}$, which indicates, that agent H has the respective keys only *after* the operation gen_H.

$$\begin{aligned}
&\neg(pk_{H}@H \vee sk_{H}@H)\ \mathcal{W}^{+}_{H}\langle \mathrm{gen}_{H}\rangle_{H}\top\\
\wedge\ &\neg(pi_{G}@KV \vee si_{G}@KV)\ \mathcal{W}^{+}_{KV}\langle \mathrm{gen}^{i}_{KV}\rangle_{KV}\top\\
\wedge\ &\neg(pp_{G}@KV \vee sp_{G}@KV)\ \mathcal{W}^{+}_{KV}\langle \mathrm{gen}^{p}_{KV}\rangle_{KV}\top\\
\wedge\ &\neg(pi_{D_1}@D1 \vee si_{D_1}@D1)\ \mathcal{W}^{+}_{D_1}\langle \mathrm{gen}^{i}_{D_1}\rangle_{D_1}\top\\
\wedge\ &\neg(pp_{D_1}@D1 \vee sp_{D_1}@D1)\ \mathcal{W}^{+}_{D_1}\langle \mathrm{gen}^{p}_{D_1}\rangle_{D_1}\top\\
\wedge\ &\neg(pi_{D_2}@D2 \vee si_{D_2}@D2)\ \mathcal{W}^{+}_{D_2}\langle \mathrm{gen}^{i}_{D_2}\rangle_{D_2}\top\\
\wedge\ &\neg(pp_{D_2}@D2 \vee sp_{D_2}@D2)\ \mathcal{W}^{+}_{D_2}\langle \mathrm{gen}^{p}_{D_2}\rangle_{D_2}\top\\
\wedge\ &\neg(pk_{F}@F \vee sk_{F}@F)\ \mathcal{W}^{+}_{F}\langle \mathrm{gen}_{F}\rangle_{F}\top
\end{aligned} \tag{4}$$

The KV cannot generate the group keys of group G until it has stored the respective individual keys of all physicians belonging to group G, i.e. of both physicians D_1 and D_2.

$$
\begin{aligned}
&[\mathrm{gen}^{\mathsf{i}}_{\mathsf{KV}}]_{\mathsf{KV}}\bot \;\; \mathcal{W}_{\mathsf{KV}}\ (\mathsf{pi}_{\mathsf{D}_1}@\mathsf{KV} \wedge \mathsf{pi}_{\mathsf{D}_2}@\mathsf{KV}) \\
\wedge\ &[\mathrm{gen}^{\mathsf{p}}_{\mathsf{KV}}]_{\mathsf{KV}}\bot \;\; \mathcal{W}_{\mathsf{KV}}\ (\mathsf{pp}_{\mathsf{D}_1}@\mathsf{KV} \wedge \mathsf{pp}_{\mathsf{D}_2}@\mathsf{KV})
\end{aligned}
\tag{5}
$$

Agents need public keys of other agents for verification purposes. Always, when some agent i publishes a public key p to some agent j then after the publishing operation agent j will hold the public key of agent i forever.

$$
\bigwedge_{\mathsf{k}\in\mathrm{pubkey}} \; \bigwedge_{\substack{\mathsf{i},\mathsf{j}\in\mathsf{Ag}\\ \mathsf{i}\neq\mathsf{j}}} \; \Box_{\mathsf{j}}\ [\mathrm{pub}_{(\mathsf{i},\mathsf{j})}(\mathsf{k})]_{\mathsf{j}}\ \Box_{\mathsf{j}}\mathsf{k}@\mathsf{j}
\tag{6}
$$

No agent i has the public key of another agent until some agent j publishes the key to agent i.

$$
\bigwedge_{\mathsf{i}\in\mathsf{Ag}} \; \bigwedge_{\mathsf{k}\in\mathrm{pubkey}\setminus\mathrm{pubkey}(\mathsf{i})} \Big(\neg\mathsf{k}@\mathsf{i}\;\; \mathcal{W}^{+}_{\mathsf{i}}\ \big(\bigvee_{\substack{\mathsf{j}\neq\mathsf{i}\\ \mathsf{j}\in\mathsf{Ag}}} \langle\mathrm{pub}_{(\mathsf{j},\mathsf{i})}(\mathsf{k})\rangle_{\mathsf{i}}\top\big)\Big)
\tag{7}
$$

It always holds that agent j can publish the key to agent i only if j holds the key herself:

$$
\bigwedge_{\mathsf{k}\in\mathrm{pubkey}} \; \bigwedge_{\substack{\mathsf{i},\mathsf{j}\in\mathsf{Ag}\\ \mathsf{i}\neq\mathsf{j}}} \Big(\Box_{\mathsf{j}}\big(\neg\mathsf{k}@\mathsf{j} \Rightarrow [\mathrm{pub}_{(\mathsf{j},\mathsf{i})}(\mathsf{k})]_{\mathsf{j}}\bot\big)\Big)
\tag{8}
$$

Issuing I-certificates

The health insurance issues I-certificates to the policy holders who can use them to claim treatment from physicians and medicaments from the pharmacy according to prescriptions. We now specify the effects of the operation $\mathrm{issue}^{\mathrm{I}}_{(\mathsf{H},\mathsf{P})}$ by which the health insurance H issues I-certificates to policy holders.

Operation $\mathrm{issue}^{\mathrm{I}}_{(\mathsf{H},\mathsf{P})}$ is the only operation, through which I-certificates are issued.

$$\bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{ps\in PS} \Big(\neg \mathrm{I}(ps)\ \mathcal{W}^{+}_{P}\ \langle \mathrm{issue}^{\mathrm{I}}_{(H,P)}\rangle_{P}\top\Big) \tag{9}$$

For issuing I-certificates to its policy holders, the health insurance always needs its secret key sk_H:

$$\bigwedge_{P\in\{P_1,P_2\}} \Big(\Box_{H}\big(\neg sk_H@H \Rightarrow [\mathrm{issue}^{\mathrm{I}}_{(H,P)}]_{H}\bot\big)\Big) \tag{10}$$

Once a policy holder holds an I-certificate, she will hold it forever.

$$\bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{ps\in PS} \Big(\Box_{P}\big(\mathrm{I}(ps)@P\ \Rightarrow\ \Box_{P}\mathrm{I}(ps)@P\big)\Big) \tag{11}$$

Each I-certificate (identified by its pseudonym) can be issued only to one policy holder:

$$\begin{aligned}\bigwedge_{ps\in PS} \Bigg(& \Box_{H}\Bigg([\mathrm{issue}^{\mathrm{I}}_{(H,P_1)}]_{H}\Big(\mathrm{I}(ps)@P_1 \Rightarrow \Box_{H}\big([\mathrm{issue}^{\mathrm{I}}_{(H,P_2)}]_{H}(\neg \mathrm{I}(ps)@P_2)\big)\Big)\Bigg)\\ \wedge\ & \Box_{H}\Bigg([\mathrm{issue}^{\mathrm{I}}_{(H,P_2)}]_{H}\Big(\mathrm{I}(ps)@P_2 \Rightarrow \Box_{H}\big([\mathrm{issue}^{\mathrm{I}}_{(H,P_1)}]_{H}(\neg \mathrm{I}(ps)@P_1)\big)\Big)\Bigg)\Bigg)\end{aligned} \tag{12}$$

Each issue operation between a policy holder P and the health insurance H issues *exactly one* I-certificate[2].

There are basically two ways to formalize this: One easy way is to simply introduce four different operations, one for each pseudonym. The disadvantage of this solution is that the health insurance, that participates in each operation, can distinguish among the four operations. It would be awkward to demand that though the outcome of the operation is deterministic, the health insurance does not know, on which pseudonym each of the four operations issues an I-certificate. We prefer a different solution here: We formulate a number of constraints saying that if a policy holder P holds x I-certificates and then performs the issue operation, she will afterwards hold $x + 1$ I-certificates. Which I-certificate she receives through the operation is non-deterministic.

[2] Two different I-certificates are distinguished by their pseudonyms.

We first specify the fact that policy holder P holds exactly x I-certificates, with $x \in \{1, \ldots, 4\}$, each issued on a different pseudonyms:

$\boldsymbol{\alpha_P(ps)}$ means, policy holder $\mathsf{P} \in \{\mathsf{P}_1, \mathsf{P}_2\}$ holds exactly one I-certificate. This I-certificate is issued on pseudonym ps $\in \mathsf{Ps}$:

$$\alpha_{\mathsf{P}}(ps) ::= I(ps)@\mathsf{P} \wedge \bigwedge_{ps' \in \mathsf{PS} \setminus \{ps\}} \neg I(ps')@\mathsf{P}$$

$\boldsymbol{\alpha_P(ps_1, ps_2)}$ with ps1 $\neq ps_2$ means, policy holder $\mathsf{P} \in \{\mathsf{P}_1, \mathsf{P}_2\}$ holds exactly two I-certificates issued on pseudonyms ps_1 and $ps_2 \in \mathsf{Ps}$ respectively:

$$\alpha_{\mathsf{P}}(ps_1, ps_2) ::= I(ps_1)@\mathsf{P} \wedge I(ps_2)@\mathsf{P} \wedge \bigwedge_{ps' \in \mathsf{PS} \setminus \{ps_1, ps_2\}} \neg I(ps')@\mathsf{P}$$

$\boldsymbol{\alpha_P(ps_1, ps_2, ps_3)}$ with $ps_i \neq ps_j$ for $i, j \in \{1, 2, 3\}$ and $i \neq j$ means, policy holder $\mathsf{P} \in \{\mathsf{P}_1, \mathsf{P}_2\}$ holds exactly three I-certificates, issued on pseudonyms ps_1, ps_2 and $ps_3 \in \mathsf{Ps}$:

$$\begin{array}{rcl} \alpha_{\mathsf{P}}(ps_1, ps_2, ps_3) & ::= & I(ps_1)@\mathsf{P} \\ & \wedge & I(ps_2)@\mathsf{P} \\ & \wedge & I(ps_3)@\mathsf{P} \\ & \wedge & \displaystyle\bigwedge_{ps' \in \mathsf{PS} \setminus \{ps_1, ps_2, ps_3\}} \neg I(ps')@\mathsf{P} \end{array}$$

$\boldsymbol{\alpha_P(ps_1, ps_2, ps_3, ps_4)}$ with $ps_i \neq ps_j$ for $i, j \in \{1, 2, 3, 4\}$ and $i \neq j$ means, policy holder $\mathsf{P} \in \{\mathsf{P}_1, \mathsf{P}_2\}$ holds exactly four I-certificates, issued on pseudonyms ps_1, ps_2, ps_3 and $ps_4 \in \mathsf{Ps}$ respectively:

$$\begin{array}{rcl} \alpha_{\mathsf{P}}(ps_1, ps_2, ps_3, ps_4) & ::= & I(ps_1)@\mathsf{P} \\ & \wedge & I(ps_2)@\mathsf{P} \\ & \wedge & I(ps_3)@\mathsf{P} \\ & \wedge & I(ps_4)@\mathsf{P} \end{array}$$

We can now formulate for each policy holder the effects of the issue operation in dependency of the situation directly before the issue operation. For policy holder $\mathsf{P} \in \{\mathsf{P}_1, \mathsf{P}_2\}$ we call this formula about the effects of the issue operation β_{P}.

$\boldsymbol{\beta_P^0}$ means that if policy holder P does not have any I-certificate, he will get exactly one by performing the issuing operation:

$$\beta_P^0 ::= \neg\big(I(\spadesuit)@P \vee I(\diamondsuit)@P \vee I(\heartsuit)@P \vee I(\clubsuit)@P\big) \\ \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\heartsuit) \vee \alpha_P(\clubsuit) \vee \alpha_P(\spadesuit) \vee \alpha_P(\diamondsuit)\big)$$

$\boldsymbol{\beta_P^1}$ means that if policy holder P holds one I-certificate and performs an issue operation, he will have exactly two I-certificates after the performance of the operation, the old certificate and a new certificate issued on a different pseudonym:

$$\begin{aligned} \beta_P^1 ::= \quad & \alpha_P(\diamondsuit) \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\diamondsuit,\heartsuit) \vee \alpha_P(\diamondsuit,\spadesuit) \vee \alpha_P(\diamondsuit,\clubsuit)\big) \\ \wedge \quad & \alpha_P(\heartsuit) \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\heartsuit,\diamondsuit) \vee \alpha_P(\heartsuit,\spadesuit) \vee \alpha_P(\heartsuit,\clubsuit)\big) \\ \wedge \quad & \alpha_P(\clubsuit) \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\clubsuit,\heartsuit) \vee \alpha_P(\clubsuit,\spadesuit) \vee \alpha_P(\clubsuit,\diamondsuit)\big) \\ \wedge \quad & \alpha_P(\spadesuit) \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\spadesuit,\heartsuit) \vee \alpha_P(\spadesuit,\diamondsuit) \vee \alpha_P(\spadesuit,\clubsuit)\big) \end{aligned}$$

$\boldsymbol{\beta_P^2}$ means that if policy holder P holds two I-certificates and performs an issue operation, she will have exactly three I-certificates after the execution of the issuing operation, the old certificates and another one issued on a different pseudonym:

$$\begin{aligned} \beta_P^2 ::= \quad & \alpha_P(\diamondsuit,\heartsuit) \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\diamondsuit,\heartsuit,\spadesuit) \vee \alpha_P(\diamondsuit,\heartsuit,\clubsuit)\big) \\ \wedge \quad & \alpha_P(\diamondsuit,\spadesuit) \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\diamondsuit,\spadesuit,\heartsuit) \vee \alpha_P(\diamondsuit,\spadesuit,\clubsuit)\big) \\ \wedge \quad & \alpha_P(\diamondsuit,\clubsuit) \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\diamondsuit,\clubsuit,\heartsuit) \vee \alpha_P(\diamondsuit,\clubsuit,\spadesuit)\big) \\ \wedge \quad & \alpha_P(\spadesuit,\clubsuit) \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\spadesuit,\clubsuit,\heartsuit) \vee \alpha_P(\spadesuit,\clubsuit,\diamondsuit)\big) \\ \wedge \quad & \alpha_P(\heartsuit,\clubsuit) \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\heartsuit,\clubsuit,\diamondsuit) \vee \alpha_P(\heartsuit,\clubsuit,\spadesuit)\big) \\ \wedge \quad & \alpha_P(\heartsuit,\spadesuit) \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\big(\alpha_P(\heartsuit,\spadesuit,\diamondsuit) \vee \alpha_P(\heartsuit,\spadesuit,\clubsuit)\big) \end{aligned}$$

$\boldsymbol{\beta_P^3}$ means that if policy holder P holds three I-certificates and performs an issue operation, she will have four I-certificates after the execution of the operation, all certificates issued on different pseudonyms.

$$\beta_P^3 ::= \big(\alpha_P(\heartsuit,\clubsuit,\spadesuit) \vee \alpha_P(\heartsuit,\clubsuit,\diamondsuit) \vee \alpha_P(\heartsuit,\spadesuit,\diamondsuit) \vee \alpha_P(\clubsuit,\spadesuit,\diamondsuit)\big) \\ \Rightarrow [\mathrm{issue}^I_{(H,P)}]_P\alpha_P(\heartsuit,\diamondsuit,\spadesuit,\clubsuit)$$

$\boldsymbol{\beta^4_P}$ means that if policy holder P holds four I-certificates, she cannot perform another issue operation:

$$\beta^4_{\mathsf{P}} ::= \alpha_{\mathsf{P}}(\heartsuit, \clubsuit, \spadesuit, \diamondsuit) \Rightarrow [\text{issue}^{\mathsf{I}}_{(\mathsf{H},\mathsf{P})}]_{\mathsf{P}}\bot \tag{13}$$

We can now define the effect of the issue operation by formalizing that for both policy holders it holds that they will get exactly one more I-certificate by participating in the issue operation:

$$\bigwedge_{\mathsf{P}\in\{\mathsf{P}_1,\mathsf{P}_2\}} \Box_{\mathsf{P}}(\beta^0_{\mathsf{P}} \land \beta^1_{\mathsf{P}} \land \beta^2_{\mathsf{P}} \land \beta^3_{\mathsf{P}} \land \beta^4_{\mathsf{P}}) \tag{14}$$

Requesting Treatment

A policy holder requests treatment from a physician by showing one of her I-certificates.

When a policy holder P shows her I-certificate to a physician D, the physician receives the transcript of the I-certificate denoted by proposition $\mathsf{t}^{\mathsf{I}}(\mathsf{ps})@\mathsf{D}$, temporarily registers P as a patient under pseudonym ps, denoted by proposition $(\mathsf{P},\mathsf{ps})@\mathsf{D}$ and learns forever that health insurance H has issued an I-certificate on pseudonym ps, denoted by proposition $(\mathsf{H},\mathsf{ps})@\mathsf{D}$.

$$\bigwedge_{\mathsf{D}\in\{\mathsf{D}_1,\mathsf{D}_2\}} \bigwedge_{\mathsf{P}\in\{\mathsf{P}_1,\mathsf{P}_2\}} \bigwedge_{\mathsf{ps}\in\mathsf{Ps}} \Big(\Box_{\mathsf{D}} \quad [\text{show}_{(\mathsf{P},\mathsf{D})}(\mathsf{ps})]_{\mathsf{D}} \left(\mathsf{t}^{\mathsf{I}}(\mathsf{ps})@\mathsf{D} \land (\mathsf{P},\mathsf{ps})@\mathsf{D} \land \Box_{\mathsf{D}}(\mathsf{H},\mathsf{ps})@\mathsf{D}\right)\Big) \tag{15}$$

A policy holder can register at a physician only by showing her an I-certificate. A physician receives a transcript and gets to know which I-certificates health insurance H has issued only through a show operation. For proposition $(\mathsf{H},\mathsf{ps})@\mathsf{D}$ we model this constraint differently from the other propositions. According to constraint (15) once the physician D has learned that the health insurance H has issued an I-certificate with pseudonym ps, she will never forget this. Thus, once proposition $(\mathsf{H},\mathsf{ps})@\mathsf{D}$ is true, it will always remain true. This is not the case for propositions $(\mathsf{P},\mathsf{ps})@\mathsf{D}$ and $\mathsf{t}^{\mathsf{I}}(\mathsf{ps})@\mathsf{D}$.

$$\bigwedge_{D \in D_1, D_2} \bigwedge_{P \in \{P_1, P_2\}} \bigwedge_{ps \in Ps} \Big(\Box_D \Big((\neg t^I(ps)@D) \Rightarrow (\neg t^I(ps)@D) \ \mathcal{W}^+_D \ (\bigvee_{P' \in \{P_1, P_2\}} \langle \mathrm{show}_{(P',D)}(ps) \rangle_D \top) \Big)$$
$$\wedge \ \Box_D \Big((\neg (P, ps)@D) \Rightarrow (\neg (P, ps)@D) \ \mathcal{W}^+_D \ \langle \mathrm{show}_{(P,D)}(ps) \rangle_D \top \Big)$$
$$\wedge \ \neg (H, ps)@P \ \mathcal{W}^+_D \ (\bigvee_{P \in \{P_1, P_2\}} \langle \mathrm{show}_{(P,D)}(ps) \rangle_D \top) \Big) \tag{16}$$

We have three preconditions for the show operation: A policy holder can show only those I-certificates to physicians that she holds. A physician does only participate in a show operation, if pseudonym ps has not already been shown to her before and if she has the public key of the health insurance so that she can verify the I-certificates.

$$\bigwedge_{P \in \{P_1, P_2\}} \bigwedge_{D \in \{D_1, D_2\}} \bigwedge_{ps \in Ps} \Big(\Box_P \big((\neg I(ps)@P) \Rightarrow [\mathrm{show}_{(P,D)}(ps)]_P \bot \big)$$
$$\wedge \quad \Box_D \big((H, ps)@P \Rightarrow [\mathrm{show}_{(P,D)}(ps)]_D \bot \big)$$
$$\wedge \quad \Box_D \big((\neg pk_H@D) \Rightarrow [\mathrm{show}_{(P,D)}(ps)]_P \bot \big) \Big) \tag{17}$$

Issuing prescriptions While a policy holder is under treatment by a physician, the physician may issue a prescription to the policy holder. The policy holder adds one of his (fresh) pseudonyms to the prescription under which she will claim the medicament from a pharmacy. As already motivated in section 7.1.2, we prefer to model the process of issuing a prescription by a sequence of three operations: First, the issuing process starts with a joint operation $\mathrm{s\_issue}_{(D,P)}$ of both the physician D and the policy holder P. Then the policy holder adds a fresh pseudonym by operation $\mathrm{add}_P(ps)$ and finally the process is completed by the joint operation $\mathrm{e\_issue}_{(D,P)}$. Figure 7.1 illustrates this process.

We now describe the prescription process. From the physician's point of view, there is always a completion operation directly after the starting operation of an issuing process. From the point of view of the policy holder, there are always three operations directly in a row. The completion operation always follows directly after the

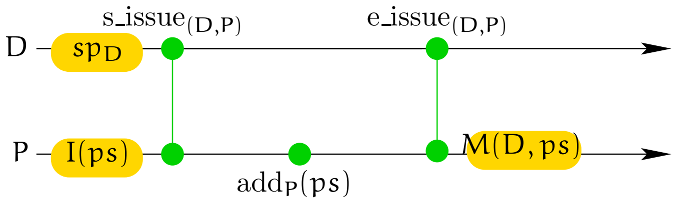


Figure 7.1: Prescription process

operation $add_P(ps)$ which in turn follows directly after the starting operation of the issuing process.

$$\begin{aligned}
&\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{P\in\{P_1,P_2\}} \\
&\Big(\Box_D[\text{s_issue}_{(D,P)}]_D\langle \text{e_issue}_{(D,P)}\rangle_D\top\Big) \\
\wedge\ &\Big(\Box_P[\text{s_issue}_{(D,P)}]_P\big(\bigvee_{ps\in Ps}(\langle add_P(ps)\rangle_P\langle \text{e_issue}_{(D,P)}\rangle_P\top)\big)\Big)
\end{aligned} \tag{18}$$

From the physician's point of view, the completion of the issuing process cannot take place after any other operation than the start operation. From the point of view of the policy holder, adding a pseudonym to a prescription can take place only directly after the starting operation of the issuing process and the completion of the issuing process can take place only directly after adding the pseudonym to the prescription.

$$\begin{aligned}
&\bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{D\in\{D_1,D_2\}} \\
&\Big(\ \Box_D \bigwedge_{op\in\mathcal{O}_D\setminus\{\text{s_issue}_{(D,P)}\}} \big([op]_D[\text{e_issue}_{(D,P)}]_D\bot\big) \\
&\wedge \Box_P \bigwedge_{op\in\mathcal{O}_P\setminus\{\text{s_issue}_{(D,P)}\}} \bigwedge_{ps\in Ps} \big([op]_P[add_P(ps)]_P\bot\big) \\
&\wedge \Box_P \bigwedge_{ps\in Ps} \bigwedge_{op\in\mathcal{O}_P\setminus\{add_P(ps)\}} \big([op]_P[\text{e_issue}_{(D,P)}]_P\bot\big)\Big)
\end{aligned} \tag{19}$$

After the process of issuing a prescription, the participating policy holder holds the issued prescription:

$$\bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{ps\in Ps} \Big(\Box_P \left([\mathrm{add}_P(ps)]_P\, [\mathrm{e\_issue}_{(D,P)}]_P\, M(D,ps)@P\right)\Big) \tag{20}$$

The only way for a policy holder to get hold of a prescription is to follow the issuing process.

$$\bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{ps\in Ps} \Box_P\Big(\neg M(D,ps)@P \Rightarrow \left(\neg M(D,ps)@P\ \mathcal{W}_P^{+}\langle \mathrm{add}_P(ps)\rangle_P \langle \mathrm{e\_issue}_{(D,P)}\rangle_P \top\right)\Big) \tag{21}$$

A physician needs her secret prescription key sp_D, her public prescription key pp_D and her public group key for prescriptions pp_G to issue a prescription. She issues prescriptions only to those policy holders that are currently under her treatment. The policy holder must hold an I-certificate to participate in the prescription process and she can only add a pseudonym of one of her I-certificates.

$$\begin{aligned}
&\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{P\in\{P_1,P_2\}} \\
&\quad \Box_D\Big(\neg\big(sp_D@D \wedge pp_D@D \wedge pp_G@D\big) \Rightarrow [\mathrm{s_issue}_{(D,P)}]_D \bot\Big) \\
\wedge &\quad \Big(\big(\bigwedge_{ps\in Ps} (\neg(P,ps)@D)\big) \Rightarrow [\mathrm{s_issue}_{(D,P)}]_D \bot\Big) \\
\wedge &\quad \Box_P\Big(\big(\bigwedge_{ps\in Ps} \neg I(ps)@P\big) \Rightarrow [\mathrm{s_issue}_{(D,P)}]_P \bot\Big) \\
\wedge &\quad \bigwedge_{ps\in Ps} \Big(\Box_P\big(\neg I(ps)@P \Rightarrow [\mathrm{add}_P(ps)]_P \bot\big)\Big)
\end{aligned} \tag{22}$$

Claiming Medicine from a Pharmacy

A policy holder claims prescribed medicaments from the pharmacy by showing the prescription and the corresponding I-certificate. If the pharmacy accepts the certificates, it receives a transcript of the prescription and the I-certificate and delivers the requested medicament.

From the show operation, the pharmacy receives a transcript of each show-operation and learns (forever) that the health insurance H has issued an I-certificate with pseudonym ps.

$$\bigwedge_{\mathsf{P}\in\{\mathsf{P}_1,\mathsf{P}_2\}} \bigwedge_{\mathsf{D}\in\{\mathsf{D}_1,\mathsf{D}_2\}} \bigwedge_{\mathsf{ps},\mathsf{ps}'\in\mathsf{Ps}} \Box_{\mathsf{F}}\Big([\mathrm{gShow}_{(\mathsf{P},\mathsf{F})}(\mathsf{ps},\mathsf{D},\mathsf{ps}')]_{\mathsf{F}}\big(\mathsf{t}^{\mathsf{M}}(\mathsf{ps},\mathsf{D},\mathsf{ps}')@\mathsf{F} \wedge (\Box_{\mathsf{F}}(\mathsf{H},\mathsf{ps})@\mathsf{F})\big)\Big) \tag{23}$$

The show operation is the only operation through which the pharmacy receives a transcript and learns, to which health insurance a pseudonym belongs.

$$\begin{aligned} &\Bigg(\bigwedge_{\mathsf{D}\in\{\mathsf{D}_1,\mathsf{D}_2\}} \bigwedge_{\mathsf{ps},\mathsf{ps}'\in\mathsf{Ps}} \Box_{\mathsf{F}}\Big(\neg \mathsf{t}^{\mathsf{M}}(\mathsf{ps},\mathsf{D},\mathsf{ps}')@\mathsf{F} \Rightarrow \\ &\qquad \neg \mathsf{t}^{\mathsf{M}}(\mathsf{ps},\mathsf{D},\mathsf{ps}')@\mathsf{F}\ \mathcal{W}^{+}_{\mathsf{F}} \bigvee_{\mathsf{P}\in\{\mathsf{P}_1,\mathsf{P}_2\}} \langle \mathrm{gShow}_{(\mathsf{P},\mathsf{F})}(\mathsf{ps},\mathsf{D},\mathsf{ps}')\rangle\top\Big)\Bigg) \\ \wedge\ &\Bigg(\bigwedge_{\mathsf{ps}\in\mathsf{Ps}} \bigwedge_{\mathsf{P}\in\{\mathsf{P}_1,\mathsf{P}_2\}} \Big(\neg \mathsf{K}_{\mathsf{F}}\mathrm{I}(\mathsf{ps})@\mathsf{P}\ \mathcal{W}^{+}_{\mathsf{F}} \\ &\qquad \bigvee_{\mathsf{D}\in\{\mathsf{D}_1,\mathsf{D}_2\}} \bigvee_{\mathsf{ps}'\in\mathsf{Ps}} \langle \mathrm{gShow}_{(\mathsf{P},\mathsf{F})}(\mathsf{ps},\mathsf{D},\mathsf{ps}')\rangle_{\mathsf{F}}\top\Big)\Bigg) \end{aligned} \tag{24}$$

Note, that the above formalization is not allowed by the restricted language $\mathcal{L}^c$: The sub formula $\mathsf{K}_{\mathsf{F}}\mathrm{I}(\mathsf{ps})@\mathsf{P}$ is syntactically not possible in $\mathcal{L}^c$. We could modify the formula by introducing a new proposition $(\mathrm{I}(\mathsf{ps}),\mathsf{P})@\mathsf{F}$ which intuitively stands for “the pharmacy holds a note, that policy holder P holds I-certificate $\mathrm{I}(\mathsf{ps})$”. The sub formula $\mathsf{K}_{\mathsf{F}}\mathrm{I}(\mathsf{ps})@\mathsf{P}$ could then be substituted by formula $\mathsf{K}_{\mathsf{F}}(\mathrm{I}(\mathsf{ps}),\mathsf{P})@\mathsf{F}$.

During the show operation, the pharmacy verifies the prescription and the I-certificate. To do so, it must hold the public group key for prescriptions pp_{G} and the public key of the health insurance, pk_{H}.

She accepts each I-certificate only once.

$$\bigwedge_{\mathsf{P}\in\{\mathsf{P}_1,\mathsf{P}_2\}} \bigwedge_{\mathsf{D}\in\{\mathsf{D}_1,\mathsf{D}_2\}} \bigwedge_{\mathsf{ps},\mathsf{ps}'\in\mathsf{Ps}} \Box_{\mathsf{F}}\Big((\neg(\mathsf{pp}_{\mathrm{G}}@\mathsf{F} \wedge \mathsf{pk}_{\mathsf{H}}@\mathsf{F}) \vee (\mathsf{H},\mathsf{ps})@\mathsf{F}) \Rightarrow [\mathrm{gShow}_{(\mathsf{P},\mathsf{F})}(\mathsf{ps},\mathsf{D},\mathsf{ps}')]_{\mathsf{F}}\bot\Big) \tag{25}$$

Policy holder P claims medicaments from the pharmacy F by showing one of her prescriptions together with the corresponding I-certificate. To do so, she must hold both the prescription and the I-certificate.

$$\bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{ps,ps'\in Ps} \Box_P\Big(\neg\big(I(ps)@P \wedge M(D,ps')@P\big) \Rightarrow [\mathrm{gShow}_{(P,F)}(ps,D,ps')]_P\bot\Big) \tag{26}$$

Clearing – Physicians

To get reimbursed for her treatment, a physician encloses the appropriate transcript of an I-certificate in an invoice, signs the invoice with her secret signature key si_D and sends the signed invoice to the KV that will transfer it to the health insurance. The health insurance accepts the invoice if it originates from a physician of group G.

The health insurance H may then verify whether the I-certificate that led to the transcript enclosed in the invoice has been used before. Further, the health insurance H may either ask the KV to identify the signer of the invoice or must eventually send the money to the KV that has to transfer it to the appropriate physician.

A physician encloses the appropriate transcript of an I-certificate into an invoice and signs the invoice. After the sign operation, the signer holds a signed invoice.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{ps\in Ps} \Box_D\Big([\mathrm{sign}_D(\mathrm{inv}(t^I(ps)))]_D\ \sigma_D^{\mathrm{inv}}(t^I(ps))@D\Big) \tag{27}$$

The sign operation is the only operation by which a physician can get a signed invoice.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{ps\in Ps} \Box_D\Big(\neg\sigma_D^{\mathrm{inv}}(t^I(ps))@D \Rightarrow \big(\neg\sigma_D^{\mathrm{inv}}(t^I(ps))@D\ \mathcal{W}_D^{+}\langle\mathrm{sign}_D(\mathrm{inv}(ps))\rangle_D\top\big)\Big) \tag{28}$$

To perform the sign operation the physician needs her secret individual invoice key si_D, her public individual invoice key pi_D and the public group key for signing invoices, pi_G. Further, the physician must hold the transcript that she encloses in the invoice.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{ps\in Ps} \Box_D\Big(\neg\big(\mathsf{si}_D@D \wedge \mathsf{pi}_D@D \wedge \mathsf{pi}_G@D \wedge t^I(ps)\big) \Rightarrow [\mathrm{sign}_D(\mathrm{inv}(ps))]_D\bot\Big) \tag{29}$$

The physician can deposit her signed invoice at the KV. As explained above, we do not distinguish between several deposits of the same invoice. We assume that if the agent already holds an invoice, a second generation or deposit of the same invoice is simply ignored. After the physician has deposited the signed invoice at the KV, the KV holds the invoice.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{\sigma\in\Sigma(D)} \Box_{KV}\Big([\mathrm{dep}_{(D,KV)}(\sigma)]_{KV}\ \sigma@KV\Big) \tag{30}$$

The deposit operation is the only way for the KV to receive a signed invoice from a physician.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{\sigma\in\Sigma(D)} \Big(\Box_{KV}\big(\neg\sigma@KV \Rightarrow \neg\sigma@KV\ \mathcal{W}^{+}_{KV}\langle\mathrm{dep}_{(D,KV)}(\sigma)\rangle_{KV}\top\big)\Big) \tag{31}$$

The physician can only deposit signed invoices that she holds. The KV does only accept a signed invoice from a physician the public signature key of whom it holds.

$$\begin{aligned} &\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{\sigma\in\Sigma(D)} \\ &\quad \Box_D\Big(\neg\sigma@D \Rightarrow [\mathrm{dep}_{(D,KV)}(\sigma)]_D\bot\Big) \\ \wedge &\quad \Box_{KV}\Big(\neg\mathsf{pi}_D@KV \Rightarrow [\mathrm{dep}_{(D,KV)}(\sigma)]_{KV}\bot\Big) \end{aligned} \tag{32}$$

If the KV holds a signed invoice of a physician, it deposits the invoice at the health insurance. After the deposit operation, the health insurance holds the signed invoice and stores it forever.

$$\bigwedge_{D \in \{D_1, D_2\}} \quad \bigwedge_{\sigma \in \Sigma(D)} \Box_H \Big([\mathrm{dep}_{(\mathsf{KV},\mathsf{H})}(\sigma)]_H \Box_H \sigma @ H \Big) \tag{33}$$

The health insurance receives a signed invoice only through the deposit operation.

$$\bigwedge_{D \in \{D_1, D_2\}} \quad \bigwedge_{\sigma \in \Sigma(D)} \Big(\neg \sigma @ H \; \mathcal{W}_H^{+} \langle \mathrm{dep}_{(\mathsf{KV},\mathsf{H})}(\sigma) \rangle_H \top \Big) \tag{34}$$

The KV can only deposit those invoices at the health insurance that it holds. The health insurance accepts an invoice only if it can verify that the signer of the invoice belongs to group G. To verify this, the insurance must hold the public group key for invoices, $\mathtt{pi}_\mathsf{G}$.

$$\begin{aligned} & \bigwedge_{D \in \{D_1, D_2\}} \quad \bigwedge_{\sigma \in \Sigma(D)} \\ \Big(\quad & \Box_{\mathsf{KV}} \Big(\neg \sigma @ \mathsf{KV} \Rightarrow [\mathrm{dep}_{(\mathsf{KV},\mathsf{H})}(\sigma)]_{\mathsf{KV}} \bot \Big) \\ \wedge \quad & \Box_H \Big(\neg \mathtt{pi}_\mathsf{G} @ H \Rightarrow [\mathrm{dep}_{(\mathsf{KV},\mathsf{H})}(\sigma)]_H \bot \Big) \Big) \end{aligned} \tag{35}$$

The health insurance always suspects deception, if it receives the same invoice more than once.

$$\begin{aligned} & \bigwedge_{D \in \{D_1, D_2\}} \quad \bigwedge_{\sigma \in \Sigma(D)} \\ & \Box_H \Big(\sigma @ H \Rightarrow [\mathrm{dep}_{(\mathsf{KV},\mathsf{H})}(\sigma)]_H \Big(\big(\Box_H [\mathrm{reimb}_{(\mathsf{H},\mathsf{KV})}(\sigma)]_H \bot \big) \wedge \Diamond_H \langle \mathrm{req}_{(\mathsf{H},\mathsf{KV})}(\sigma) \rangle_H \top \Big) \Big) \end{aligned} \tag{36}$$

If the health insurance requests the identity of the signer of an invoice from the KV, the KV reveals the signer's identity.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{\sigma\in\Sigma(D)} \Box_{KV}\Big([\mathrm{req}_{(H,KV)}(\sigma)]_{KV} id(D,\sigma)@H\Big) \tag{37}$$

The KV needs its public and secret group keys for invoices, pi_G and si_G to verify the identity of the issuer.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{\sigma\in\Sigma(D)} \neg\big(pi_G@KV \wedge si_G@KV\big) \Rightarrow [\mathrm{req}_{(H,KV)}(\sigma)]_{KV}\bot \tag{38}$$

Clearing – Pharmacy

To claim reimbursement from the health insurance, the pharmacy encloses the transcript of a show operation into an invoice and signs the invoice. After the signature operation, it holds the signed invoice.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{ps,ps'\in Ps} \Box_F\Big([\mathrm{sign}_F(\mathrm{inv}(t^M(ps,D,ps')))]_F \sigma_F^{\mathrm{inv}}(t^M(ps,D,ps'))@F\Big) \tag{39}$$

The sign operation is the only operation by which the pharmacy can get a signed invoice.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{ps,ps'\in Ps} \Box_F\Big(\neg\sigma_F^{\mathrm{inv}}(t^M(ps,D,ps'))@F \Rightarrow \big(\neg\sigma_F^{\mathrm{inv}}(t^M(ps,D,ps'))@F\ \mathcal{W}_F^{+}\langle \mathrm{sign}_F(\mathrm{inv}(t^M(ps,D,ps')))\rangle_F\big)\Big) \tag{40}$$

The pharmacy needs its secret key to sign an invoice. It can only enclose those transcripts into an invoice that it holds.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{ps,ps'\in Ps} \wedge\quad \Box_F\Big(\neg(sk_F@F \wedge t^M(ps,D,ps')@F) \Rightarrow [\mathrm{sign}_F(\mathrm{inv}(t^M(ps,D,ps')))]_F\bot\Big) \tag{41}$$

The pharmacy claims reimbursement by depositing her signed invoice at the health insurance. It needs to hold the invoice in order to deposit it.

$$\bigwedge_{\sigma\in\Sigma(F)} \Big(\wedge\Box_F\big(\neg\sigma@F \Rightarrow [\mathrm{dep}_{(F,H)}(\sigma)]_F\bot\big)\Big) \tag{42}$$

The health insurance receives a signed invoice from the pharmacy by a deposit operation. It holds a received invoice forever.

$$\bigwedge_{\sigma\in\Sigma(F)} \Box_H\Big([\mathrm{dep}_{(F,H)}(\sigma)]_H(\Box_H\sigma@H)\Big) \tag{43}$$

The health insurance receives a signed invoice from the pharmacy only through a deposit operation.

$$\bigwedge_{\sigma\in\Sigma(F)} \Big(\neg\sigma@H\ \mathcal{W}_H^+\langle\mathrm{dep}_{(F,H)}(\sigma)\rangle_H\top\Big) \tag{44}$$

The health insurance accepts an invoice from the pharmacy only, if it receives the invoice for the first time, if it can verify the signature, i.e. it holds the public key of the pharmacy, and if it can verify the group of the issuer of the prescription, i.e. it holds the public prescription key of group G.

$$\bigwedge_{\sigma\in\Sigma(F)} \Box_H\Big(\big(\sigma@H \vee \neg pk_F \vee \neg pp_G\big) \Rightarrow [\mathrm{dep}_{(F,H)}(\sigma)]_H\bot\Big) \tag{45}$$

From an invoice of the pharmacy, the health insurance learns the group of the physician that has issued the corresponding prescription.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{ps,ps'\in Ps} \Box_F\big(\sigma_F^{\mathrm{inv}}(t^M(ps,D,ps'))@H \Rightarrow \mathrm{group}(G,\sigma_F^{\mathrm{inv}}(t^M(ps,D,ps')))@H\big) \tag{46}$$

After the health insurance has received a signed invoice from the pharmacist, it will eventually reimburse the pharmacist. For each signed invoice, it will reimburse the pharmacy only once.

$$\bigwedge_{\sigma\in\Sigma(F)} \Box_H\Big([\mathrm{dep}_{(F,H)}(\sigma)]_H \Diamond_H[\mathrm{reimb}_{(H,F)}(\sigma)]_H\top\Big) \tag{47}$$

7.2.3 Semantic Constraints – Epistemic Behavior

In the previous section we have modeled semantic constraints about the temporal behavior. Each agent knows some of the specified constraints. This section shows how we can model the effects of operations on the knowledge of agents. In this context it is important to remember that the logic $\mathcal{L}$ does not assume perfect recall of agents. This means, that agents can forget information by performing operations. The following formulae are characteristic for perfect recall and **do not hold** in general:

$$K_i[a]_i\phi \Rightarrow [a]_iK_i\phi$$
$$K_i\Box_i\phi \Rightarrow \Box_iK_i\phi$$

We will not model the change of knowledge for all operations as this would imply a large number of formulae without giving us more insight into how these constraints can be modeled. Instead we restrict ourselves to only few examples.

Epistemic effects of publishing keys In equation (6) we formalized that after the publish operation, the agent to whom the key was published, holds the key forever. We assume that this constraint is always known to the publisher:

$$\bigwedge_{k\in pubkey} \bigwedge_{\substack{i,j\in Ag\\ i\neq j}} \Box_i\left(K_i\left(\Box_i\left([\mathrm{pub}_{(i,j)}(k)]_i\ (\Box_j k@j)\right)\right)\right) \tag{48}$$

Since we do not assume in our framework that agents have perfect recall, the above constraint does not imply that after publishing a key to agent j, agent i always know that agent j holds the key forever. For example, without having perfect recall, agent i may simply forget that it has published the key to agent j. So, in our framework, we have to explicitly formalize that agent i does not forget that she has published the key to another agent:

$$\bigwedge_{k \in \text{pubkey}} \bigwedge_{\substack{i,j \in Ag \\ i \neq j}} \Box_i \, [\text{pub}_{(i,j)}(k)]_i \left(\left(K_i(\Box_j k@j) \right) \wedge \left(\Box_i(K_i k@j) \right) \right) \tag{49}$$

Note, that the above formula does *not* belong to the restricted language $\mathcal{L}^c$ for which the tableau system is complete: Agent i *knows* that something local to agent j holds. However, agent i does in practice have no control over the fact local to j. In our scenario, it might for example happen that agent j's system crashes and will delete all the keys. Agent i wouldn't know that, so how can agent i *know* that agent j holds the key?

Another way to model the constraint which says that the publishing agent will never forget, that the agent to whom she has published a key will hold the key forever is by specifying that the publishing agent keeps "a record" of this operation. To specify this, we introduce a new proposition k^j for each public key k and for each agent j.

$$\bigwedge_{k \in \text{pubkey}} \bigwedge_{\substack{i,j \in Ag \\ i \neq j}} \begin{pmatrix} & \Box_i \, [\text{pub}_{(i,j)}(k)]_i \, \Box_i k^j@i \\ \wedge & \Box_i\left(k^j@i \Rightarrow K_i k@j\right) \\ \wedge & \neg k^j@i \; \mathcal{W}_i^+ \langle \text{pub}_{(i,j)}(k) \rangle_i \top \end{pmatrix} \tag{50}$$

Depending on the point of view, we might favor either the first or the second scenario.

We can further assume that whenever some agent j holds a public key of another agent, agent j believes that the other agent holds the public key as well and also holds the corresponding secret key (we assume that every actor knows that there exists exactly one group G of physicians which includes both physicians). (Note, that we distinguish between "agent i believes that agent j holds a key" and "agent i *holds* the key". If we, for example, formulate $B_j sk_H@H$ then this means that agent j believes that agent H holds the secret key named by sk_H. This, however, does not mean that agent j believes the value of the key sk_H and can thus *use* the key. If we wanted to express that agent j believes the value of the key, then we would have to formulate $B_j sk_H@B$.

$$\bigwedge_{j\in Ag} \Box_j\Big(\begin{array}{ll} pk_H@j & \Rightarrow \Box_j\big(B_j(pk_H@H \wedge sk_H@H)\big) \\ \wedge \quad pk_F@j & \Rightarrow \Box_j\big(B_j(pk_F@F \wedge sk_F@F)\big) \\ \wedge \quad pp_G@j & \Rightarrow \Box_j\big(B_j\big((pp_G@D_1 \wedge sp_G@D_1) \vee (pp_G@D_2 \wedge sp_G@D_2)\big)\big) \\ \wedge \quad pp_G@j & \Rightarrow \Box_j\big(B_j\big(pp_{D_1}@D_1 \wedge pp_{D_2}@D_2 \wedge sp_{D_1}@D_1 \wedge sp_{D_2}@D_1\big)\big) \\ \wedge \quad pp_G@j & \Rightarrow \Box_j\big(B_j\big(pp_G@KV \wedge sp_G@KV\big)\big) \\ \wedge \quad pi_G@j & \Rightarrow \Box_j\big(B_j\big((pi_G@D_1 \wedge si_G@D_1) \vee (pi_G@D_2 \wedge si_G@D_2)\big)\big) \\ \wedge \quad pi_G@j & \Rightarrow \Box_j\big(B_j\big(pi_{D_1}@D_1 \wedge pi_{D_2}@D_2 \wedge si_{D_1}@D_1 \wedge si_{D_2}@D_1\big)\big) \\ \wedge \quad pi_G@j & \Rightarrow \Box_j\big(B_j\big(pi_G@KV \wedge si_G@KV\big)\big)\Big) \end{array} \tag{51}$$

Epistemic constraints concerning i**-certificates** The health insurance always knows that each I-certificate is issued only to one policy holder:

$$\bigwedge_{ps\in Ps} \Box_H\Big(K_H \neg\big(I(ps)@P_1 \wedge I(ps)@P_2\big)\Big) \tag{52}$$

The health insurance knows that after each issue operation, the participating policy holder holds one more I-certificate. Let β_{issue} denote formula (14). Then this constraints can be formalized as

$$\Box_H\ K_H\beta_{\text{issue}} \tag{53}$$

7.2.4 Specification of Security Constraints

The introduced example of a privacy oriented clearing scheme has to satisfy a number of security constraints which will be discussed in this section. According to the introduction we can distinguish among various aspects of security constraints:

- Secrecy: We have to prevent the improper disclosure of data.
- Integrity: We have to prevent the improper modification of data.

- Availability: We have to prevent denial of service.

We can find all these three aspects in our example system. The main purpose of the system as it is described in [BS97a], [BS97b] is to maintain as much privacy for all participants of the health care system as possible on the one hand and to ensure as much "control" as is necessary on the other hand. For example, though the health insurance must know that an invoice of a physician is "correct" and that the treated patient is actually insured by the health insurance, it does not need to know the identities of the issuing physician or of the patient/policy holder.

We will now discuss a number of security constraints of the system concerning the three aspects mentioned above and show how they can be formalized in our specification language:

Secrecy The main secrecy requirement of the described health care system is that **the health insurance shall not know which physician has treated which policy holder** without the help of the KV and unless the policy holder has misbehaved and shown an I-certificate twice.

This main secrecy constraint can be divided into basically two simpler constraints:

1. **The health insurance does not know the identity of the holder of an I-certificate unless the holder has misbehaved and shown an I-certificate twice.**

   We can even strengthen this constraint: The health insurance does not know the identity of the holder of an I-certificate unless **it knows** that the holder has misbehaved and shown an I-certificate twice.

   Our logic does not provide any temporal past operators. We thus introduce two new propositions local to the policy holder. The new proposition $\mathtt{I}^{\mathtt{u}}(\mathtt{ps})$ is true, if and only if the I-certificate with pseudonym $\mathtt{ps}$ has already been shown. The new proposition $\mathtt{ds}(\mathtt{ps})$ (double-show) is true if and only if the policy holder has shown an I-certificate at least twice.

   When a policy holder shows an I-certificate to a physician or pharmacy, she remembers, which I-certificate she has used. Showing it to a physician or pharmacy is the only way to use it and to remember that it is used.

$$\begin{aligned}
\alpha_1(P, ps) ::= &\Box_P\Big(\bigwedge_{D\in\{D_1,D_2\}} [\mathrm{show}_{(P,D)}(ps)]_P \Box_P I^u(ps)@P \\
&\wedge \big(\bigwedge_{D\in\{D_1,D_2\}} [\mathrm{gShow}_{(P,F)}(ps, D, ps)]_D \Box_P I^u(ps)@P\big)\Big) \\
\alpha_2(P, ps) ::= &\neg I^u(ps)@P\ \mathcal{W}_P^+ \\
&\Big(\bigvee_{D\in\{D_1,D_2\}} \langle \mathrm{show}_{(P,D)}(ps)\rangle_P \top \\
&\vee \big(\bigvee_{D\in\{D_1,D_2\}} \langle \mathrm{gShow}_{(P,F)}(ps, D, ps)\rangle_D \top\big)\Big)
\end{aligned}$$

If a policy holder shows a used I-certificate again to some physician or pharmacy, she has double-shown it. The only way to double-show an I-certificate is to show a used certificate again.

$$\begin{aligned}
&\alpha_3(P, ps) ::= \\
&\Box_P I^u(ps)@P \Rightarrow \bigwedge_{D\in\{D_1,D_2\}} \Big([\mathrm{show}_{(P,D)}(ps)]_P \big(\Box_P ds(ps)@P\big) \\
&\qquad \wedge [\mathrm{gShow}_{(P,F)}(ps, D, ps)]_D \big(\Box_P ds(ps)@P\big)\Big)
\end{aligned}$$

$$\begin{aligned}
&\alpha_4(P, ps) ::= \\
&\neg ds(ps)@P\ \mathcal{W}_P^+\Big(I^u(ps)@D \wedge \\
&\qquad \bigvee_{D\in\{D_1,D_2\}} \big(\langle \mathrm{show}_{(P,D)}(ps)\rangle_P \top \vee \langle \mathrm{gShow}_{(P,F)}(ps, D, ps)\rangle_D \top\big)\Big)
\end{aligned}$$

The health insurance always knows the constraints $\alpha_1(P, ps), \alpha_3(P, ps), \alpha_2(P, ps), \alpha_4(P, ps)$.

$$\bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{ps\in Ps} \Box_H\Big(K_H\big(\alpha_1(P, ps) \wedge \alpha_3(P, ps) \wedge \alpha_2(P, ps) \wedge \alpha_4(P, ps)\big)\Big) \tag{54}$$

Now we formalize the constraint that the health insurance knows the identity of the holder of an I-certificate if and only if the holder has misbehaved as follows:

$$
\begin{aligned}
&\bigwedge_{P \in \{P_1,P_2\}} \bigwedge_{ps \in Ps} \\
&\quad \Box_H\Big(K_H\big(I(ps)@P \wedge ds(ps)@P\big) \Rightarrow K_H\big(id(P,ps)@H\big)\Big) \\
\wedge &\quad \Box_H\Big(K_H\big(id(P,ps)@H\big) \Rightarrow K_H\big(I(ps)@P \wedge ds(ps)@P\big)\Big)
\end{aligned}
\tag{55}
$$

Note again, that this specification uses the full language $\mathcal{L}$ and is not covered by the restricted language $\mathcal{L}^c$.

If we argue that the health insurance cannot actually know that the policy holder has misbehaved, then we could again introduce an new proposition that intuitively says: “the health insurance keeps a note that is suspects the policy holder to have misbehaved”. This suspicion can then be verified with the policy holder.

2. **The health insurance always knows the identity of the signer of an invoice and identity of the issuer of a prescription only with the knowledge of the KV.**

   We would naively (and wrongly) wish to formalize (the first part of) the above constraint as follows:

$$
\begin{aligned}
&\bigwedge_{D \in \{D_1,D_2\}} \bigwedge_{ps \in Ps} \\
&\ \Box_H(K_H id(D, \sigma_D^{inv}(t^I(ps)))@H \Rightarrow K_{KV}(K_H id(D, \sigma_D^{inv}(t^I(ps)))@H)) \\
&\wedge \Box_{KV}(K_H id(D, \sigma_D^{inv}(t^I(ps)))@H \Rightarrow K_{KV}(K_H id(D, \sigma_D^{inv}(t^I(ps)))@H))
\end{aligned}
$$

   First of all, this formalization is syntactically not correct: The temporal modality $\Box_i$ requires a formula local to agent i and the sub-formulae that follow the box operators in the formula above are of type $\{H, KV\}$. However, also semantically, it is questionable to say “always in the future of health insurance H something must hold for the KV”. Suppose, for example a configuration local to health insurance H, in which the health insurance does not know the identity of the signer. Then the health insurance performs some operation without the participation of other agents. After the operation, it knows the identity. From the point of view of the KV, both configurations before and after the operation are equivalent. So, how can the KV gain knowledge about some other agent, if it cannot even distinguish these two configurations?

   We reformulate the first part of the constraint as follows: Both the health insurance and the KV always know that the health insurance knows the identity of the signer of an invoice only with the knowledge of the KV.

$$
\begin{aligned}
\bigwedge_{P\in\{D_1,D_2\}} \bigwedge_{ps\in Ps} \quad & \Box_H\Big(&& K_H\Big(K_H\mathrm{id}(D,\sigma_D^{\mathrm{inv}}(t^I(ps)))@H \Rightarrow \\
& && K_{KV}\big(K_H\mathrm{id}(D,\sigma_D^{\mathrm{inv}}(t^I(ps)))@H\big)\Big)\Big) \\
\wedge \quad & \Box_{KV}\Big(&& K_{KV}\Big(K_H\mathrm{id}(D,\sigma_D^{\mathrm{inv}}(t^I(ps)))@H \Rightarrow \\
& && K_{KV}\big(K_H\mathrm{id}(D,\sigma_D^{\mathrm{inv}}(t^I(ps)))@H\big)\Big)\Big)
\end{aligned}
\tag{56}
$$

The second part of the constraint, namely that the health insurance always knows the identity of the issuer of a prescription only with the knowledge of the KV can be formalized analogously:

$$
\begin{aligned}
\bigwedge_{P\in\{D_1,D_2\}} \bigwedge_{ps,ps'\in Ps} \quad & \Box_H\Big(&& K_H\Big(K_H\mathrm{id}(D,t^M(ps,D,ps'))@H \Rightarrow \\
& && K_{KV}\big(K_H\mathrm{id}(D,t^M(ps,D,ps))@H\big)\Big)\Big) \\
\wedge \quad & \Box_{KV}\Big(&& K_{KV}\big(K_H\mathrm{id}(D,t^M(ps,D,ps'))@H \Rightarrow \\
& && K_{KV}\big(K_H\mathrm{id}(D,t^M(ps,D,ps'))@H\big)\big)\Big)
\end{aligned}
\tag{57}
$$

Integrity We have already encountered a number of integrity constraints in section 7.2.2. Here are some examples:

- A physician does only participate in a show operation, if she has not already "seen" the shown pseudonym before (see constraint (17)).
- A physician issues prescriptions only to those policy holders that are currently under treatment (see constraint (22)).
- The pharmacy accepts each I-certificate only once (see constraint (25)).

Sometimes, the participating parties need an explicit authorization or "right" to participate.

1. A policy holder may only request treatment on account of the health insurance from a physician if she is insured by some health insurance.

In our example, being insured by some health insurance is equivalent to having I-certificates of this health insurance. The policy holder proves her right by showing the physician one of her I-certificates.

So, we can formalize the above constraint as follows:

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{ps\in Ps} \Box_P\Big(\neg I(ps)@P \Rightarrow [\text{show}_{(P,D)}(ps)]_P\bot\Big) \tag{58}$$

2. A physician is entitled to provide treatment on account of the health insurance only if she is registered by the health insurance.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{P\in\{P_1,P_2\}}$$
$$\Big([\text{show}_{(P,D)}(ps)]_D\bot \ \mathcal{W}_D\langle\text{register}_{(D,KV,H)}\rangle_P\top\Big) \tag{59}$$

3. Each policy holder shall receive only the prescribed medicaments. In particular, each prescribed medicament shall be received only according to prescription.

   There are several ways how this constraint could be violated: A policy holder could show the prescription together with an I-certificate that does not correspond to the prescription, she could show a prescription more than once, or she could use a forged prescription.

   According to the first possibility to misuse a prescription, we require that the pharmacy accepts a prescription only if the policy holder shows the corresponding I-certificate along with the prescription, i.e. $ps = ps'$:

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{\substack{ps,ps\in Ps\\ ps\neq ps'}} \Big(\Box_F[\text{gShow}_{(P,F)}(ps, D, ps')]_F\bot\Big) \tag{60}$$

   According to the second possibility to misuse a prescription, we require that each policy holder can show the same prescription only once:

$$\bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{ps,ps'\in Ps} \bigwedge_{D\in\{D_1,D_2\}}$$
$$\Box_P\Big([\text{gShow}_{(P,F)}(ps, D, ps')]_P \ \Box_P\big(\bigwedge_{ps''\in Ps}[\text{gShow}_{(P,F)}(ps'', D, ps')]_P\bot\big)\Big)$$

(61)

This constraint is satisfied in our setting: We consider only one pharmacy and according to constraint (23) this pharmacy always remembers, which I-certificates it has already been shown. According to constraint (25) the pharmacy does not accept the same I-certificate twice. Now, if the pharmacy accepts only prescriptions with corresponding I-certificates, as requested in constraint (60), then the policy holder can not show the same prescription more than once to the same pharmacy.

Now consider the case of two pharmacies F_1 and F_2, each working in the same way as described above, which means, all the constraints (23), (24), (26), (25), and from (39) to (45) apply for both F_1 and F_2. Then the constraint that a policy holder shall not be able to show the same prescription more than once can be formalized similar as above:

$$\alpha := \bigwedge_{P \in \{P_1, P_2\}} \bigwedge_{ps, ps' \in Ps} \bigwedge_{D \in \{D_1, D_2\}} \bigwedge_{F, F' \in \{F_1, F_2\}} \Box_P \Big([\text{gShow}_{(P,F,H)}(ps, D, ps')]_P \Box_P \big(\bigwedge_{ps'' \in Ps} [\text{gShow}_{(P,F',H)}(ps'', D, ps')]_P \bot \big) \Big) \tag{62}$$

As the system is described above, this constraint would not be enforced. It would be possible for a policy holder to show the same prescription twice to different pharmacies. In this case, only the health insurance would identify the policy holder that has used the same prescription twice after the pharmacies have claimed reimbursement and thus sent their invoices to the health insurance. We could for example change the system in a way suggested by Bleumer and Schunter: Each time a policy holder shows a prescription to a pharmacy, the pharmacy first consults the health insurance. If according to the health insurance, the shown I-certificate and prescription have not been shown before, the pharmacy accepts the prescription.

To change the system appropriately, we add the operation $\text{gShow}_{(P,F,H)}(ps, D, ps')$ to the health insurance's operation alphabet. The health insurance remembers the shown I-certificate by proposition $(H, ps)@H$, thus we modify the set of propositions of the health insurance as follows:

$$\mathcal{P}'_H := \mathcal{P}_H \cup \{(H, ps) \mid ps \in Ps\}$$

For the new show operation $\text{gShow}_{(P,F,H)}(ps, D, ps')$ we require all constraints that also apply to operation $\text{gShow}_{(\mathsf{P},\mathsf{F})}(\mathsf{ps}, \mathsf{D}, \mathsf{ps}')$. Further, we require that the health insurance remembers each shown I-certificate and only participates, if the shown I-certificate and the shown prescription correspond, if the I-certificate has not been shown before and if the physician that has issued the prescription belongs to group G of which the health insurance holds the public group key for prescriptions:

$$\begin{aligned} & \bigwedge_{\mathsf{P}\in\{\mathsf{P}_1,\mathsf{P}_2\}} \bigwedge_{\mathsf{D}\in\{\mathsf{D}_1,\mathsf{D}_2\}} \bigwedge_{\mathsf{F}\in\{\mathsf{F}_1,\mathsf{F}_2\}} \bigwedge_{\substack{\mathsf{ps},\mathsf{ps}'\in\mathsf{Ps}\\ \mathsf{ps}\neq\mathsf{ps}'}} \\ & \qquad \Box_{\mathsf{H}}[\text{gShow}_{(\mathsf{P},\mathsf{F},\mathsf{H})}(\mathsf{ps}, \mathsf{D}, \mathsf{ps})]_{\mathsf{H}}(\mathsf{H}, \mathsf{ps})@\mathsf{H} \\ \wedge & \qquad \Box_{\mathsf{H}}[\text{gShow}_{(\mathsf{P},\mathsf{F},\mathsf{H})}(\mathsf{ps}, \mathsf{D}, \mathsf{ps}')]_{\mathsf{H}}\bot \\ \wedge & \qquad \Box_{\mathsf{H}}\Big(\big((\mathsf{H}, \mathsf{ps})@\mathsf{H} \vee \neg\mathsf{pp}_{\mathsf{G}}\big) \Rightarrow [\text{gShow}_{(\mathsf{P},\mathsf{F},\mathsf{H})}(\mathsf{ps}, \mathsf{D}, \mathsf{ps})]_{\mathsf{H}}\bot\Big) \end{aligned} \tag{63}$$

A policy holder shall not be able to claim medicaments on behalf of a forged prescription, i.e. on behalf of a prescription that cannot be verified using the public group key of the physicians' group. Up to now, we have only modeled prescriptions that are issued by physicians. Then the constraint is that the pharmacy must hold the public group key of the issuing physician's group:

$$\begin{aligned} & \bigwedge_{\mathsf{P}\in\{\mathsf{P}_1,\mathsf{P}_2\}} \bigwedge_{\mathsf{D}\in\{\mathsf{D}_1,\mathsf{D}_2\}} \bigwedge_{\mathsf{ps},\mathsf{ps}'\in\mathsf{Ps}} \\ & \Box_{\mathsf{F}}\Big(\neg\mathsf{pp}_{\mathsf{G}}@\mathsf{F} \Rightarrow [\text{gShow}_{(\mathsf{P},\mathsf{F})}(\mathsf{ps}, \mathsf{D}, \mathsf{ps}')]_{\mathsf{F}}\bot\Big) \end{aligned} \tag{64}$$

We could also consider that the policy holder actually holds forged prescriptions that are not issued by a physician. To formulate this we add new propositions to the set of propositions of the policy holder:

$$\mathcal{P}'_{\mathsf{P}} := \mathcal{P}_{\mathsf{P}} \cup \{\mathsf{M}(\mathsf{X}, \mathsf{ps})@\mathsf{P} | \mathsf{ps} \in \{\spadesuit, \diamondsuit, \clubsuit, \heartsuit\}\}$$

The prescriptions that are represented by this new propositions are not issued by anyone inside the system. The policy holder may get them at any time. We introduce an operation that says, the policy holder "receives a forged" prescription "externally":

$$\mathcal{O}'_{\mathsf{P}} := \mathcal{O}_{\mathsf{P}} \cup \{\text{external}_{\mathsf{P}}\}$$

This external operation is the only operation that may cause $\mathsf{M}(\mathsf{X}, \mathsf{ps})$ to be true (though it does not have to), there are no preconditions for this operation to be executed. Initially, we assume that no policy holder has any forged prescriptions.

$$\bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{ps\in Ps} \Big(\neg M(X,ps)@P \\ \wedge \quad \Box_P\big(\neg M(X,ps)@P \Rightarrow \big(\neg M(X,ps)@P\ \mathcal{W}^+_P\langle\text{external}_P\rangle_P\top\big)\big)\Big) \tag{65}$$

The pharmacy uses the public group key to verify, whether a shown prescription is signed by a physician or not. Since none of these "new" prescriptions is signed by a physician, the pharmacy shall not accept any of them:

$$\bigwedge_{P\in\{P_1,P_2\}} \bigwedge_{ps,ps'\in Ps} \Box_F[\text{gShow}_{(P,F)}(M(X,ps),],_{)}F\bot \tag{66}$$

4. The pharmacist may claim reimbursement only for prescriptions, for which the corresponding I-certificate has been shown. We model this by saying that the health insurance (which has to pay for the medicaments) only accepts signed invoice, if the I-certificate and the prescription shown to the pharmacy, are issued on the same pseudonym.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{\substack{ps,ps'\in Ps\\ ps\neq ps'}} \\ \Box_H[\text{dep}_{(F,H)}(\sigma_F^{\text{inv}}(t^M(ps,D,ps')))]_H\bot \tag{67}$$

5. The physician may claim reimbursement only for giving treatment to a policy holder who has registered as a patient.

   This constraint is already ensured by formulae (17) and (29), since we have requested for the show operation that the policy holder holds the I-certificate that she shows, and the physician must holds the transcript of the I-certificate that she includes in an invoice.

Availability

1. After the registration, the registering parties know that they must eventually exchange their cryptographic keys:

$$\bigwedge_{D\in\{D_1,D_2\}} K_H \Box_H [\mathrm{register}_{(D,KV,H)}]_H \Big(\Diamond_H \langle \mathrm{pub}_{(D,H)}(pp_G)\rangle_H \top \wedge \Diamond_H \langle \mathrm{pub}_{(D,H)}(pi_G)\rangle_H \top \wedge \Diamond_H \langle \mathrm{pub}_{(H,D)}(pk_H)\rangle_H \top \Big) \tag{68}$$

2. After the generation of the group keys the KV must eventually publish the public group keys to the corresponding physicians:

$$\begin{aligned} &\Box_{KV} \quad [\mathrm{gen}^{i}_{KV}]_{KV} \left(\Diamond_{KV}\langle \mathrm{pub}_{(KV,D_1)}(pi_G)\rangle_{KV}\top \wedge \Diamond_{KV}\langle \mathrm{pub}_{(KV,D_1)}(pi_G)\rangle_{KV}\top\right) \\ \wedge\ &\Box_{KV} \quad [\mathrm{gen}^{p}_{KV}]_{KV} \left(\Diamond_{KV}\langle \mathrm{pub}_{(KV,D_1)}(pp_G)\rangle_{KV}\top \wedge \Diamond_{KV}\langle \mathrm{pub}_{(KV,D_1)}(pp_G)\rangle_{KV}\top\right) \end{aligned} \tag{69}$$

3. After the health insurance has received an invoice from the KV, it either has to eventually reimburse the physician, i.e. transfer the money to the KV that gives it to the physician or, if it suspects deception, must eventually ask the KV to reveal the identity of the signer of the invoice (and notify the respective physician).

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{\sigma\in\Sigma(D)} \Box_H \Big([\mathrm{dep}_{(KV,H)}(\sigma)]_H \Diamond_H \big(\langle \mathrm{reimb}_{(H,KV)}(\sigma)\rangle_H \top \vee \langle \mathrm{req}_{(H,KV)}(\sigma)\rangle_H \top \big) \Big) \tag{70}$$

4. If the KV receives reimbursement from the health insurance, it must eventually forward the money to the corresponding physician.

$$\bigwedge_{D\in\{D_1,D_2\}} \bigwedge_{\sigma\in\Sigma(D)} \Box_{KV} \Big([\mathrm{reimb}_{(H,KV)}(\sigma)]_{KV} \Diamond_{KV} \langle \mathrm{reimb}_{(KV,D)}(\sigma)\rangle_{KV} \top \Big) \tag{71}$$

5. If the health insurance accepts an invoice of the pharmacist, it must eventually

reimburse the pharmacist.

$$\bigwedge_{\sigma \in \Sigma(F)} \Box_H \, [\mathrm{dep}_{(F,H)}(\sigma)]_H \Diamond_H \langle \mathrm{reimb}_{(H,F)}(\sigma) \rangle_H \top \tag{72}$$

6. If a policy holder claims medicaments from a pharmacy by showing a prescription and the corresponding I-certificate, the health insurance must eventually deliver the medicament.

$$\begin{aligned} &\bigwedge_{P \in \{P_1,P_2\}} \bigwedge_{D \in \{D_1,D_2\}} \bigwedge_{ps \in Ps} \\ &\Box_P \left([\langle \mathrm{gShow}_{(P,F)}(ps, D, ps) \rangle]_P \Diamond_P \langle \mathrm{deliver}_{F,P}(ps) \rangle_P \top \right) \end{aligned} \tag{73}$$

7. If a physician D sends her public key pi_D or pp_D to the KV, the KV must eventually use the public key to generate a pair of group keys.

$$\begin{aligned} \bigwedge_{D \in \{D_1,D_2\}} \Box_{KV} \Big(\quad & [\mathrm{pub}_{(D,KV)}(pi_D)]_{KV} \, \Diamond_{KV} \langle \mathrm{gen}^i_{KV} \rangle_{KV} \top \\ \wedge \; & [\mathrm{pub}_{(D,KV)}(pp_D)]_{KV} \, \Diamond_{KV} \langle \mathrm{gen}^p_{KV} \rangle_{KV} \top \Big) \end{aligned} \tag{74}$$

8. If the KV generates a pair of group keys pi_G, si_G or pp_G, sp_G, the public group keys pi_G or pp_G respectively, must eventually be sent to the health insurance.

$$\begin{aligned} \Box_{KV} \Big(\; & [\mathrm{gen}^i_{KV}]_{KV} \Diamond_{KV} \langle \mathrm{pub}_{(KV,H)}(pi_G) \rangle_{KV} \top \\ \wedge \; & [\mathrm{gen}^p_{KV}]_{KV} \Diamond_{KV} \langle \mathrm{pub}_{(KV,H)}(pp_G) \rangle_{KV} \top \Big) \end{aligned} \tag{75}$$

Modeling rights In our example, execute rights for operations are modeled by propositions: A policy holder can for example only register at a physician by showing an I-certificate to the physician. To do so, show of course needs an I-certificate that has before been issued by the health insurance. Issuing an I-certificate to a policy holder can be seen as granting the right to request treatment from a physician on account of the health insurance.

To request medicaments for a pharmacy, the policy holder needs a prescription that has before been issued by a physician. As long as the policy holder does not hold a prescription (and a corresponding I-certificate), she cannot perform the show operation together with the pharmacy - and may not receive medicaments from the pharmacy. Issuing a prescription to a policy holder can be seen as granting the right to request medicaments from a pharmacy.

Similarly, secret keys can be seen as execute rights: The secret invoice key of the pharmacy for example allows, and enables, the pharmacy to sign invoices. The public group key and the secret individual key allow and enable a physician to sign an invoice with the group signature, the public group key for prescriptions and a physicians individual key for prescriptions enable a physician to issue a prescription to a policy holder. And finally, the secret key of the health insurance enables the health insurance to issue I-certificates.

When a right already granted is to be revoked, this can also be done by an operation performed together of the agent that revokes the right and the agent from which the right is revoked. The possible consequences of a revocation operation (change of truth values of propositions, restrictions about the behavior of the agent of which the right has been revoked) must be explicitly modeled in the provided language.

7.3 Features and Limitations of the Framework

In the previous section we have shown how the various types of semantic constraints and security constraints can be encoded in our framework.

The example system is seen as a distributed system in which each actor (e.g. the health insurance, physician, policy holder, etc.) performs operations sequentially. Operations by different actors are either performed concurrently (e.g. generation of keys) or jointly (e.g. issuing I-certificates or issuing prescriptions). These distribution aspects are naturally encoded in our framework as each acting site of the information system is seen as an agent.

(Execute) rights for operations are modeled by local propositions. The value of propositions is changed through operations.

Knowledge of each agent is determined by the agent's local state, change of knowledge is due to operations performed by the agent.

Reasoning about past behavior Consider again the constraint that the health insurance knows the identity of a pseudonym `ps` only if it knows that the corresponding policy holder has shown the I-certificate issued on pseudonym `ps` twice. It would be convenient to formalize something like: If the health insurance knows the identity of a pseudonym, then it knows that at some time in the past, the corresponding policy holder has shown the I-certificate for a second time. This, however, is not possible in our framework. Instead, we introduced a new proposition local to the policy holder that is only true after an I-certificate has been shown twice. Then we can formalize that the health insurance knows the identity of the pseudonym only if it knows that this proposition local to the policy holder is true. This means that even if we can rule out models, in which the health insurance knows the identity of a pseudonym without the corresponding policy holder having shown the I-certificate issued on this pseudonym twice, we can not formalize that the health insurance knows that the policy holder misbehaved in the past.

Perfect Recall As already discussed in the previous section, we do not assume in our framework that agents have perfect recall. Perfect recall implies a strong interaction between knowledge and action and usually adds to the complexity of the framework.

While specifying secrecy constraints, assuming perfect recall might be desirable: When we want to ensure that secrets are kept from an agent, it makes sense to assume that an agent always remembers all her knowledge already acquired in the past and all the operations performed in the past. In our framework, we must explicitly state for all information the agent acquires that she will never forget this information in the future. In subsection 7.2.3 we demonstrated two possibilities how this can be specified.

However, assuming that agents have perfect recall is not always desirable. When specifying availability constraints it makes sense to assume, that agents may forget information they already acquired. An agent can for example model a human administrator who has to perform certain action at the end of each calendar year (e.g. archiving of old data) to ensure the correct functioning of the system. Though humans do not exhibit perfect recall the correct functioning of the system is still desired. If this system is modeled in our framework, the annual action must be ensured by the specification of the system without the assumption that the agent representing the administrator has perfect recall.

Real World View In many frameworks, knowledge of agents is related to the real world. When an agent knows some fact, this fact must hold in the real world (in the actual real world as well as in all worlds which the agent considers to be possible). However, in our setting we do not model the real world. Agents' knowledge and belief is modeled only in relation to the facts agents 'hold on their desk'. If we wanted to relate knowledge about the real world, we would have to introduce a representing agent for the real world inside the system that explicitly synchronizes with the system for example through a global clock (again modeled as agent of the system). As long as there is no explicit synchronization with the "real-world", we could view agents as sitting in a black box. They can only reason about the system "inside the box", but since they do not notice changes in the real world outside the system (box) it is in our opinion difficult to reason about "the actual state" of the real world.

Frame Problem As it is well-known, any attempt to formalize local effects of operations somehow has to deal with the non-effects of operations, too. This requirement is particularly important for security considerations. The problem is known as the frame problem and was first mentioned by [MH69, Sha97]. Due to the law of inertia, only few facts, represented as propositions in our approach, are changed by an action, most of the facts remain unchanged.

We approach this problem as follows: For every operation we define a formula that specifies the effects of the formula. For every proposition, we define two formulae, one that specifies through which operations the value of the proposition may be toggled from true to false, and another one to specify, when the value of the proposition may be toggled from false to true. For each proposition we specify that if a proposition is true, it will remain true until some particular operation will be performed. And if the proposition is false, it will remain false, until some operation will be performed that changes the value of the proposition from false to true. Note, that due to definition 3.3.2(4) of the accessibility relation for knowledge an operation can only change those propositions that are in the proposition alphabet of agents participating in the operation.

In our example, we defined formulae for every operation that specify the effects of the operation, e.g. the interpretation of which propositions is changed. For each proposition, we identified a set of operations that may change the interpretation of the proposition and then specified the value will remain unchanged *until* a certain operation is performed which may change the interpretation of the proposition.

We also approach this frame problem by the definition 3.3.2(4) of the accessibility relation for knowledge. This definition ensures that the knowledge of an agent cannot be affected by operations performed by other agents. This definition is related to various solutions to the frame problem: In [CGH99] Castilho et al combine a logic of actions and plans with an explicit notion of dependence/independence between propositions and actions. An action can only affect the truth value of a proposition if the proposition is dependent on the action. In [SL93] Levesque and Scherl propose successor state axioms as a solution to the frame problem for epistemic fluents. An "accessibility" predicate K for situations and a knowledge predicate *Knows* is defined. The successor state axiom then requires that two situations s_1 and s'_1 are "accessible" (in terms of this predicate) and situation s_1 leads to a successor situation s_2 via an action a, iff action a performed in situation s'_1 leads to a successor state s'_2 which is "accessible" from situation s_2.

Chapter 8

Conclusion and Outlook

8.1 Summary

In this thesis, we developed a unifying framework for specifying semantic constraints and security constraints in information systems.

After the motivation, we studied various types of semantic constraints and security constraints that can occur in information systems and developed some basic terminology in chapter 1.

In chapter 2 we discussed the advantages and disadvantages of the related work with respect to our problem and justified the need for a new approach.

In chapter 3, we captured our view of a distributed information system in a computational model. In this model, we uniformly represented each component (actors and objects) of the information system as a sequential agent. Each agent can perform operations either autonomously or jointly with other agents as synchronization operations. Each agent is affiliated with a set of operations and with a set of propositions. To capture agents' knowledge and belief, we defined for each agent one indistinguishability relation each for knowledge and for belief.

In chapter 4, we developed a temporal and epistemic logic, the semantics of which is defined on the computational model described above. The logic contains a local next-operator and a local until-operator as temporal operators and a knowledge and a belief operator for each agent as epistemic operators.

In chapter 5, we defined a tableau based proof system for satisfiability for a subset of this logic. This subset does not contain any epistemic operators for belief. The main difficulty of the tableau system arose through the interaction between operations and knowledge of agents. In the computational model we required that the enabling of an operation as well as its effects only depend on the knowledge (and configuration) of the participating agents. Further, an operation cannot have any effect on the knowledge of non participating agents. Because of this restriction a kind of "global view" on the whole structure under construction is needed to detect and repair violations of this restriction. To solve this problem, we defined a kind of explicit tableau. We extended the logical language such that we are able to directly talk about the structure of the computational model within the language.

We showed soundness and completeness of this proof system in chapter 6.

Finally in chapter 7, we gave a specification example by modeling a secrecy oriented approach to the German health care billing and clearing system introduced by Bleumer and Schunter in [BS97a]. We interpreted their system as a distributed information system and specified the most important semantic constraints and security constraints that occur in the system.

8.2 Competency of the Framework

The aim of this work was to develop a unifying framework for the specification of semantic constraints and security constraints. In section 1.3 we analyzed the problem and identified the requirements of the framework. In the following we will briefly discuss how far our framework meets these requirements.

States and (Sequences of) Updates We required that our framework has a notion of states and sequences of updates to be able to model static semantic constraints that restrict possible states of the system and dynamic semantic constraints that restrict possible (sequences of) updates of the system. In our framework, we view the information system as a distributed system and do not assume a global state of the system but model local states of each component. Each component is represented by an agent and the local state of a component is modeled by the interpretation of the agent's local propositions in the agent's local situation. Updates are modeled by joint operation of agents, as we will see below on page 221. We can formalize statements about sequences of updates through temporal next operators in the logic.

Actor's Knowledge and Execute Rights for Read Operations We required that our framework must have a notion of actor's knowledge and execute rights for read operations. Actor's knowledge about facts she has read from the database is represented by propositions local to the agent representing the actor. Actor's knowledge about the behavior of the system and about the states of other agents can be formalized by epistemic operators within the language as we have seen in the example. Execute rights for read operations are also modeled by propositions local to the agent representing the reading actor. The read operation can be performed in a situation only if the proposition representing the corresponding execute right is true in this situation.

Execute Rights for Write Operations We required that our framework has a notion of execute rights for write operations. Analogous to execute rights for read operations, execute rights for write operations are modeled by propositions local to the agent representing the writing actor. A write operation can be performed in a situation only if the proposition representing the corresponding execute right is true in this situation.

Time We required that our framework has a notion of time. The introduced logic is a temporal logic with local next and until operators. Each agent performs operations along a local time line. Timeliness of different agents are synchronized via synchronization operations performed by them. The logic allows to make temporal statements about multiple local timeliness.

Actors and Data-Objects We required a formalism for actors data-objects in our framework. Each actor as well as each data-object is uniformly modeled as a sequential agent that performs operations. Stored data is modeled by the interpretation of local propositions.

Read, Update and Authorization Operations We required three types of operations, read operations or queries, update operations and authorization operations that our framework has to capture. A read operation is an operation in which an actor queries the database and updates her knowledge with the information received from the database. An update operation is an operation in which an actor updates an object of the database and an authorization operation is an operation in which an actor grants a right to or revokes a right from another actor. In our framework we do not distinguish among these types of operations in general. Read operations, update operations and authorization operations are uniformly modeled as synchronization operations between a group of agents.

An update operation is for example modeled by a joint operation of the user-agent representing the actor and the repository-agent representing the data object. In the introduced language we then formalize how this joint operation changes the local situations of the participating agents by defining the meaning of the operation over the interpretation of the relevant propositions.

Overview by Example of Constraints The following table gives an overview of the various types of constraints in an information system and relates them to constraints in our specification example formalized in chapter 7. The first column of the table contains the type of the constraint, the second column describes the constraint and the third column relates the constraint to constraints in our example.

Type	Description	Example
static semantic constraints	restrict possible states of the information system	an I-certificate cannot be unused and double shown at the same time the same I-certificate cannot be issued to two policy holders.
dynamic semantic constraints	restrict possible (sequences of) updates	the health insurance can verify the group of the signer of an invoice, only if it has the corresponding group key for invoices. each issue operation increases the number of I-certificates of the participating policy holder by one.
secrecy constraints	restrict the flow of information	The health insurance knows the identity of the signer of an invoice only with the knowledge of the KV. The health insurance knows the identity of the holder of an I-certificate, only if it knows that holder has double shown the I-certificate.
integrity constraints	restricts the possible (sequences of) updates by restricting actors' write access to objects	a policy holder may only request treatment from a physician if she is insured by some health insurance. a physician is entitled to provide treatment on account of the health insurance only if she is registered by the health insurance.

Type	Description	Example
availability constrains	impose obligations on the system	if the health insurance accepts an invoice from the pharmacy, it must eventually reimburse the pharmacy. if the pharmacy accepts the prescription from a policy holder, it must eventually deliver the prescribed medicaments.

8.3 Open Problems

Computational model In our work we follow the traditional approach of modeling agents as logically omniscient, we assume that an agent knows whatever is true in all the situations the agent considers to be possible. This, however, means that all agents know all the consequences of their knowledge and they know all valid formulae. In the context of modeling secrecy this is a reasonable approach: If a secret has to be kept, it is sensible to assume that the reasoning agent has logical omniscience. However, in the context of modeling availability it is more appropriate in some cases to assume that the agents are not logical omniscient: If a system administrator is supposed to take some appropriate action as soon as he knows that an account is being misused it makes sense to assume that the administrator is not logical omniscient. Human administrators may draw faulty conclusions or may not possess the computational power to infer all conclusions. Machine administrators are resource bounded and may not have the capacity to compute all the conclusions.

While modeling belief of agents in an information system the question about belief update and belief revision arises. When an agent gets information about changes in the system she must *update* her beliefs, whereas when she gets new information about a (local) situation in the system, she must *revise* her beliefs. One aim of our system is to investigate consistency of a set of specified constraints. We did not define any belief update or revision strategies in our model. Belief can be revised and updated completely arbitrarily. It might be desired to investigate consistency of a set of constraints under the assumption that the agents have intelligent belief update and revision strategies which we did not treat in our work.

Language The modal operators defined in our language are sufficient to express most of the required semantic constraints and security constraints as we demonstrated in the example in chapter 7. However, our language has following limitations as we mentioned in section 7.3:

Often it is desirable to be able to formalize that an agent reasons about past behavior of other agents. We for example observed this while formalizing security constraints in our example: When the health insurance receives two invoices in which transcripts of the same I-certificate are enclosed, we cannot formalize that the insurance knows that the holder of the I-certificate *has shown* the I-certificate twice. Our language only allows to reason about situations and *future behavior* of agents. In order to be able to formalize the past behavior of an agent it could be useful to introduce temporal past operators.

Our framework only allows to reason about the knowledge of individual agents. The temporal logic introduced by Niebert in [Nie97] allows to reason about temporal behavior of groups of agents. To our knowledge it is still unclear whether adding temporal operators for group of agents provides extra expressiveness. In the context of knowledge, it may be desired to reason about the knowledge of a group of agents. For example, we might want to model that a secret is kept even if two or more agents cooperate and put their knowledge together to build a group. In order to be able to reason about the knowledge of a group of agents it could be useful to introduce a modal epistemic operator for distributed knowledge.

Another limitation of our logic is the absence of an epistemic operator for common knowledge. Common knowledge is for example needed for modeling agreement between agents [FHMV95]. While distributed knowledge usually does not add extra expressiveness to a logic, common knowledge generally does [FHMV95].

Tableau The tableau based proof method defined in chapter 5 is shown to be sound and complete. However, decidability is still an open issue. When constructing an open tableau for a set of formulae, we can construct from an open branch of the tableau a model with a situation that satisfies the set of formulae. There are examples for sets of formulae that can be satisfied in models containing only a finite set of runs while our procedure constructs a model that contains an infinite set of runs. It is still unclear, how to modify our tableau procedure, such that only necessary runs are constructed. It is not clear either, if there exist formulae that require a model containing infinitely many runs.

The tableau procedure of chapter 5 is constructed for a subset of the logic introduced in chapter 4 that does not contain modal operators for belief. It would be interesting to extend the tableau system so that possible interaction between belief, knowledge and action can be studied.

Bibliography

[And97] Ross Anderson, editor. *Personal Medical Information: Security, Engineering and Ethics*. Springer-Verlag, 1997.

[BC96] Mark A. Brown and Jose Carmo, editors. *Deontic Logic, Agency and Normative Systems (DEON)*, Workshops in Computing. Springer, 1996.

[BS97a] Gerrit Bleumer and Matthias Schunter. Datenschutzorientierte Abrechnung Medizinischer Leistungen. *Datenschutz und Datensicherheit*, 2(21):24–35, 1997.

[BS97b] Gerrit Bleumer and Matthias Schunter. Privacy oriented clearing for the german health-care system. In Anderson [And97], pages 175–194.

[BS03] Joachim Biskup and Barbara Sprick. Towards unifying semantic constraints and security constraints in distributed information systems. In Bernhard Thalheim, Klaus-Dieter Schewe, Leopoldo Bertossi, and Gyula Katona, editors, *Semantics in Databases*, volume 2582 of *LNCS*. Springer-Verlag, 2003.

[CD96] Frederic Cuppens and Robert Demolombe. A deontic logic for reasoning about confidentiality. In Brown and Carmo [BC96], pages 66–79.

[CD97] Frederic Cuppens and Robert Demolombe. Modal logical framework for security policies. In Ybigniew Ras and Andrzeij Skowron, editors, *Foundations of Intelligent Systems*, volume 1325 of *Lecture Notes in Artificial Intelligence*, pages 579–589. Springer, October 1997.

[CGH99] Marcos A. Castilho, Olivier Gasquet, and Andreas Herzig. Formalizing action and change in modal logic I: the frame problem. *Journal of Logic and Computation*, 9(5):701–735, 1999.

[CJ94] Jose Carmo and Andrew J.I. Jones. Deontic database constraints and the characterization of recovery. In Andrew J.I. Jones and Marek Sergot, editors, *Second International Workshop on Deontic Logic in Computer Science, Oslo*, pages 56–85, 1994.

[CS98] Jan Chomicki and Gunter Saake, editors. *Logics for Databases and Information Systems*. Kluwer Academic Publisher, 1998.

[CT98] Jan Chomicki and David Toman. *Temporal Logic in Information Systems*, chapter 3, pages 31–70. In Chomicki and Saake [CS98], 1998.

[DG01] Jennifer M. Davoren and Rajeev P. Gore. Bimodal logics for reasoning about continuous dynamics. In Wolter et al. [WWdRZ01].

[DGHP99] Marcello D'Agostino, Dov Gabbay, Reiner Haehnle, and Joachim Posegga, editors. *Handbook of Tableau Methods*. Kluwer Academic Publishers, 1999.

[DMWK96] Frank Dignum, John-Jules Ch. Meyer, Roel Wieringa, and Ruurd Kuiper. A modal approach to intentions, commitments and obligations: Intention plus commitment yields obligation. In Brown and Carmo [BC96], pages 80–97.

[ECSD98] Hans-Dieter Ehrich, Carlos Caleiro, Amilcar Sernadas, and Grit Denker. *Logics for Databases and Information Systems*, chapter 6, pages 167–198. In Chomicki and Saake [CS98], 1998.

[FHMV95] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

[Fit83] Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logic*. D. Reidel Publishing, 1983.

[Fit96] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 2nd edition, 1996.

[GMP92] Janice Glasgow, Glenn MacEwen, and Prakash Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10/3:226–264, August 1992.

[Gor99] Rajeev Goré. *Tableau Methods for Modal and Temporal Logics*, chapter 6, pages 297–396. In D'Agostino et al. [DGHP99], 1999.

[HS98] Michael N. Huhns and Munindar P. Singh, editors. *Readings in Agents*. Morgan Kaufman, 1998.

[JP85] Andrew J.I. Jones and Ingmar Pörn. Ideality, sub-ideality and deontic logic. *Synthese*, 65:275–290, 1985.

[Lam94] Leslie Lamport. A temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994.

[LPRT93] Kamal Lodaya, Rohit Parikh, R. Ramanujam, and P.S. Thiagarajam. A logical study of distributed transition systems. Tutorial handout, Computer Science Dept., Århus University, Århus, Denmark, August 1993. Summerschool in Logical Methods in Concurrency, August 2-13, 1993.

[Mas94] Fabio Massacci. Strongly analytic tableaux for normal modal logics. In Alan Bundy, editor, *Proceeding of the Twelfth International Conference on Automated Deduction (CADE'94)*, volume 814, pages 723–737. Springer, 1994.

[Maz95] Antoni Mazurkiewicz. Introduction to trace theory. In *The Book of Traces*, chapter 1, pages 1–42. World Scientific, 1995.

[MCK96] Glenn MacEwen, Xiao Jun Chen, and Scott Knight. An action-based logic of causality, knowledge, permission and obligation. Technical report, Queen's University, Department of Computing and Information Science, 1996.

[MH69] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In Bernhard Meltzer and Donald Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[MOP89] Antoni Mazurkiewicz, Edward Ochmanski, and Wojciech Penczek. Concurrent systems and inevitability. *Theoretical Computer Science*, 64:281 – 304, 1989.

[MP92] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[Nie95] Peter Niebert. A ν-calculus with local views for systems of sequential agents. In *Mathematical Foundations of Computer Science (MFCS)*, volume 969 of *Lecture Notes in Computer Science*, pages 563–573. Springer, 1995.

[Nie97] Peter Niebert. *A Temporal Logic for the Specification and Verification of Distributed Behavior*. PhD thesis, Universität Hildesheim, 1997.

[NS97] Peter Niebert and Barbara Sprick. A tableau proof system for a mazurkiewicz trace logic with fixpoints. In Didier Galmiche, editor, *Automated reasoning with Analytic Tableaux and Related Methods*, volume 1227 of *LNAI*, pages 291–306, Pont-a-Mousson, France, May 1997. Springer.

[Pen93] Wojciech Penczek. Temporal logics for trace systems: on automated verification. *International Journal of Foundations of Computer Science*, 4(1):31–68, 1993.

[Ram94] R. Ramanujam. Knowledge and the next state modality. In *Proc. Indian National Seminar in TCS*, pages 62–80, The Institute of Mathematical Sciences, Chennai 600 113, 1994.

[Ram96] R. Ramanujam. Local knowledge assertions in a changing world (extended abstract). In *Proceedings Theoretical Aspects of Rationality and Knowledge*, pages 1–17. Morgan Kaufmann, 1996.

[Ram99] R. Ramanujam. View-based explicit knowledge. *Annals of Pure and Applied Logic*, 96:343–368, 1999.

[Rei90] Raymond Reiter. On asking what a database knows. In *Proceedings of the Symposium on Computational Logic*, pages 95–113. Springer, 1990.

[Rei91] Raymond Reiter. *AI and Math theory of computation*. Academic Press, 1991.

[Sha97] Murray Shanahan. *Solving the Frame Problem*. The MIT Press, 1997.

[SL93] Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 689–697, Washington, D.C., USA, 1993. AAAI Press/MIT Press.

[Thi93] P.S. Thiagarajan. A trace based extension of PTL. Tutorial handout, Computer Science Dept., Århus University, Århus, Denmark, August 1993. Summerschool in Logical Methods in Concurrency, August 2-13,1993.

[Thi94] P.S. Thiagarajan. A trace based extension of Linear Time Temporal Logic. In *IEEE Symposium on Logic in Computer Science (LICS)*, volume 9, pages 438–447, 1994.

[vdHM92] Wiebe van der Hoek and John-Jules Ch. Meyer. Making some issues of implicit knowledge explicit. *International Journal on Foundations of Computer Science*, 3(2):193–223, 1992.

[Wei99] Gerhard Weiss, editor. *Multiagent Systems*. The MIT Press, 1999.

[Woo00] Michael Wooldridge. *Reasoning about rational agents*. The MIT Press, Massachusetts Institute of Technology, Cambridge, Massachusetts 02142, 2000.

[WWdRZ01] Frank Wolter, Heinrich Wansing, Maarten de Rijke, and Michael Zakharyaschev, editors. *Advances in Modal Logic*, volume 3. CSLI Publications, 2001.

Appendix A

Notations

Model

$Ag = \{1, \ldots, k\}$	set of agents	Def. 3.1, page 21
ag, i	agent	
$\tilde{\mathcal{O}} = \{\mathcal{O}_1, \ldots, \mathcal{O}_k\}$	distributed set of operations	Def. 3.1, page 22
$\mathcal{O} = \bigcup_{i \in Ag} \mathcal{O}_i$	set of all operations	Def. 3.1, page 22
$\mathcal{O}_i$	set of operations of agent i	
op, a, a_i	operations	
$ag(op)$	set of agents that are involved in operation op	
$\tilde{\mathcal{P}} = \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$	distributed set of propositions	Def. 3.1, page 21
$\mathcal{P} = \bigcup_{i \in Ag} \mathcal{P}_i$	set of all propositions	
$\mathcal{P}_i$	set of propositions local to agent i	
p, p_i	propositions	
$(Ag, \tilde{\mathcal{O}}, \tilde{\mathcal{P}})$	static declarations of an information system	
$\downarrow M$	downward closure	Def. 3.2.1, page 23
E	set of events	Def. 3.2.2, page 23
E_i	set of events of agent i	Def. 3.2.2, page 23
λ	labelling function of events	Def. 3.2.2, page 23
$F_{(Ag, \tilde{\mathcal{O}}, \tilde{\mathcal{P}})}, F$	run of an information system	Def. 3.2.2, page 23
$\mathcal{A}$	a set of runs of an information system	
$\mathcal{C}_F$	set of all configurations of a run F	Def. 3.2.3, page 25
c	Configuration of $\mathcal{C}_F$	Def. 3.2.3, page 25
$\downarrow^i c$	i-view of agent i on configuration c	Def. 3.2.3, page 25
$\downarrow^i (F, c)$	i-view of agent i on situation (F, c)	Def. 3.2.5, page 26
$c \equiv_i c'$	i-equivalence of configurations	Def. 3.2.3, page 25
$(F, c) \equiv_i (F, c')$	i-equivalence of situations	Def. 3.2.5, page 26
$c \xrightarrow{a} c'$	successor-relation on configurations	Def. 3.2.3, page 25

Model (cont'd.)

$(F, c) \xrightarrow{a} (F, c')$	successor-relation on situations	Def. 3.2.5, page 26
R_i^K	accessibility relation for knowledge	Def. 3.3.2, page 29
R_i^B	accessibility relation for belief	Def. 3.3.3, page 30
$\mathcal{I}$	interpretation of a set of runs	Def: 3.3.4, page 31
$\mathcal{M} = (\mathcal{A}, \mathcal{R}^K, \mathcal{I})$	model of an information system	pages 32

Logic

$\mathcal{L}$	temporal and epistemic logic	Def. 4.1, page 35
$\mathcal{L}^c$	logic $\mathcal{L}$ without belief operator and with the restriction for knowledge operator K, that in a formula $K_i\phi$, the sub formula ϕ must be of type $\{i\}$	Def. 6.2.1, page 130
Φ_A	set of typed formulae	page 35
ϕ, ψ, ϕ', ψ'	formulas of $\mathcal{L}$	
$\perp$	formula of $\mathcal{L}$, false	Def. 4.1, page 35
$\top$	formula of $\mathcal{L}$, true	Def. 4.1, page 35
$\neg\phi$	formula of $\mathcal{L}$, negation of ϕ	Def. 4.1, page 35
$\phi \vee \psi$	formula of $\mathcal{L}$	Def. 4.1, page 35
$\langle op \rangle_i \phi$	formula of $\mathcal{L}$	Def. 4.1, page 35
$\phi \mathcal{U}_i \psi$	formula of $\mathcal{L}$	Def. 4.1, page 35
$\mathcal{K}_i \phi$	formulas of $\mathcal{L}$	Def. 4.1, page 35
$\mathcal{B}_i \phi$	formula of $\mathcal{L}$	Def. 4.1, page 35
$\langle \mathcal{O} \rangle_i \phi$	derived formula of $\mathcal{L}$	Def. 4.3, page 44
$\phi \wedge \psi$	derived formula of $\mathcal{L}$, and	page 39
$[a]_i \phi$	derived formula of $\mathcal{L}$, weak next	page 39
$\Diamond_i \phi$	derived formula of $\mathcal{L}$, eventually	page 39
$\Box_i \phi$	derived formula of $\mathcal{L}$, always	page 39
$\phi \mathcal{W}_i \psi$	derived formula of $\mathcal{L}$, weak until	page 39
$\lvert\phi\rvert$	size of a formula	Def. 6.2.19, page 159

Tableau formulae

$\mathfrak{L}$	set of all tableau formulae	Def. 5.3.1, page 54
$\mathfrak{L}_{\mathfrak{l}}$	set of labelled formulae	Def. 5.3.1, page 54
$\mathfrak{L}_{\mathfrak{s}}$	set of structured formulae	Def. 5.3.1, page 54
$\mathfrak{label}$	set of tableau labels	Def. 5.3.1, page 54
$\mathfrak{l}, (\mathfrak{F}, \mathfrak{c})$	tableau labels	Def. 5.3.1, page 54
ℓ	$\mathcal{L}$-embedding	Def. 5.3.2, page 55
$\Gamma_{\mathcal{T}}$	set of tableau formulae, $\Gamma_{\mathcal{T}} \subset \mathfrak{L}$	
χ	chain in a set of tableau formulae	Def. 5.3.4, page 57
$\lvert\phi\rvert^{\mathsf{b1}}$	norm for logical formulae occurring in tableau formulae	Def. 5.6.4, page 81
$\lVert\mathfrak{l} \vdash \phi\rVert^{\mathsf{b1}}_{\Gamma_{\mathcal{T}}}$	norm for labelled tableau formulae occurring in the context of a finite tableau set that has the regular tableau property	Def. 5.6.5, page 83
$\lVert\Gamma_{\mathcal{T}}\rVert^{\mathsf{b1}}_{\Gamma'_{\mathcal{T}}}$	size of a set of formulae in the context of $\Gamma_{\mathcal{T}}$	Def. 5.6.5, page 83
sub(ϕ)	subformulae of a formula ϕ	Def. 5.6.9, page 92
sub($\Gamma_{\mathcal{T}}$)	subformulae occurring in a finite tableau set $\Gamma_{\mathcal{T}}$	Def. 5.6.10, page 92
$\lVert\Gamma_{\mathcal{T}}\rVert^{\mathsf{b2}}$	norm for finite tableau sets that have the regular tableau property	Def. 5.6.11, page 92
$\lVert\Gamma_{\mathcal{T}}\rVert^{\mathsf{b3}}$	norm for finite tableau sets that have the regular tableau property	Def. 5.6.13, page 95
$\lvert(\mathfrak{F}, \mathfrak{c}) \vdash \phi\rvert$	size of a tableau formula	Def. 6.2.20, page 159

Tableau

$\mathcal{T}$	Tableau	Def. 5.5.1, page 69
π	path in a tableau	
Γ_π	set of tableau formulae that occur on a tableau path π	

Appendix B

A Tableau System for $\mathcal{L}$

B.1 The Tableau Rules

Axioms and local rules

Axioms for unsatisfiability

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \phi, (\mathfrak{F},\mathfrak{c}) \vdash \neg\phi}{} \quad (\textbf{TR 1}) \qquad\qquad \frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \bot}{} \quad (\textbf{TR 2})$$

Rules referring to logical implication

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \top}{\Gamma_{\mathcal{T}}} \quad (\textbf{TR 3})$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \phi \wedge \psi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \phi, (\mathfrak{F},\mathfrak{c}) \vdash \psi} \quad (\textbf{TR 4}) \qquad\qquad \frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \phi \vee \psi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \phi \quad | \quad \Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \psi} \quad (\textbf{TR 5})$$

Local knowledge rule (reflexivity)

Rule **TR 6** may be applied only if the principal formula is unmarked. The side formula $(\mathfrak{F}, \mathfrak{c}) \vdash \mathsf{K}_i\phi$ will be marked.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \mathsf{K}_i\,\phi \ \text{(unmarked)}}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \mathsf{K}_i\,\phi, (\mathfrak{F}, \mathfrak{c}) \vdash \phi} \quad (\textbf{TR 6})$$

Induction rules

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \phi\,\mathcal{U}_i\,\psi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \psi \quad | \quad \Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \neg\psi \wedge \phi \wedge \langle\mathcal{O}\rangle_i\,\phi\,\mathcal{U}_i\,\psi} \quad (\textbf{TR 7})$$

(where $\langle\mathcal{O}\rangle_i\phi$ abbreviates $\bigvee_{a\in\mathcal{O}_i}\langle a\rangle_i\phi$)

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \phi\mathcal{W}_i\,\psi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \psi \quad | \quad \Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \phi \wedge \neg\psi \wedge [\mathcal{O}]_i\,\phi\mathcal{W}_i\,\psi} \quad (\textbf{TR 8})$$

(where $[\mathcal{O}]_i\phi$ abbreviates $\bigwedge_{a\in\mathcal{O}_i}[a]_i\phi$)

Rules Concerning Epistemic Structure

Creation of new R_i-successors

Rule (**TR 9**) may be applied only for unmarked principal formulae. The side formula $(\mathfrak{F}, \mathfrak{c}) \vdash \neg\mathsf{K}_i\phi$ will be marked.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \neg\mathsf{K}_i\,\phi \quad \text{(unmarked)}}{\Gamma_{\mathcal{T}},\ (\mathfrak{F}, \mathfrak{c}) \vdash \neg\mathsf{K}_i\,\phi, (\mathfrak{F}, \mathfrak{c})\mathsf{R}_i(\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}', \mathfrak{c}'), (\mathfrak{F}', \mathfrak{c}') \vdash \neg\phi} \quad (\textbf{TR 9})$$

Transitive and symmetric closure of the R_i-relation

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}',\mathfrak{c}')}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}',\mathfrak{c}'),(\mathfrak{F}',\mathfrak{c}')R_i(\mathfrak{F},\mathfrak{c})}\quad (\textbf{TR 10})$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}',\mathfrak{c}'),(\mathfrak{F}',\mathfrak{c}')R_i(\mathfrak{F}'',\mathfrak{c}'')}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}',\mathfrak{c}'),(\mathfrak{F}',\mathfrak{c}')R_i(\mathfrak{F}'',\mathfrak{c}''),(\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}'',\mathfrak{c}'')}\quad (\textbf{TR 11})$$

Transfer of formulae between R_i-indistinguishable situations

The principal formula in rule (**TR 12**) may be marked or unmarked. In both cases, the side formula $(\mathfrak{F}',\mathfrak{c}') \vdash \chi$ is unmarked.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}',\mathfrak{c}'),(\mathfrak{F},\mathfrak{c}) \vdash \chi \qquad \text{with } \chi \in \{K_i\,\phi, P, \neg P | P \in \mathcal{P}_i, \phi \in \mathcal{L}\}}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}',\mathfrak{c}'),(\mathfrak{F},\mathfrak{c}) \vdash \chi,(\mathfrak{F}',\mathfrak{c}') \vdash \chi}\quad (\textbf{TR 12})$$

The principal formula in rule (**TR 13**) may be marked or unmarked. In both cases, the side formula $(\mathfrak{F}',\mathfrak{c}') \vdash \neg K_i\phi$ is marked.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}',\mathfrak{c}'),(\mathfrak{F},\mathfrak{c}) \vdash \neg K_i\,\phi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c})R_i(\mathfrak{F}',\mathfrak{c}'),(\mathfrak{F},\mathfrak{c}) \vdash \neg K_i\,\phi,(\mathfrak{F}',\mathfrak{c}') \vdash \neg K_i\,\phi}\quad (\textbf{TR 13})$$

Rules Concerning Temporal Structure

Creation of next-successors

Rule (**TR 14**) may only be applied if the principal formula $(\mathfrak{F},\mathfrak{c})$ is unmarked. The side formula $(\mathfrak{F},\mathfrak{c})$ will be marked. Rule (**TR 14**) creates one denominator for each $a_i \in \mathcal{O}$.

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \langle . \rangle_i\,\phi,(\mathfrak{F},\mathfrak{c}) \quad \text{(unmarked)}}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \langle . \rangle_i\,\phi,(\mathfrak{F},\mathfrak{c}) \xrightarrow{a_1} (\mathfrak{F},\mathfrak{c}a_1),(\mathfrak{F},\mathfrak{c}),(\mathfrak{F},\mathfrak{c}a_1) \quad | \dots | \quad \Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \vdash \langle . \rangle_i\,\phi,(\mathfrak{F},\mathfrak{c}) \xrightarrow{a_n} (\mathfrak{F},\mathfrak{c}a_n),(\mathfrak{F},\mathfrak{c}),(\mathfrak{F},\mathfrak{c}_n)}$$
(**TR 14**)

Saturation of temporal successors

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \langle a\rangle_i \phi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash \phi} \tag{TR 15}$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash [a]_i \phi}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash \phi} \tag{TR 16}$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \langle b\rangle_i \phi \text{ with } i \in ag(a), a \neq b}{(\mathfrak{F},\mathfrak{c}) \vdash \bot} \tag{TR 17}$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \langle b\rangle_i \phi \text{ with } i \notin ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash \langle b\rangle_i \phi} \tag{TR 18}$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash [b]_i \phi \text{ with } i \notin ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash [b]_i \phi} \tag{TR 19}$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \langle \mathcal{O}\rangle_i \phi \text{ with } i \in ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash \phi} \tag{TR 20}$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash [\mathcal{O}]_i \phi \ \text{ with } i \in ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash \phi} \quad (\textbf{TR 21})$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash \langle\mathcal{O}\rangle_i \phi \ \text{ with } i \notin ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash \langle\mathcal{O}\rangle_i \phi} \quad (\textbf{TR 22})$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) \vdash [\mathcal{O}]_i \phi \ \text{ with } i \notin ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}a) \vdash [\mathcal{O}]_i \phi} \quad (\textbf{TR 23})$$

Interaction between knowledge and action

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) R_i (\mathfrak{F},\mathfrak{c}a) \ \text{ for all } i \notin ag(a)} \quad (\textbf{TR 24})$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) R_i (\mathfrak{F}',\mathfrak{c}') \ \text{ for all } i \in ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F},\mathfrak{c}) R_i (\mathfrak{F}',\mathfrak{c}'), (\mathfrak{F}',\mathfrak{c}') \vdash \langle a\rangle_i \top \ \text{ for all } i \in ag(a)} \quad (\textbf{TR 25})$$

$$\frac{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F}',\mathfrak{c}') \xrightarrow{a} (\mathfrak{F}',\mathfrak{c}'a), (\mathfrak{F},\mathfrak{c}) R_i (\mathfrak{F}',\mathfrak{c}') \ \text{ for all } i \in ag(a)}{\Gamma_{\mathcal{T}},\ (\mathfrak{F},\mathfrak{c}) \xrightarrow{a} (\mathfrak{F},\mathfrak{c}a), (\mathfrak{F}',\mathfrak{c}') \xrightarrow{a} (\mathfrak{F}',\mathfrak{c}'a), (\mathfrak{F},\mathfrak{c}) R_i (\mathfrak{F}',\mathfrak{c}'), (\mathfrak{F},\mathfrak{c}a) R_i (\mathfrak{F}',\mathfrak{c}'a) \ \text{ for all } i \in ag(a)} \quad (\textbf{TR 26})$$

B.2 Blocks for Rule Applications

The set of tableau formula is diveded into three blocks:

Block 1: Rules (**TR 3**), (**TR 4**), (**TR 5**), (**TR 6**), (**TR 7**), (**TR 8**), (**TR 15**), (**TR 16**), (**TR 17**), (**TR 18**), (**TR 19**), (**TR 20**), (**TR 21**), (**TR 22**), (**TR 23**)

Block 2: Rules (**TR 10**), (**TR 11**), (**TR 12**), (**TR 13**), (**TR 24**),(**TR 25**) and (**TR 26**)

Block 3: Rules (**TR 9**), (**TR 14**)

Appendix C

Tableau Example

In the following we give examples of tableau proofs. The first two examples demonstrate how a finite set of tableau formulae may lead to infinite tableaux.

In the tableau proofs, we will always mark the activator of a rule through a colored background and the $\boxed{\text{side formulae}}$ of a rule through a red box around the formula. Marked formulae will have a lighter color as unmarked formulae.

Example 1

Suppose for the first example the following static part of a model: $Ag = \{i\}, \mathcal{O}_i = \{a\}, \mathcal{P}_i = \emptyset$

We construct a tableau proof for the set of tableau formulae $\Gamma_0^{\mathcal{T}} = \{l_o \vdash \top\, \mathcal{U}_i \bot, l_o\}$:

$$l_o \vdash \top\, \mathcal{U}_i \bot, l_o$$

(**TR 7**)

$$\boxed{l_o \vdash \bot}, l \qquad \text{(TR 2)} \qquad \vdots$$

$$\Big| \quad \boxed{l_o \vdash \top \wedge \top \wedge \langle \mathcal{O} \rangle_i (\top\, \mathcal{U}_i \bot)}, l_o$$

(**TR 4**),(**TR 4**)

$$\boxed{l_o \vdash \top, l_o \vdash \langle \mathcal{O} \rangle_i (\top\, \mathcal{U}_i \bot)}, l_o$$

(**TR 3**)

$$l_o \vdash \langle \mathcal{O} \rangle_\jmath (\top\, \mathcal{U}_\jmath \bot), l_o$$

(**TR 14**)

$$\boxed{l_o \vdash \langle \mathcal{O} \rangle_i (\top\, \mathcal{U}_i \bot), l_o \xrightarrow{a} la\; l_o, la}$$

(**TR 20**)

$$\boxed{la \vdash \top\, \mathcal{U}_i \bot\; l_o \xrightarrow{a} la}, l_o, la$$

(**TR 7**)

$$\vdots$$

Example 2:

Suppose for the second example the following static part of a model: $\mathsf{Ag} = \{\mathsf{i}, \mathsf{j}\}, \mathcal{O}_\mathsf{i} = \{\mathsf{b}\}, \mathcal{O}_\mathsf{j} = \{\mathsf{a}\}, \mathcal{P}_\mathsf{i} = \mathcal{P}_\mathsf{j} = \emptyset$

We construct a tableau proof for the set of tableau formulae $\{\mathsf{l_o} \vdash \mathsf{K_i}\langle \mathsf{a} \rangle_\mathsf{j} \top, \mathsf{l_o}\}$.

$$\mathfrak{l}_o \vdash K_i\langle a\rangle_j\top,\ \mathfrak{l}_o$$

$$\overline{}\ (\mathbf{TR\ 6})$$

$$\boxed{\mathfrak{l}_o \vdash K_i\langle a\rangle_j\top,\ \mathfrak{l}_o \vdash \langle a\rangle_j\top},\ \mathfrak{l}_o$$

$$\overline{}\ (\mathbf{TR\ 14})$$

$$\dfrac{\mathfrak{l}_o \vdash K_i\langle a\rangle_j\top,\ \boxed{\mathfrak{l}_o \xrightarrow{b} \mathfrak{l}\mathfrak{b},\ \mathfrak{l}_o \vdash \langle a\rangle_j\top,\ \mathfrak{l}_o,\ \mathfrak{l}_\mathfrak{b}}}{\vdots}\ (\mathbf{TR\ 18})$$

$$|\quad \mathfrak{l}_o \vdash K_i\langle a\rangle_j\top,\ \boxed{\mathfrak{l}_o \xrightarrow{a} \mathfrak{l}\mathfrak{a},\ \mathfrak{l}_o \vdash \langle a\rangle_j\top,\ \mathfrak{l}_o,\ \mathfrak{l}_\mathfrak{a}}$$

$$\overline{}\ (\mathbf{TR\ 15})$$

$$\mathfrak{l}_o \vdash K_i\langle a\rangle_j\top,\ \boxed{\mathfrak{l}_o \xrightarrow{a} \mathfrak{l}\mathfrak{a},\ \mathfrak{l}\mathfrak{a} \vdash \top},\ \mathfrak{l}_o,\ \mathfrak{l}_\mathfrak{a}$$

$$\overline{}\ (\mathbf{TR\ 4})$$

$$\mathfrak{l}_o \vdash K_i\langle a\rangle_j\top,\ \mathfrak{l}_o \xrightarrow{a} \mathfrak{l}\mathfrak{a},\ \mathfrak{l}_o,\ \mathfrak{l}_\mathfrak{a}$$

$$\overline{}\ (\mathbf{TR\ 24})$$

$$\mathfrak{l}_o \vdash K_i\langle a\rangle_j\top,\ \boxed{\mathfrak{l}_o \xrightarrow{a} \mathfrak{l}\mathfrak{a},\ \mathfrak{l}_o R_i \mathfrak{l}\mathfrak{a}},\ \mathfrak{l}_o,\ \mathfrak{l}_\mathfrak{a}$$

$$\overline{}\ (\mathbf{TR\ 12})$$

$$\boxed{\mathfrak{l}_o \vdash K_i\langle a\rangle_j\top},\ \mathfrak{l}_o \xrightarrow{a} \mathfrak{l}\mathfrak{a},\ \boxed{\mathfrak{l}_o R_i \mathfrak{l}\mathfrak{a},\ \mathfrak{l}\mathfrak{a} \vdash K_i\langle a\rangle_j\top},\ \mathfrak{l}_o,\ \mathfrak{l}_\mathfrak{a}$$

$$\overline{}\ (\mathbf{TR\ 6})$$

$$\vdots$$

Example 3:

The next example will be a sketch of a tableau proof for the set of formulae $\Gamma_0 = \{K_i[a]_i p_1, \neg K_i\langle a\rangle_i \neg p_2, [a]_i K_i p_3, \langle a\rangle_i K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3)\}$.

The tableau proof will show, that this set of formulae is not satisfiable. We construct a set of tableau formulae from this set of logical formulae according to the construction defined in 5.3.3:

$\iota_o \vdash K_i[a]_i p_1, \iota_o \vdash \neg K_i\langle a\rangle_i \neg p_2, \iota_o \vdash [a]_i K_i p_3, \iota_o \vdash \langle a\rangle_i K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota_o$ Then we apply the appropriate tableau rules. The unsatisfiability mainly comes from the requirement about interaction of knowledge and time in definition 3.3.2(4).

$$\iota \vdash K_i[a]_i p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota \vdash [a]_i K_i p_3, \iota \vdash \langle a\rangle_i K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota$$

(TR 6)

$$\iota \vdash K_i[a]_i p_1, \iota \vdash [a]_i p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota \vdash [a]_i K_i p_3, \iota \vdash \langle a\rangle_i K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota$$

(TR 9)

$$\iota \vdash K_i[a]_i p_1, \iota \vdash [a]_i p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' \vdash [a]_i p_2, \iota \vdash [a]_i K_i p_3, \iota \vdash \langle a\rangle_i K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota, \iota'$$

(TR 14)

$$\iota \vdash K_i[a]_i p_1, \iota \vdash [a]_i p_q, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota' \vdash [a]_i p_2, \iota \vdash [a]_i K_i p_3, \iota \vdash \langle a\rangle_i K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota \xrightarrow{a} \iota a, \iota, \iota', \iota a$$

(TR 15)

$$\iota \vdash K_i[a]_i p_1, \iota \vdash [a]_i p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' \vdash [a]_i p_2, \iota \vdash [a]_i K_i p_3, \iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota \xrightarrow{a} \iota a, \iota, \iota', \iota a$$

(TR 16)

$$\iota \vdash K_i[a]_i p_1, \iota a \vdash p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' \vdash [a]_i p_2, \iota a \vdash K_i p_3, \iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota \xrightarrow{a} \iota a, \iota', \iota a, \iota$$

(TR 6)

$$\iota \vdash K_i[a]_i p_1, \iota a \vdash p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' \vdash [a]_i p_2, \iota a \vdash K_i p_3, \iota a \vdash p_3, \iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota a \vdash \neg p_1 \vee \neg p_2 \vee \neg p_3, \iota \xrightarrow{a} \iota a, \iota', \iota a, \iota$$

(TR 25)

$$\iota \vdash K_i[a]_i p_1, \iota a \vdash p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' \vdash [a]_i p_2, \iota a \vdash K_i p_3, \iota a \vdash p_3, \iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3),$$
$$\iota a \vdash \neg p_1 \vee \neg p_2 \vee \neg p_3, \iota \xrightarrow{a} \iota a, \iota' \vdash \langle a\rangle_i \top, \iota a, \iota', \iota$$

(TR 10), (TR 12), (TR 13)

$$\iota \vdash K_i[a]_i p_1, \iota' \vdash K_i[a]_i p_1, \iota a \vdash p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota' \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' R_i \iota, \iota' \vdash [a]_i p_2, \iota a \vdash K_i p_3, \iota a \vdash p_3, \iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota a \vdash \neg p_1 \vee \neg p_2 \vee \neg p_3,$$
$$\iota \xrightarrow{a} \iota a, \iota' \vdash \langle a\rangle_i \top, \iota a, \iota', \iota$$

(TR 14)

$$\iota \vdash K_i[a]_i p_1, \iota' \vdash K_i[a]_i p_1, \iota a \vdash p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota' \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' R_i \iota, \iota' \vdash [a]_i p_2, \iota a \vdash K_i p_3, \iota a \vdash p_3,$$
$$\iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota a \vdash \neg p_1 \vee \neg p_2 \vee \neg p_3, \iota \xrightarrow{a} \iota a, \iota' \vdash \langle a\rangle_i \top, \iota' \xrightarrow{a} \iota' a, \iota a, \iota', \iota, \iota' a$$

(TR 6)

$$\iota \vdash K_i[a]_i p_1, \iota' \vdash K_i[a]_i p_1, \iota' \vdash [a]_i p_1, \iota a \vdash p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota' \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' R_i \iota, \iota' \vdash [a]_i p_2, \iota a \vdash K_i p_3, \iota a \vdash p_3,$$
$$\iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota a \vdash \neg p_1 \vee \neg p_2 \vee \neg p_3, \iota \xrightarrow{a} \iota a, \iota' \vdash \langle a\rangle_i \top, \iota' \xrightarrow{a} \iota' a, \iota a, \iota' a, \iota, \iota'$$

(TR 15), (TR 16)

$$\iota \vdash K_i[a]_i p_1, \iota' \vdash K_i[a]_i p_1, \iota' a \vdash p_1, \iota a \vdash p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota' \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' R_i \iota, \iota' a \vdash p_2, \iota a \vdash K_i p_3, \iota a \vdash p_3,$$
$$\iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota a \vdash \neg p_1 \vee \neg p_2 \vee \neg p_3, \iota \xrightarrow{a} \iota a, \iota' a \vdash \top, \iota' \xrightarrow{a} \iota' a, \iota a, \iota' a, \iota, \iota'$$

(TR 3)

$$\iota \vdash K_i[a]_i p_1, \iota' \vdash K_i[a]_i p_1, \iota' a \vdash p_1, \iota a \vdash p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota' \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' R_i \iota, \iota' a \vdash p_2, \iota a \vdash K_i p_3, \iota a \vdash p_3,$$
$$\iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota a \vdash \neg p_1 \vee \neg p_2 \vee \neg p_3, \iota \xrightarrow{a} \iota a, \iota' \xrightarrow{a} \iota' a, \iota a, \iota' a, \iota, \iota'$$

(TR 26)

$$\iota \vdash K_i[a]_i p_1, \iota' \vdash K_i[a]_i p_1, \iota' a \vdash p_1, \iota a \vdash p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota' \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' R_i \iota, \iota' a \vdash p_2, \iota a \vdash K_i p_3, \iota a \vdash p_3,$$
$$\iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota a \vdash \neg p_1 \vee \neg p_2 \vee \neg p_3, \iota \xrightarrow{a} \iota a, \iota' \xrightarrow{a} \iota' a, \iota a R_i \iota' a, \iota a, \iota' a, \iota, \iota'$$

(TR 10), (TR 12)

$$\iota \vdash K_i[a]_i p_1, \iota' \vdash K_i[a]_i p_1, \iota' a \vdash p_1, \iota a \vdash p_1, \iota \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota' \vdash \neg K_i\langle a\rangle_i\neg p_2, \iota R_i \iota', \iota' R_i \iota, \iota' a \vdash p_2, \iota a \vdash p_2, \iota a \vdash K_i p_3, \iota a \vdash p_3,$$
$$\iota a \vdash K_i(\neg p_1 \vee \neg p_2 \vee \neg p_3), \iota a \vdash \neg p_1 \vee \neg p_2 \vee \neg p_3, \iota \xrightarrow{a} \iota a, \iota' \xrightarrow{a} \iota' a, \iota a R_i \iota' a, \iota' a R_i \iota a, \iota a, \iota' a, \iota, \iota'$$

(TR 1)