Belief Change Operations under Confidentiality Requirements in Multiagent Systems

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Cornelia Tadros

Dortmund

2013

technische universität
dortmund

Tag der mündlichen Prüfung: 08.04.2014

Dekan: Prof. Dr.-Ing. Gernot A. Fink

1. Gutachter: Prof. Dr. Joachim Biskup
2. Gutachterin: Prof. Dr. Gabriele Kern-Isberner

Danksagung

Vor allen möchte ich meinem Gott und meinem Erlöser, Jesus Christus, danken, der alle meine Wege zu einem guten Ende führt und mir in allen Schwierigkeiten im Blick auf Ihn Kraft gibt hindurchzugehen. Denn ich darf mich erinnern, wie groß und unüberwindbar die Schwierigkeiten auch erscheinen, Er verlässt mich nicht und ist immer derselbe, wie Amy Carmichael in einem ihrer Gedichte schreibt[1]:

> My soul, thou hast proved thy God,
> And fearest thou now? Behold Him, thy Light and thy Cover,
> Thy Champion, Companion, Lover,
> Thy Stay [. . .]
> Thy Song 'mid a thousand distresses.
> Is this all new to thee?
> The Lord thy God – hath He stood aloof?
> (Verily thou hast put Him to proof)

Des Weiteren haben auch viele Menschen zum Abschluss meiner Dissertation beigetragen, denen ich an dieser Stelle danken möchte.
An erster Stelle möchte ich Joachim Biskup danken, dessen sorgfältige Korrekturen meiner Entwürfe für mich von großen Wert sind und der sich gerne viel Zeit für ausführliche Diskussion meiner Ideen genommen hat. Seine Ratschläge waren mir sehr hilfreich. Auch hat er mich in allen Dingen unterstützt, die zum Erfolg meiner Arbeit beigetragen haben. Außerdem möchte ich noch Gabriele Kern-Isberner, meiner zweiten Gutachterin, danken, die mir wichtige zusätzliche Einblicke in ihren Forschungsschwerpunkt der Wissensrepräsentation gegeben hat, so dass ich meine Dissertation noch besser aus dem Blickwinkel dieser für Teile meiner Dissertation grundlegenden Forschung betrachten konnte. Ihre Korrekturen und Diskussion meiner Entwürfe für die Dissertation waren mir auch eine wichtige Hilfe.
Danken möchte ich auch Thomas Schwentick und Thomas Ruhroth für ihre Bereitschaft sich Zeit für meine Dissertation zu nehmen und an meiner Prüfungskommission teilzunehmen.
Ohne den Sonderforschungsbereich 876 wäre meine Dissertation in dieser Form

[1] Aus *Mountain Breezes, the collected poems of Amy Carmichael, CLC publications, 2001.*

nicht möglich gewesen und so gilt mein Dank natürlich allen, die zum Entstehen dieses Sonderforschungsbereichs beigetragen haben insbesondere der DFG. Dazu konnte ich im Sonderforschungsbereich (SFB) auch viele andere Forschungsrichtungen und Kollegen anderer Fächer kennenlernen und so viele interessante Gespräche und Diskussionen über mein Thema hinaus führen. Besonders die Diskussion mit Patrick Krümpelmann, meinem Kollegen im Teilprojekt A5 des SFB, haben meinen Wissenshorizont über intelligente Agenten erweitert. Die gemeinsamen durchdachten Fragenstellungen, wie Forschung bezüglich intelligenter Agenten (und anderer Themen der künstlichen Intelligenz) und Computersicherheit einander bereichern können, sind immer noch ein spannendes Thema für mich. In dieser Zusammenarbeit haben wir auch die Gelegenheit gehabt, die theoretischen Ansätze in der Multiagenten Plattform Angerona zu erproben. An dieser Stelle haben besonders Tim Janus, Sebastian Homann und Pia Wierzoch durch ihre Mitarbeit und ihren Einsatz für die Angerona Implementierung sehr geholfen. Auch Dirk Schalge möchte ich für seine Implementierung meiner Arbeit in dem Prototyp der kontrollierten Interaktionsauswertung danken.
Meine aktuellen und ehemaligen Kollegen und Kolleginnen am Lehrstuhl haben mich den Dissertationsstress und anderen Stress in den Pausen vergessen lassen und viel zu einer angenehmen Arbeitsatmosphäre beigetragen. Besonders möchte ich noch meiner ehemaligen Kollegin Lena Wiese danken, die mir am Anfang der Zeit am Lehrstuhl eine Ermutigung und Unterstützung war. Zuletzt gilt mein Dank noch meinem Mann Ihab – ohne den diese Dissertation wahrscheinlich noch nicht fertig gestellt wäre.

Abstract

Multiagent systems are populated with autonomous computing entities called agents which pro-actively pursue their goals. The design of such systems is an active field within artificial intelligence research with one objective being flexible and adaptive agents in dynamic and inaccessible environments. An agent's decision-making and finally its success in achieving its goals crucially depends on the agent's information about its environment and the sharing of information with other agents in the multiagent system. For this and other reasons, an agent's information is a valuable asset and thus the agent is often interested in the confidentiality of parts of this information. From research in computer security it is well-known that confidentiality is not only achieved by the agent's control of access to its data, but by its control of the flow of information when processing the data during the interaction with other agents. This thesis investigates how to specify and enforce the confidentiality interests of an agent $\mathcal{D}$ while it reacts to iterated query, revision and update requests from another agent $\mathcal{A}$ for the purpose of information sharing.

First, we will enable the agent $\mathcal{D}$ to specify in a dedicated confidentiality policy that parts of its previous or current belief about its environment should be hidden from the other requesting agent $\mathcal{A}$. To formalize the requirement of hiding belief, we will in particular postulate agent $\mathcal{A}$'s capabilities for reasoning about $\mathcal{D}$'s belief and about $\mathcal{D}$'s processing of information to form its belief. Then, we will relate the requirements imposed by a confidentiality policy to others in the research of information flow control and inference control in computer security.

Second, we will enable the agent $\mathcal{D}$ to enforce its confidentiality aims as expressed by its policy by refusing requests from $\mathcal{A}$ at a potential violation of its policy. A crucial part of the enforcement is $\mathcal{D}$'s simulation of $\mathcal{A}$'s postulated reasoning about $\mathcal{D}$'s belief and the change of this belief. In this thesis, we consider two particular operators of belief change: an update operator for a simple logic-oriented database model and a revision operator for $\mathcal{D}$'s assertions about its environment that yield the agent's belief after its nonmonotonic reasoning. To prove the effectiveness of $\mathcal{D}$'s means of enforcement, we study necessary properties of $\mathcal{D}$'s simulation of $\mathcal{A}$ and then based on these properties show that $\mathcal{D}$'s enforcement is effective according to the formal requirements of its policy.

Contents

Chapter 1

Introduction

1.1 Confidentiality in Multiagent Systems

In the design and maintenance of IT systems today there are various challenges like distribution of computing entities, rapid change of processes, more and more complexity of the overall system and large amounts of data. As a solution to these challenges the *agent paradigm* emerged in software engineering. In simple terms, an *intelligent agent* is an autonomous computing entity that interacts with its environment, in particular with other agents, and is directed by internal *goals* [95, 94]. The agent's *autonomy* mainly distinguishes it from other computing entities such as objects in the object-oriented paradigm. An agent's ability to pro-actively pursue its goals and, at the same time, to react to changes in its environment with autonomous decision-making makes it appropriate to face the above mentioned challenges. Moreover, in a multiagent system, agents may *collaborate* to achieve a joint goal, for example, by negotiation. In conclusion, the agent paradigm helps to reduce the complexity in the design of a computing entity in an *inaccessible, dynamic environment* [94] where that entity neither is able to obtain complete, accurate and up-to-date information about the environment nor has total control over the changes in the environment. Moreover, the agent paradigm contributes to increasing the flexibility and scalability of the design. Research has widely investigated the application of this paradigm in many sectors such as the business sector and the health care sector, cf. [68, 62, 39, 5, 97] as a small selection. Examples of application areas are flexible business process management, supply chain management in particular and electronic commerce.

In many applications, *information* is a most valuable asset. It might be a business factor with costly acquisition or personal information under the protection of law. Consequently, the dissemination of information usually is restricted by the interests of its owner. For example, a company would not want to leak details about its product proposals under development to its competitors. A patient in a hospital would neither want its health report be read by others than the responsible hospital staff. These interests are commonly summarized by the security interest of *confidentiality of information*. Motivated by such interests, there is a long line of research in computer security that deals with the control of the flow of information or the control of access to data. The focus of this thesis is to apply and to extend several results of this research to design confidentiality preserving multiagent systems. Mainly, those results come from the areas of *database security* and *information flow control*.

In database security it is well-known that preventing only direct access to sensitive data items might not suffice to ensure confidentiality of the information represented by these data. Rather, for example, after accessing non-protected data items by query requests, a database user might be able to deduce sensitive information from the retrieved data. Moreover, such a deduction might exploit the user's knowledge of database constraints and further *a priori knowledge*. The challenge to prevent such deductions is known as the *inference problem* [49, 17]. *Controlled Interaction Execution* [16, 18] (CIE) presents solutions to the inference problem for query-answering and updates in logic-oriented database systems.
Another security interest in database security, conflicting with confidentiality, is the *availability of information* [35]. A database user should be provided with the information needed within his duties.

Database systems and multiagent systems by design show differences that make the application of database security to confidentiality in multiagent systems more challenging. Database security research often assumes data models like the relational data model whose theory bases on classical logic such as first order logic. Moreover, interaction with database systems follows the *client-server paradigm*.

In contrast, to achieve its goals, an intelligent agent has to make decisions on the basis of incomplete information about its environment. To support its decision making, usually, the agent processes this information with some non-classical logic such as nonmonotonic logic which results in the agent's *belief* about its environment. Further, an agent's autonomy contrasts with the client-server paradigm of database systems: A database offers a predefined range of services under preconditions like sufficient authorization of a database user, without any autonomy on the part of the database.

In [24], there is a first proposal to benefit from the results in database security research in the specification and construction of a BDI-agent enforcing its confidentiality aims in a multiagent system. The BDI-agent model describes an agent's process of decision making with its beliefs, desires and intentions [95, 94]. In this agent model, the agent's belief and its ability to share information with other agents is a core functionality of an intelligent agent.

In this thesis, we will focus on this core functionality. In particular, we will focus on *epistemic agents*. Epistemic agents have belief about their environment, themselves and other agents and are capable to change their belief upon their perceptions. The agent's *belief* are propositions that the agent accepts to hold in its environment or about some agent. Apart from epistemic capabilities, an intelligent agent is supposed to have further capabilities such as reactivity, pro-activity and social ability [94]. These capabilities are not studied further in this thesis.

With our focus on epistemic agents, we mainly consider the following aspects of a multiagent system under confidentiality requirements.
First, an agent's epistemic capabilities base on the representation of the agent's belief and its processing of this belief by *belief change operators* upon receiving information from another agent. In this thesis, we consider the belief change operators of *update* and *revision*.
Second, in the interaction with other agents an agent is principally motivated to share parts of its belief with the others (for example, to achieve joint goals), but at the same time the agent usually aims to hide other sensitive parts of its belief from other agents as an interest in the confidentiality of these belief.
Third, we investigate a scenario of two interacting epistemic agents only whose

interaction nevertheless may be viewed as a private conversation in a group of more than two agents. More specifically, we consider that one agent reacts to the requests of another agent while at the same time the reacting agent aims to enforce its confidentiality interests towards the requester.

Our study of confidentiality in multiagent systems under these three aspects partly starts by making use of results in database security research. However, whereas a major contribution of this thesis is confidentiality under belief change, in database security research, there is little work on enforcing confidentiality towards a database user who may update a database beside issuing queries to that database. Beyond database security research, we will consider the enforcement of the confidentiality interests of an agent that forms its belief by nonmonotonic reasoning [38].

Moreover, our study of confidentiality in multiagent systems under these three aspects bases on further research in computer security. In computer security research it is well-known that apart from explicitly communicated information a system may leak information if the system executes some publicly known algorithm and discloses parts of its output. This issue is dealt with in research within the field of *information flow control*, cf. [15, 79, 4] as examples. It must be accounted for in the formal specification of an agent's confidentiality interests. Information flow control in multiagent systems is made difficult by the complexity of the data structures for the representation of an agent's belief and the operators processing these data structures (in this thesis, the belief operators of update and revision). The specification and control of the information flow within an intelligent agent with its full functionality, for example, including decision-making with its desires, intentions and beliefs and planning its actions, is even more involved. But the consideration of the full functionality under confidentiality requirements is left for future work beyond this thesis. Rather, in this thesis we lay the formal foundation for specifying an (abstract) agent's confidentiality aims towards other agents, basing and relating to research on information flow control. Then, we demonstrate how to enforce these aims in two exemplary implementations of an epistemic agent.

We will give a detailed overview of the contributions of this thesis in the following two sections.

1.2 Research Questions

We illustrate the research questions addressed in this thesis with a running example of negotiating agents in a supply chain, cf. [97]. The scenario is motivated by the literature which suggests the use of agents for flexible business processes in a complex and changing business environment [68], including proposals in industrial projects [39]. In this context, informational resources are seen as a business factor whose acquisition might be costly and whose disclosure might cause severe loss in profit [5].
We roughly outline the scenario in the following. A manufacturer has a portfolio with different kinds of products and runs its own product development department. Each project undergoes a feasibility study which assesses whether the proposed (further) development of a product is realizable. The manufacturer may have different confidentiality interests:

- While the manufacturer's product portfolio is well known to everybody, the manufacturer does not want its competitors to know which products or even which kind of products in the portfolio have been proposed for (further) development.

- If proposals had already been leaked to competitors, the manufacturer still would not want to disclose a positive result of the respective feasibility study.

- In general, the manufacturer does not even trust possible part suppliers not to leak this sensitive information to the manufacturer's competitors. On the contrary, the manufacturer suspects those suppliers to self-interestedly reveal this information in order to increase the demand in their parts on the market.

- Moreover, the manufacturer does not want any supplier to know whether he has a contract for deployment of parts with another supplier.

Conflicting with the confidentiality interests, the manufacturer needs to share information with possible part suppliers. First, the manufacturer will make a request for proposal including the requirement specification of the parts. In turn, suppliers will give their offers with the technical specification of the parts.

Afterwards, there may be further negotiation between the two parties refining the parts' specification and maybe finally ending in a contract between the manufacturer and a part supplier.
In this scenario, consider a dialog between two intelligent agents, agent $\mathcal{D}$ on behalf of the manufacturer and agent $\mathcal{A}$ on behalf of part supplier #1. The dialog is a part of the negotiation of supply contracts between the manufacturer and part supplier #1. Moreover, during this dialog, agent $\mathcal{D}$ is responsible for defending the manufacturer's confidentiality interests towards agent $\mathcal{A}$ as a potential attacker of these interests. Beforehand, another supplier #2 and the manufacturer agreed on the deployment of part #1 from supplier #2.

1. Agent $\mathcal{A}$ inquires of $\mathcal{D}$ whether the manufacture's product proposal is realizable if supplier #1 can supply part #2.
2. Agent $\mathcal{D}$ replies with yes.
3. Agent $\mathcal{A}$ informs agent $\mathcal{D}$ that supplier #1 can supply part #2.
4. Agent $\mathcal{D}$ notifies $\mathcal{A}$ that it takes into account the received information.
5. Agent $\mathcal{A}$ informs agent $\mathcal{D}$ that part #1 from supplier #2 and part #2 from supplier #1 are incompatible.
6. Agent $\mathcal{D}$ notifies $\mathcal{A}$ that it takes into account the received information.
7. Agent $\mathcal{A}$ inquires of $\mathcal{D}$ whether the product proposal is realizable.
8. Agent $\mathcal{D}$ replies with no.
9. Agent $\mathcal{A}$ inquires of $\mathcal{D}$ whether the manufacturer's product proposal would be realizable if the incompatibility between part #1 from supplier #2 and part #2 from supplier #1 was resolved.
10. Agent $\mathcal{D}$ replies with yes.
11. Agent $\mathcal{A}$ informs agent $\mathcal{D}$ that the incompatibility has been resolved.
12. Agent $\mathcal{D}$ notifies $\mathcal{A}$ that it takes into account the received information.

Intuitively, although agent $\mathcal{D}$ neither explicitly talked about the positive realizability of the proposal nor about the previous agreement with another supplier, it obviously failed in the responsibility for the manufacturer's confidentiality interests. For on an intuitive level we understand that agent $\mathcal{D}$'s replies have revealed several sensitive pieces of information to agent $\mathcal{A}$. For example, agent $\mathcal{A}$ might combine $\mathcal{D}$'s positive answer to the query in step 1 with $\mathcal{D}$'s confirmation in step 4 and infer that $\mathcal{D}$ considers the manufacturer's product proposal realizable in step 4. Basing on this inference, agent $\mathcal{A}$ might even infer together with $\mathcal{D}$'s answer in step 8 that the manufacturer deploys part #1 from supplier #2 in the proposed product. This small scenario should make clear the motivation and challenges behind the following research questions that we will treat in this thesis.

1. How could agent $\mathcal{D}$ effectively control its reactions to *query requests* and *revision requests* from agent $\mathcal{A}$ (that is, being informed by $\mathcal{A}$ about the current situation)? (Steps 1 to 8 in the dialog)

2. How could agent $\mathcal{D}$ effectively control its reactions to *query requests* and *update requests* from agent $\mathcal{A}$ (that is, being informed by $\mathcal{A}$ about a change in the current situation)? (Steps 9 to 12 in the dialog)

3. How could agent $\mathcal{D}$ model and compute agent $\mathcal{A}$'s inferences about agent $\mathcal{D}$'s processing of the information received from $\mathcal{A}$? (Here, the processing is either by revision or update operators)

4. How should agent $\mathcal{D}$ *formally declare its confidentiality aims* towards a rational attacker $\mathcal{A}$ so that the effectiveness of its control mechanisms could be formally proven?

5. How far do $\mathcal{D}$'s control mechanisms agree with agent $\mathcal{D}$'s further goal to *cooperatively share information* with agent $\mathcal{A}$?

Although we will first treat research questions 1 and 2 in separation, we will further discuss the design of a confidentiality-preserving epistemic agent that processes both update and revision requests and outline the challenges in this design.

1.3 Outline of Thesis

This thesis is divided into two parts: Part I covers the formal specification of confidentiality while Part II covers the implementation of an agent effectively enforcing confidentiality according to this specification and the verification of the effectiveness of this enforcement.

In Part I, the main objective is to provide an epistemic agent with means to declare its confidentiality interests in a confidentiality policy with a formal semantics. Hence, Part I presents a solution to research question 4 raised in the previous section. Actually, research on information flow control and database security is full of formalisms for the specification of confidentiality interests. In this thesis, we have chosen the framework of Runs & Systems [48] for the specification of confidentiality basing on the work [58] by Halpern and O'Neill with various definitions of secrecy. Our aim is to adapt their definitions of secrecy according to the requirements of an epistemic agent with confidentiality interests. At the same time, our aim is to relate the adapted definitions to alternative definitions in the literature. We proceed as follows.
In Chapter 2, we survey related work on the specification of confidentiality requirements of a computing system that interacts with a potential attacker against these requirements. Our emphasis is on *possibilistic* approaches which define confidentiality as an *indistinguishability property*, or in other terms, as a *confinement of the attacker's knowledge* about the system. Especially, we detail the framework [58] by Halpern and O'Neill which we use for our own specification in this thesis.
Chapter 3 generalizes Halpern and O'Neill's secrecy definitions in Runs & Systems to *possibility policies* and their semantics of *policy-based secrecy*. Policy-based secrecy is shown to be more expressive than all of Halpern and O'Neill's possibilistic secrecy definitions. The richer expressiveness provides an epistemic agent with suitable means to declare its confidentiality interests as we investigate in Chapter 4. Moreover, we discuss how policy-based secrecy may be related to other confidentiality requirements in the literature.
In Chapter 4, following up the informal dialog of two agents in Section 1.2, we model a scenario of two epistemic agents that share information from the perspective of security engineering in the Runs & Systems framework: The

functionality of the agent with confidentiality interests and the capabilities of the other agent attacking these interests are postulated and represented in this model. In terms of this model, we define the semantics of an agent's confidentiality policy. To put this semantics into the context of other definitions on information flow confinement, we show that, when translating confidentiality policies to a single possibility policy, confidentiality can be expressed by policy-based secrecy.
In Chapter 5, we summarize our contributions to the formal specification of an epistemic agent's confidentiality aims and discuss open problems and directions for future work.

Part II details the construction of a defending epistemic agent enforcing its confidentiality aims declared within the framework of Part I. We give two examples of the construction each with a different implementation of the agent's belief component which represents the agent's information about its environment. The first implementation is a simple logic-oriented database model so that the construction of the agent may follow and benefit from results in database security. The second implementation bases the agent's belief on nonmonotonic reasoning and is a standard in artificial intelligence for intelligent agents in inaccessible environments.
In the construction, we follow the line of work of Controlled Interaction Execution (CIE) [16, 18]: The defending agent controls its interaction with the other attacking agent by means of a *control function* which *simulates* the attacker's reasoning about the defender's belief. The attacker's reasoning has been postulated by the model of Part I while now Part II focuses on the simulation of the postulated reasoning. The essential differences between postulated and simulated reasoning are as follows. On the one hand, the postulated reasoning is about the multiagent system specified by abstract states and executions on these states and is used by the security engineer for the specification of confidentiality requirements and their verification. On the other hand, the simulated reasoning is usually formalized in a known logic or one of its fragments and is used by the defending agent for computing the attacker's inferences at runtime. Overall, Part II covers all research questions from the previous section but question 4. More specifically, in Part II we proceed as follows.

In Chapter 6, we detail the implementation of the defending agent's belief component and the agents' interaction in the two examples, yet without the control functions for enforcing confidentiality. Further, we set the two implementations into the context of related work from the database and artificial intelligence research communities, with an emphasis on work that suggests how to combine or enhance the two implementations. Based on the related work, we outline an implementation of the agent that combines and extends the functionality of the two proposed implementations for future work.
Chapter 7 bases on the first exemplary implementation: The defending agent employs a *complete propositional database instance* with integrity constraints and reacts to *view update transaction requests* and queries from the attacking agent. In particular, Chapter 7 presents a solution to research questions 2 and 3 from the previous section. The chapter focuses on four aspects:

- first, a solution to the conflict between integrity and confidentiality discussed in database security research;
- second, the enforcement of the confidentiality of the fact that the agent has previously believed a formula (continuous confidentiality of Chapter 4);
- third, the correctness of the simulation of the attacker's reasoning postulated in Chapter 4;
- fourth, the cooperativeness of the defending agent in sharing information with the other agent under confidentiality requirements.

With the latter aspect, we address research question 5 raised in the previous section. The chapter completes the construction of the defending agent with control functions for iterated query and view update transaction requests and a proof of their effectiveness in enforcing confidentiality.
Chapter 8 bases on the second exemplary implementation: The defending agent reasons from incomplete information (called *current assertions*) about its environment with a *nonmonotonic consequence relation* and, this way, forms its belief about the environment. The defending agent may acquire additional information through belief revision requests from the attacker. In artificial intelligence research, nonmonotonic consequence relations model rational

reasoning from incomplete information. In contrast to classical consequence relations such as propositional entailment, nonmonotonic consequence relations might withdraw a conclusion in the presence of additional assertions.
Altogether, Chapter 8 presents a solution to research questions 1 and 3 from the previous section where research question 3 is the more intricate one. Due to the complexity of the second implementation, we will not investigate research question 5 in depth, but only discuss possible approaches to this question.
Addressing research question 3, the chapter elaborates on the simulation of the attacker's postulated reasoning about the defender's current belief. The main challenge for the simulation and its computation is the situation that the defending agent might have hidden parts of its current assertions and its nonmonotonic consequence relation from the attacking agent in order to protect its confidential belief. For the computation of the simulation, we define "allowance properties" of an axiomatization of a class of nonmonotonic consequence relations. If the defending agent's nonmonotonic reasoning is an instance of such a class, then the attacker's reasoning about the defender's current belief may be computed by deduction with an "allowed axiomatization" of this class.
The chapter completes the construction of the defending agent with control functions for iterated query and revision requests and a proof of their effectiveness in enforcing confidentiality.
Finally, in the conclusion of this thesis we highlight the relevance of our contributions to the perspective of confidentiality preserving intelligent agents in multiagent systems. For the reader's convenience, we included the complete lists of all definitions, figures, assumptions, propositions etc. and an index of all relevant terms and symbols into the appendix of this thesis. Moreover, we moved the complete proofs to the appendix. But we added sketches of proofs at places where it contributes to a better understanding of our concepts.

1.4 Contributions of Previously Published Work

All publications which contributed to this thesis are joint work with my advisor Joachim Biskup. Being my advisor, he proofread my drafts, discussed them with me and suggested to investigate further research questions during our discussions. His particular contributions are made clear in the following overview of the publications.

- In [28], we relate the confidentiality requirements in the work of Controlled Interaction Execution (CIE) [16, 18] to secrecy in multiagent systems in [58]. The publication for the first time elaborates and shows the commonalities between research in information flow control and the framework of CIE. Whereas the publication studies an agent system with a database user and a database with a query interface, in this thesis we extend this work and consider general systems of two interacting epistemic agents with any type of request. The contributions of the publication are partly included in Chapter 3 and Chapter 4.

- In [29], we study controlled processing of query requests and view update transactions to propositional databases for preserving confidentiality of information in the database. Our study especially proposes a solution to the conflict between confidentiality and maintenance of the integrity constraints of the database by refusing transactions for the sake of confidentiality. Previously, in database security research this conflict was supposed not to be solvable this way.

  Since the complexity of information flow control for transactions is considerably higher than for query processing, the question naturally emerged whether the procedures restrict the availability of information unnecessarily. Thus, we complemented our proposal of control procedures with an analysis of the availability of information provided by these procedures and of a possible improvement of the procedures under that aspect.

  The contributions of this publication are found in Chapter 7 while we added detailed proofs and adapted the definition of the system to the system model of Chapter 4. Moreover, we rewrote the parts on the simulation of the attacker $\mathcal{A}$'s reasoning in order to thoroughly relate that simulation to the postulated reasoning of Chapter 4. This way, we intended to make clear the translation from postulated to simulated reasoning and the essential requirements on the simulation.

The definition of local optimality for the availability analysis was inspired by an earlier talk of my advisor Joachim Biskup on that issue. The talk suggested properties of control functions for query processing and a notion of optimality under the availability aspect for functions with these properties.

- In [30], we investigate the situation of an agent $\mathcal{D}$ with nonmonotonic reasoning that defends its confidentiality interests towards another agent $\mathcal{A}$ while reacting to $\mathcal{A}$'s query and belief revision requests. We present control procedures for the defender $\mathcal{D}$ to effectively enforce confidentiality and model the attacker $\mathcal{A}$'s skeptical entailment about $\mathcal{D}$'s belief that is based on an ordinal conditional function [89] for nonmonotonic reasoning.

  The work is preliminary to the following publication in [31] which significantly contributed to this thesis. The results of [30] are entirely covered by those in [31]. But as a contribution of its own, their proofs as sketched in [30] argue model-theoretically in terms of ordinal conditional functions. Those proofs are not included in this thesis.

- The publication [31] extends the preliminary publication in [30] under several respects. First of all, it includes the details of all proofs. Second, while the work in [30] solely is based on the implementation of the defending agent $\mathcal{D}$'s reasoning by an ordinal conditional function [89], we in the work of [31] offer the flexibility to take the fixed reasoning from some class of reasoning that can be characterized by an axiomatization satisfying dedicated "allowance properties". Third, while the preliminary publication does not detail an approach for computing the presented skeptical entailment of the attacker $\mathcal{A}$, we in [31] also indicate the algorithms needed by the defender $\mathcal{D}$.

  The contributions of the publication are mainly found in Chapter 8 while the security engineer's model of the agent system in [31] is brought forward in Chapter 4.

My advisor Joachim Biskup suggested to elaborate and to motivate the model of the agent system from the security engineering perspective and thus substantially contributed to the security engineer's model as it is found in this thesis. Moreover, he put forward the idea to use proof theory and axiomatizations instead of generic models such as plausibility structures which led me to consider classes of consequences relations characterized by axiomatizations with "allowance properties" for the defender's reasoning.

In this thesis, we further made some effort to present the results of all publications in a unified agent model and to highlight their connections under our leading research question of how to design a confidentiality preserving epistemic agent.

Part I

Specifying Confidentiality in Multiagent Systems

The central theme of this thesis will be the task of the security engineer to construct an interacting epistemic agent enforcing its confidentiality interests. As anticipated, in this thesis we focus on information sharing between two epistemic agents, cf. the example scenario in Section 1.2. An agent $\mathcal{D}$ reacts to the requests of the other agent $\mathcal{A}$ while it has confidentiality interests towards the requester. Thus, from the security engineering perspective, the requesting agent $\mathcal{A}$ is seen as an *attacker* of the reacting agent $\mathcal{D}$'s confidentiality interests. Conversely, the reacting agent is seen as a *defender* of these interests. In the following, we will outline the main activities of the security engineer for constructing the reacting agent according to [15].

Above all, agent $\mathcal{D}$ must be provided with means to declare its confidentiality interests. This declaration is understood (in terms of a formal semantics) as requiring a well-defined *confidentiality property*. In this context, Biskup in [15, Chapter 1.2] highlights that specifying *security requirements* involves identifying the *informational activities* of the party with security interests as well as the *suspected threats* to these interests. Further, all *assumptions* underlying an analysis of the achievements with respect to these requirements should attentively be laid open.

Adhering to these guidelines, the following three activities make up the first part of the security engineering task, the *formal specification*.

1. The functionality of the agent $\mathcal{D}$ to be constructed by the security engineer must be specified as a basis of security engineering (referring to its "informational activities" and "assumptions").

2. The computational means and informational resources of the potentially attacking agent $\mathcal{A}$ must be postulated and modeled (referring to "suspected threats" and "assumptions").

3. The means to declare confidentiality interests in a dedicated policy language and its semantics must be specified (referring to "security requirements").

Further, in Part II of this thesis, we will present the second part of the security engineering task, the *implementation and verification of the specification*.

Chapter 2

Information Flow Control in the Literature

As a preliminary to the formal specification of an agent's confidentiality interests, in this chapter, we survey several approaches for specifying security requirements by *information flow control*. These requirements have in common that they confine the attacker's *information gain*. In this thesis, confidentiality properties will be specified in terms of what the attacker *should not know* about the defender basing on the framework by Halpern and O'Neill surveyed in Section 2.3. In the literature, properties based on the attacker's knowledge are also called *possibilistic* since the attacker knows some fact if it does not consider possible that this fact is not true. The survey in this chapter should provide the basis to relate the specification of confidentiality properties in this thesis to the literature. We will do so in Chapter 3 and Chapter 4.

The chapter is organized as follows. Section 2.1 introduces the basic concepts for the specification of information flow confinement which are common to a variety of approaches in the literature. Section 2.2 to 2.3 summarize different frameworks with a possibilistic model of the attacker's information gain. The framework of Section 2.3 on which we base the specification of confidentiality properties in this thesis will be given a more detailed overview. The last two sections should make clear the scope of this thesis by pointing to other aspects in the specification of confidentiality properties to be considered in future work. Namely, Section 2.4 introduces *probabilistic models* of the attacker's information gain. Section 2.5 surveys literature which specifies confidentiality properties syntactically in the language of some logic.

2.1 Modeling the Attacker's Information Gain

The model of the attacker's information gain is part of the security engineering task of formal specification. Within such a model the security engineer postulates the means of the attacker to acquire information about the defender, including what the attacker a priori knows about the defender and the system in general. In the following, we often refer to the defending agent $\mathcal{D}$ as the *responder* and to the attacking agent $\mathcal{A}$ as the *observer*. This way, we emphasize that $\mathcal{D}$ only reacts to $\mathcal{A}$'s requests in the focused scenario of Section 1.2, while in turn $\mathcal{A}$ observes $\mathcal{D}$'s reactions.

The responder's confidentiality interests usually are that the observer is not able to gain particular pieces of information from its observations. This requirement may reflect the need to protect business information, personal information or organizational information. Complementing and often conflicting confidentiality interests, an agent might have other security interests such as *anonymity* (hiding the identity of an agent involved in some event) and *integrity* "as unmodified content" [15] (preventing the distortion of information), cf. [15] for an overview. Like confidentiality, the mentioned interests can be expressed as requirements of information flow confinement, cf. [57, 42] as examples.

As part of the formal specification of security requirements, here confidentiality, the observer's ability as a potential attacker to gain particular pieces of information from its observations must be described in a mathematical model. The literature proposes various such models of the observer's information gain and corresponding formal frameworks to specify confidentiality requirements or other security requirements. According to the survey by Biskup in [15, Chapter 4] these proposals have commonalities which we will outline in the following.

First, Biskup points out that an observation is a syntactic object, for example a string sent as a message, whereas confidentiality is concerned with semantic objects represented by such syntactic objects, for example, events in the system or values of program variables. Therefore, second, the observer interprets the syntactic object as some semantic object. Then, third, the observer aims to gain information from the interpreted object about other semantic objects accessible to the responder, exploiting *a priori knowledge* about the responder. The a priori knowledge is expressed in a mathematical model where the mathematical model includes the representation of semantic objects. Biskup in [15] surveys three classes of simple models: algebra-oriented models, logic-oriented models and probability-oriented models. Here, we select examples of modeling the observer's information gain and the respective formal framework for specifying confidentiality requirements. As mentioned in the introduction of this chapter, this selection includes possibilistic models of information gain as alternatives to that of Section 2.3 which we will use. Further, the selection includes models to be considered in future work.

2.2 Traces

In this section, the semantic objects the observer attempts to gain information about are events occurring in the multiagent system. The activities of the agents in the system generate a sequence of events called a *trace* of the system. In the following, we summarize an overview of this model from [15, Section 5.7]. Formally, a system is comprised of a set of *events* E including input events I and output events O and a set of traces $Sys \subseteq E^*$. The set Sys must be *closed under prefixes*, that is, all prefixes of each trace in Sys are included in Sys. In the context of our scenario of one observer agent and one responder agent, the observer may observe any event from the set[1] $V \subseteq E$ (either as a sender or receiver of that event) whereas events in the set[2] $C \subseteq E$ are not observable and may represent internal events of the responder. In more complex scenarios, which events an agent might observe may be determined by a *classification* of events and a *clearance* of that agent, cf. [15] for more details. The observer's a priori knowledge about the responder is represented in the specification of the system in the components E, O, I and Sys which are known to the observer. After observing the visible events of a trace t as the sequence $t\mid_V$, that is, the sequence of all visible events in the trace t, the

[1] V refers to visible events.
[2] C refers to confidential events.

observer might gain further information about t by determining the pre-image

$$K(Sys, V, t) = \{t' \in Sys \mid t' \mid_V = t \mid_V\}. \tag{2.1}$$

This way, the observer exploits its a priori knowledge Sys and its observation $t \mid_V$. The observer is not able to distinguish any trace in the pre-image of (2.1) from the trace t after observing $t \mid_V$. Thus, the greater the pre-image the less information the observer gained about t.

Confidentiality and several other security requirements confine the information gain of the observer. A well known requirement is *non-interference* which says intuitively that the observer should be "virtually isolated" from the responder. Non-interference thus requires that for the observed sequence $t \mid_V$ of events from any trace $t \in Sys$ the pre-image determined from the observed sequence must contain the trace $t \mid_V$. This means that the observer cannot distinguish trace t from another trace t' by observing the visible events of the two traces where in t' no confidential event from C occurs at all. The reader may find examples of different requirements in [15, 58, 79].

In the following, we will outline the "modular assembly kit for security properties" (MAKS) proposed by Mantel in [79] for the specification of security requirements basing on this model of the observer's information gain. In MAKS several other requirements from the literature, including non-interference, may be specified. In the context of multiagent systems, Schairer in [86] uses MAKS for declaring an agent's interest to hide the value of some of its internal variables from another agent while exchanging messages.

In MAKS, the partition of events into visible events V, confidential events C and neither N is called a *view* $\mathcal{V} = (V, N, C)$ *of the observer*. Above, we considered such a partition, but only made explicit the sets V and C. Note that a view essentially defines the occurrence of which events the observer is allowed to infer (events in $V \cup N$) and the observer is not allowed to infer (events in C).

A security requirement is defined by a view and a *security predicate* which roughly spoken requires a system to contain perturbations of traces to hide the occurrence or non-occurrence of events given the observer's view. Mantel [79] summarizes the idea behind security predicates as follows. "A security predicate defines (...) under which conditions can we say that no information about occurrences or non occurrences of events in C can be deduced from observations of events in V. Hence, a security predicate can be understood as a definition of what critical information flow means ...." Formally, a security predicate is a function $\mathtt{SP}_{\mathcal{V}} : \mathcal{P}(E^*) \rightarrow \{true, false\}$. Further, security predicates are "assembled" from basic security predicates that are *closure properties* of a set of traces. A closure property is a property of a set of traces such that each set of traces has a superset owning this property. Or in other words [15], "Whenever some trace is in the set, some further traces of a specific form are as well". We will illustrate how to use basic security predicates to assemble security requirements with the already introduced example of non-interference. In [79, Chapter 4.2], Mantel shows that non-interference can be expressed in MAKS with the basic security predicate $\mathtt{R}_{\mathcal{V}}$ ($\mathtt{R}$ for removal) defined as follows. Let $Tr \subseteq Sys$ be a set of traces in system Sys.

$$\mathtt{R}_{\mathcal{V}}(Tr) = \forall t \in Tr\ \exists t' \in Tr \text{ such that } t' \mid_C = \langle\rangle \text{ and } t' \mid_V = t \mid_V . \qquad (2.2)$$

Then, non-interference of system Sys given view $\mathcal{V}$ is equivalent to the satisfaction of $\mathtt{R}_{\mathcal{V}}(Sys)$, cf. [79, Theorem 4.2.20]. The predicate requires that for every trace t in a system there exists another trace t' in the system that the observer is not able to distinguish from t by observing the events in V and in that all confidential events are removed. Mantel describes this requirement from another perspective: Each trace t in the system may be perturbed by removing all confidential events C from t and possibly adding some events from the set N (neither confidential nor observable events); then, the trace obtained by these modifications must be in the system. Other basic security predicates vary in the required perturbations; Mantel gives various basic security predicates in [79] and, based thereupon, assembles them to security predicates that are equivalent to known security requirements from the literature. In Section 3.5, we will relate confidentiality properties used in this thesis to the trace-based model of the observer's information gain using MAKS.

2.3 Runs & Systems

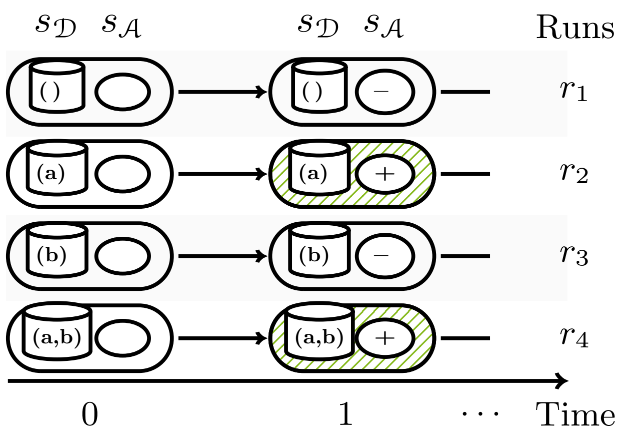
In this section, the observer's information gain is described in terms of its knowledge about the system. This way, security requirements may be formulated as requirements of what the observer should not know.

Halpern and O'Neill proposed *Runs & Systems* introduced by Fagin et al in [48] as a general model for multiagent systems to formulate various notions of secrecy [58]. A lot of security requirements in the trace-based model of information gain from Section 2.2 have been reformulated in this framework by Halpern and O'Neill in the same paper. In further work [57], Halpern and O'Neill describe several notions of *anonymity* in the same framework. All these examples show that the Runs & Systems framework is rather expressive for specifying the confinement of information flow. Moreover, Runs & Systems defines the semantics of a *logic of knowledge and time* by means of which the confinement of information flow may be declared in a more convenient way, cf. Section 2.5. Also, Halpern and O'Neill extended the basic framework to probability and plausibility measures and corresponding notions of secrecy, cf. Section 2.4.
In [58] the information accessible to an agent $\mathcal{X}$ is encoded in its local state $s_{\mathcal{X}}$ in the description of the system. A system is statically described by a global state $(s_e, s_{\mathcal{X}_1}, \ldots, s_{\mathcal{X}_n})$, that is the collection of all agent's local states $s_{\mathcal{X}_1}, \ldots, s_{\mathcal{X}_n}$ and optionally an environment state s_e, and dynamically as a set of *runs*. A *run* r is a function from discrete time $\mathbb{N}_0$ to global states. Thus, a run represents one way the system may evolve over time. A system $\mathcal{R}$ is a collection of runs and the set $\mathcal{PT}(\mathcal{R}) := \{(r, m) \mid r \in \mathcal{R}, m \in \mathbb{N}_0\}$ is called the *points* of system $\mathcal{R}$. The current view of agent $\mathcal{X}$ on the system $\mathcal{R}$ at time m in a run r is

$$\mathcal{K}_{\mathcal{X}}(r, m) = \{(r', m') \in \mathcal{PT}(\mathcal{R}) \mid r_{\mathcal{X}}(m) = r'_{\mathcal{X}}(m')\}, \tag{2.3}$$

where $r_{\mathcal{X}}(m)$ denotes the local state of agent $\mathcal{X}$ in the global state $r(m)$. In words, the set $\mathcal{K}_{\mathcal{X}}(r, m)$ is the set of points in which agent $\mathcal{X}$ has the same local state as in (r, m). As the agent's local state represents the runtime information accessible to the agent, the set $\mathcal{K}_{\mathcal{X}}(r, m)$ is called $\mathcal{X}$-*information set* and models agent $\mathcal{X}$'s reasoning about the system at point (r, m) in the following sense. In run r at time m agent $\mathcal{X}$ is not able to distinguish all points in the set $\mathcal{K}_{\mathcal{X}}(r, m)$ by its runtime information in $r_{\mathcal{X}}(m)$ whereas it rules out that the system $\mathcal{R}$ is in any point from the set $\mathcal{PT}(\mathcal{R}) \setminus \mathcal{K}_{\mathcal{X}}(r, m)$. This way, the agent reasons about possible states of the system. In this sense, the set $\mathcal{K}_{\mathcal{X}}(r, m)$ contains all *possible system states* from agent $\mathcal{X}$'s point of view in run r at time m. If we express an agent $\mathcal{X}$'s reasoning about system $\mathcal{R}$ this

Figure 2.1: A system with two agents: a responder with a database and an observer issuing queries to that database.



way, we implicitly assume that agent $\mathcal{X}$ knows the system specification $\mathcal{R}$ and is able to determine the set $\mathcal{K}_{\mathcal{X}}(r,m)$ for all $(r,m) \in \mathcal{PT}(\mathcal{R})$.

In the following example, we describe the focused agent scenario in this thesis, cf. Section 1.2, within Runs & Systems, but in a simple form.

Example 2.1 (Two interacting agents)**.**

To illustrate the concepts in this chapter, we consider a small example with very simple agents: a reacting agent $\mathcal{D}$, the "responder", with a local database and a requesting agent $\mathcal{A}$, the "observer". Agent $\mathcal{A}$ may query agent $\mathcal{D}$'s database to which $\mathcal{D}$ replies in return. Agent $\mathcal{D}$'s local database stores information in form of two propositions a and b. Its states are taken from the set $\mathcal{St}_{\mathcal{D}} = \{(), (a), (b), (a,b)\}$. We interpret state (a) as "a holds and b does not hold" and the other states analogously. Figure 2.1 shows an excerpt of a system with these two agents. In this excerpt, agent $\mathcal{A}$ issues a query about proposition a at time 0 and receives the answer at time 1. Agent $\mathcal{A}$'s state ⊕ denotes a positive answer, whereas its state ⊖ denotes a negative answer. Agent $\mathcal{A}$ reasons in run r_2 at time 1 with its runtime information (positive answer) that the system must be in one of the hatched states. This reasoning is modeled by the information set $\mathcal{K}_{\mathcal{A}}(r_2,1) = \{(r_2,1),(r_4,1),\ldots\}$ which is represented by the hatched states.

In the following, we will survey Halpern and O'Neill's notion of secrecy by means of which the responder (or defender) agent $\mathcal{D}$ might express its security interests towards the observer (or attacker) agent $\mathcal{A}$. The scenario of two interacting agents of Example 2.1 will be elaborated to a general scenario in Chapter 4.
The strictest secrecy notion, total secrecy, requires that, in any run r at any time m agent $\mathcal{A}$ cannot determine with its available information $r_{\mathcal{A}}(m)$ and the known specification of the system $\mathcal{R}$ that $\mathcal{D}$ is currently not in some of the states specified in $\mathcal{R}$.

Definition 2.1 (Total Secrecy).
Cf. Definition 3.1 of [58]

Agent $\mathcal{D}$ maintains total secrecy with respect to agent $\mathcal{A}$ in system $\mathcal{R}$ if, for all points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R})$

$$\mathcal{K}_{\mathcal{A}}(r, m) \cap \mathcal{K}_{\mathcal{D}}(r', m') \neq \emptyset.$$

Note that the set $\mathcal{K}_{\mathcal{D}}(r', m')$ represents $\mathcal{D}$'s local state at run r' at time m' where $m' \neq m$ is possible since $\mathcal{D}$ might be in the same state at different times or $\mathcal{A}$ might even not know the current time m exactly.
Halpern and O'Neill weaken their notion *total secrecy*, which does not allow $\mathcal{A}$ to learn anything about $\mathcal{D}$'s local state, in particular under two aspects, first, *what* information about $\mathcal{D}$'s local state should be protected by *total f-secrecy* as well as *C-secrecy*, and, second, *when* the protection is needed by *C-secrecy*.
The idea of total f-secrecy is to keep secret "the significant part" of agent $\mathcal{D}$'s state. For example, item a in $\mathcal{D}$'s local database of Example 2.1 might be the significant part if it represents sensitive data. Formally, agent $\mathcal{D}$ may define pieces of information in its state with a function $f : \mathcal{PT}(\mathcal{R}) \to V$ that only depends on its state. The elements of V abstractly denote pieces of information about $\mathcal{D}$'s state. These pieces of information should be kept secret with the requirement of total f-secrecy.

Definition 2.2 ($\mathcal{D}$-information function f).
Cf. Definition 3.2 of [58]

A $\mathcal{D}$-information function on $\mathcal{R}$ is a function f from $\mathcal{PT}(\mathcal{R})$ to some range V that depends only on $\mathcal{D}$'s local state; that is, $f(r,m) = f(r',m')$ if $r_{\mathcal{D}}(m) = r'_{\mathcal{D}}(m')$.

Each abstract information value $v \in V$ describes a set of $\mathcal{D}$'s states. This set comprises all local states of $\mathcal{D}$ in the points of $f^{-1}(v)$ where $f^{-1}(v)$ is simply the preimage of v, that is all points (r',m') such that $f(r',m') = v$. Total f-secrecy requires that agent $\mathcal{A}$ cannot decide in any run r at any time m with its available information $r_{\mathcal{A}}(m)$ that $\mathcal{D}$ is not in a state defined by v.

Definition 2.3 (Total f-Secrecy).
Cf. Definition 3.3 of [58]

If f is a $\mathcal{D}$-information function, agent $\mathcal{D}$ maintains total f-secrecy with respect to agent $\mathcal{A}$ in system $\mathcal{R}$ if, for all points $(r,m) \in \mathcal{PT}(\mathcal{R})$ and values v in the range V of f

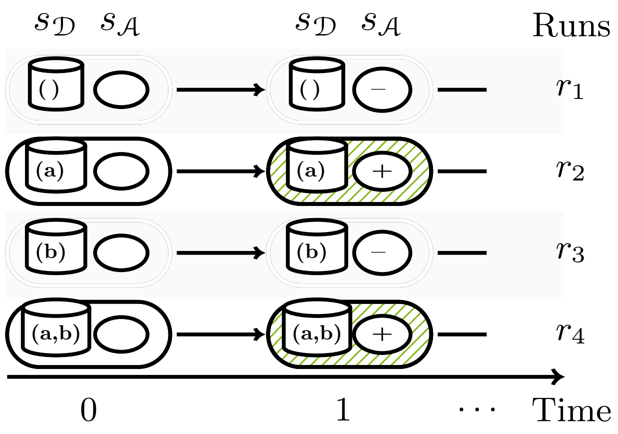
$$\mathcal{K}_{\mathcal{A}}(r,m) \cap f^{-1}(v) \neq \emptyset.$$

Example 2.2 (Total f-Secrecy).

We illustrate how to specify a $\mathcal{D}$-information function and how to check total f-secrecy in the setting of Example 2.1. The $\mathcal{D}$-information function is based on the abstract information v_1 with meaning "agent $\mathcal{D}$ stores item a" and v_2 with meaning "agent $\mathcal{D}$ does not store item a". The function is defined according to the meaning of the abstract pieces of information. Its definition is illustrated in Figure 2.2: The black bordered states are those mapped to v_1 while the double-line bordered states are those mapped to v_2. Total f-secrecy towards agent $\mathcal{A}$ is violated in run r_2 at time 1 because with the positive answer agent $\mathcal{A}$ excludes v_2, formally, $\mathcal{K}_{\mathcal{A}}(r_2,1) \cap f^{-1}(v_2) = \emptyset$.

Another weakening of total secrecy is C-secrecy. The idea of C-secrecy is, that at a point (r,m) agent $\mathcal{D}$ can locally allow agent $\mathcal{A}$ to exclude some of its states, that are all of agent $\mathcal{D}$'s states in the points $\mathcal{PT}(\mathcal{R}) \setminus C(r,m)$ defined by an allowability function C. The set $C(r,m)$ may define points that are relevant for the protection of sensitive information about $\mathcal{D}$'s state. For

Figure 2.2: Violation of total f-secrecy



$s_{\mathcal{D}}$ $s_{\mathcal{A}}$
$s_{\mathcal{D}}$ $s_{\mathcal{A}}$
Runs
()
()
−
r_1
(a)
(a)
+
r_2
(b)
(b)
−
r_3
(a,b)
(a,b)
+
r_4
0
1
…
Time

example, resuming Example 2.2, if $\mathcal{A}$ should not know that $\mathcal{D}$ stores item a with sensitive data, then for all runs r and times m agent $\mathcal{A}$ should not rule out that the system is in one of the points in $C(r,m) = f^{-1}(v_2)$ (represented by the double-line bordered states in Figure 2.2).

Definition 2.4 ($\mathcal{A}$-allowability function C).
Cf. Definition 3.5 of [58]

An $\mathcal{A}$-allowability function on $\mathcal{R}$ is a function C from $\mathcal{PT}(\mathcal{R})$ to subsets of $\mathcal{PT}(\mathcal{R})$ such that $\mathcal{K}_{\mathcal{A}}(r,m) \subseteq C(r,m)$ for all $(r,m) \in \mathcal{PT}(\mathcal{R})$.

C-Secrecy requires that in any run r at any time m agent $\mathcal{A}$ should not be able to decide with its available information $r_{\mathcal{A}}(m)$ that $\mathcal{D}$ is not in one of the states given by $C(r,m)$.

Definition 2.5 (C-secrecy).
Cf. Definition 3.6 of [58]

If C is an $\mathcal{A}$-allowability function, then $\mathcal{D}$ maintains C-secrecy with respect to agent $\mathcal{A}$ if, for all points $(r,m) \in \mathcal{PT}(\mathcal{R})$ and $(r',m') \in C(r,m)$, we have

$$\mathcal{K}_{\mathcal{A}}(r,m) \cap \mathcal{K}_{\mathcal{D}}(r',m') \neq \emptyset.$$

The last notion we will consider in this thesis is run-based secrecy. Run-based secrecy does no longer assume agent $\mathcal{A}$ to reason about possible system states but about possible runs. Formally, the reasoning in terms of system states is modeled by the operator $\mathcal{K}$ whereas the following operator[3] $\mathcal{R}$ is used to model reasoning in terms of runs of the system. Let $P \subseteq \mathcal{PT}(\mathcal{R})$ be a set of points. Then, the set of runs through these points is defined as

$$\mathcal{R}(P) = \{r \in \mathcal{R} \mid \text{there exists time } m \text{ such that } (r, m) \in P\}. \tag{2.4}$$

Based on this set, an agent $\mathcal{X}$'s reasoning about possible runs of the system may be described as follows: With its available information $r_{\mathcal{X}}(m)$ at runtime in any run r at any time m the agent $\mathcal{X}$ is not able to distinguish which of the runs $\mathcal{R}(\mathcal{K}_{\mathcal{X}}(r, m))$ is the current run r of the system.
Run-based secrecy postulates an attacker $\mathcal{A}$ with run-based reasoning by operator[4] $\mathcal{R} \circ \mathcal{K}_{\mathcal{A}}$. The requirement of run-based secrecy is that for any state s of $\mathcal{D}$ defined in system $\mathcal{R}$, with its available information $r_{\mathcal{A}}(m)$, agent $\mathcal{A}$ is not able to infer that $\mathcal{D}$ never is in state s throughout the current run.

Definition 2.6 (Run-based Secrecy).
Cf. Definition of [58]

Agent $\mathcal{D}$ maintains run-based secrecy with respect to agent $\mathcal{A}$ in system $\mathcal{R}$ if, for all points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R})$ it holds

$$\mathcal{R}(\mathcal{K}_{\mathcal{A}}(r, m)) \cap \mathcal{R}(\mathcal{K}_{\mathcal{D}}(r', m')) \neq \emptyset.$$

In Chapter 3, we will extend Halpern and O'Neill's notions to possibility policies which are more expressive to declare confidentiality properties.

2.4 Probabilities

The observer may not only consider possible that the responder is in some state but may assign a probability to this fact. The probability expresses the agent's uncertainty about the fact. Then, the observer's information gain may be described by comparing a priori and a posteriori probabilities, cf. [15].

[3] We use the same notation as the authors of [58] although this notation overloads the symbol $\mathcal{R}$. However, it should be clear from the context whether $\mathcal{R}$ refers to a set of runs or to the operator.

[4] The operator $\circ$ denotes the composition of two functions.

There are several approaches in the literature which define an agent's probabilistic reasoning about a computing system and give related means to specify security requirements of the system, cf. [58, 41, 4] as a small selection. In the following, we will briefly survey these approaches which might be relevant to refine the model of the attacker's information gain with probabilities in future work beyond this thesis.

Halpern and O'Neill [58] propose agents in the Runs & Systems framework which throughout each run of the system attribute probabilities to the system being in some global state. Formally, in each point (r, m) of system $\mathcal{R}$ an agent $\mathcal{X}$ has a probability space with probability measure $\mu_{r,m,\mathcal{X}}$ on measurable sets of points. In practice, probability spaces may be derived from probabilities that the system executes particular runs, cf. [58]. Throughout a run, an agent's probability space might change. For example, the agent $\mathcal{X}$'s probability measures $\mu_{r,m,\mathcal{X}}$ and $\mu_{r,m+1,\mathcal{X}}$ might be different in run r at time m and at time $m+1$, respectively. Requirements on information flow confinement must weigh the probabilistic information of agents beside the runtime information in their local states. Halpern and O'Neill define the strict property of *probabilistic total secrecy*. An agent $\mathcal{D}$ maintains this property with respect to another agent $\mathcal{A}$ iff the probability that $\mathcal{A}$ attributes to $\mathcal{D}$ being in a particular state is constant.

Other approaches in the literature measure information gain by different measures of *entropy*, e.g. [41, 4]. Clarkson et al [41] quantify the information that is leaked to the observing agent by the output of an algorithm executed by the responding agent. The observer may assign values to public input variables of the algorithm and observe their values after execution. Additionally, the responder has a hidden assignment of other input variables. The focus is on how the observer *revises its belief* about the responder's hidden variable assignment after observing the output values. The observer's belief is a probability distribution on assignments and usually differs from the actual distribution. Now, the observing agent's information gain about the responders's hidden variable assignment is rated as follows. The observer usually is uncertain or even mistaken about the responder's actual assignment. The *accuracy* of the observer's postulated distribution compared to the responder's actual distribution is measured by relative entropy (also called

Kullback-Leibler divergence). Information gain is the increase of accuracy after the response.
Several other entropy-based approaches are surveyed by Alvim et al in [4]. Many of these approaches define information leakage as the difference of the attacking agent's uncertainty before and after observing the responding agent. Uncertainty is seen as the effort spent in guessing.[5] Which entropy is best to measure the difference, depends on how the attacking agent's abilities to interact with the responder are understood and how its efforts to gain some desired piece of information are rated. For example, the attacking agent may query the responding agent whether an attribute has some value or not while it is interested in the actual value of the attribute. The effort is the expected numbers of queries. Here, the attacking agent does not become certain about the actual value because the agent has only a very limited way to interact with the responder.

2.5 Logics of Knowledge and Time

Several approaches define confidentiality properties in a *logic of knowledge and time* as introduced by Fagin et al in [48] to express properties of a system. The reader may find examples of these approaches in [58, 8, 7] or in [92] from the field of deontic logic. These approaches formulate the attacker's information gain in terms of the attacker's *knowledge* which can be expressed with a modality **K** in the logic. Then, confidentiality properties are syntactical expressions that confine the attacker's knowledge. Advantages of these approaches are the conciseness of presentation of those properties and their *automated verification*, for example, by model checking [7].
At first, we illustrate the main concepts of a logical formalization of confidentiality properties with an outline of Halpern and O'Neill's work in [58]. Those concepts are common to other approaches like [8, 7]. At the end of this section, we briefly survey research results on the automated verification of logically expressed confidentiality properties.
Halpern and O'Neill show in [58] that several of their notions of secrecy can be

[5] This understanding of information leakage is criticized by Clarkson et al in the work [41] summarized above.

expressed as equivalent requirements in terms of the syntax of this logic. We recapitulate the logic of knowledge and time as used by Halpern and O'Neill in brief: A formula F is built by

$$F :: \equiv A | F \vee F | F \wedge F | \mathbf{K}_{\mathcal{X}} F | \Diamond F$$

where A is an atomic proposition from alphabet $\mathcal{A}t$ and $\mathcal{X}$ is an agent identifier. The meaning of a formula F is given by an interpreted system:

Definition 2.7 (Interpreted System $\mathcal{I}$)**.**
Cf. Section 2 of [58]

Let $\mathcal{R}$ be a system and $\mathcal{A}t$ an alphabet of atomic propositions. An interpretation π of $\mathcal{A}t$ in $\mathcal{R}$ is a function

$$\pi : \mathcal{A}t \times \Sigma \to \{true, false\}$$

where Σ is the set of all global states in $\mathcal{R}$. The pair $(\mathcal{R}, \pi)$ is called an interpreted system and denoted by $\mathcal{I}$.

We denote the model-of operator for interpreted systems by $\models_{IS}$. The interpretation of the propositional connectives is as usual and the modalities are interpreted as follows:

$$\begin{array}{llllr} (\mathcal{I}, r, m) \models_{IS} \mathbf{K}_{\mathcal{X}} F & \text{iff} & (\mathcal{I}, r', m') \models_{IS} F & \text{for all} & (r', m') \in \mathcal{K}_{\mathcal{X}}(r, m), \\ (\mathcal{I}, r, m) \models_{IS} \Diamond F & \text{iff} & (\mathcal{I}, r, n) \models_{IS} F & \text{for some} & n \in \mathbb{N}_0. \end{array}$$

In order to give an impression how this logic helps to express confidentiality properties we cite a result from [58]:

> Agent $\mathcal{D}$ maintains total secrecy with respect to $\mathcal{A}$ in system $\mathcal{R}$ iff
> for every interpretation π
> if F is a $\mathcal{D}$-local formula[6] in the interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$
> and true in at least one point of system $\mathcal{R}$, then $\mathcal{I} \models_{IS} \neg \mathbf{K}_{\mathcal{A}} F$.[7]

The above results says that $\mathcal{A}$ may not know any information inherent in some of $\mathcal{D}$'s states.

[6] That is, a formula whose evaluation depends only on the state of agent $\mathcal{D}$ in $\mathcal{I}$.

[7]The expression $\mathcal{I} \models_{IS} G$ denotes the validity of formula G in the interpreted system $\mathcal{I}$, that is for all points (r, m) it holds $(\mathcal{I}, r, m) \models_{IS} G$.

Balliu et al in [8] demonstrate the expressiveness of the logic of knowledge and time by the formalization of *declassification* properties within this logic. Declassification properties relax strong requirements that confine the attacker's information gain such as non-interference of the trace-based model in Section 2.2. Roughly, these properties define in what situations the attacker may know what properties of the defender's state.
At the end, we summarize a recent result of Balliu et al [7] on the automated verification of confidentiality properties. The summary should make clear in which contexts automated verification might be applied. Balliu et al [7] consider that the defender executes a sequential while-program and aims to hide properties of hidden input variables of the program from the attacker. In turn, the attacker may observe output from the program, but may not contribute any input value. In this setting, Balliu et al in [8] use atomic propositions of the form $\mathsf{Init}_x(e)$ in the logic of knowledge and time with the meaning that the initial value of variable x equals the current value of expression e (and beside atoms of this form they use only the equality of expressions). In [7], Balliu et al use abstract properties of the initial value of x as atomic propositions. The prototype described in [7] is able to do automated verification for any Java program that uses integer computations only.
The automated verification builds an execution tree, which represents the control flow of the program on all input states, and based thereupon builds a symbolic output tree, which represents under which condition (attached to its edges) the execution yields which properties of the input variables (these properties are attached to its nodes). Finally, the symbolic output tree is transformed into an interpreted system and the syntactic confidentiality property is verified with a model checker. Alternatively, the symbolic output tree and the confidentiality property can be transformed into a formula of the existential fragment of first order logic whose satisfiability can be tested with dedicated solvers.

Chapter 3

Policy-Based Secrecy

In this chapter we introduce the two notions of *policy-based secrecy* and *policy-based run-based secrecy* in line of the work [58] by Halpern and O'Neill surveyed in Section 2.3. These notions will be adequate to formally specify the defending agent $\mathcal{D}$'s confidentiality interests in terms of a model of the multiagent system in Runs & Systems including a postulated attacking agent $\mathcal{A}$. The secrecy notions proposed by Halpern and O'Neill do not suffice for this purpose as we will argue in Chapter 4. Our two proposed notions adopt Halpern and O'Neill's models of $\mathcal{A}$'s information gain: On the one hand, policy-based secrecy postulates $\mathcal{A}$ as a reasoner about possible points with operator $\mathcal{K}_{\mathcal{A}}$. On the other hand, policy-based run-based secrecy postulates $\mathcal{A}$ as a reasoner about possible runs with operator $\mathcal{R} \circ \mathcal{K}_{\mathcal{A}}$. Thus, in particular, policy-based secrecy and policy-based run-based secrecy will be possibilistic confidentiality properties, cf. Chapter 2.
Beyond Halpern and O'Neill's framework of [58], the proposed two notions of secrecy base on a *local* declaration of the defender, called *possibility policy*. Such a policy may specify

- properties of global states,
- properties of the defender's local state
- or properties of runs for the protection of confidential information.

First, in Section 3.1, we introduce possibility policies about states and, then, their semantics by policy-based secrecy in Section 3.2.

In Section 3.3, we argue that declaring the defender's interests with possibility policies is more flexible than the declarations proposed by Halpern and O'Neill. As one contribution within this chapter, we show that all of Halpern and O'Neill's possibilistic notions of secrecy can be expressed by policy-based secrecy.
Then, in Section 3.4, we briefly study declaring confidentiality interests towards $\mathcal{A}$ as a reasoner about possible runs with a possibility policy about runs. The semantics of such a policy will be defined by policy-based run-based secrecy. Yet, as an immediate result, we will see that policy-based run-based secrecy can be expressed by policy-based secrecy, but is less expressive.
Figure 3.1 summarizes our results on the expressivity of policy-based secrecy.
Finally, in Section 3.4, basing on Halpern and O'Neill's achievements, we relate policy-based secrecy and policy-based run-based secrecy to other requirements of information flow confinement in the literature.

3.1 Possibility Policies and $\mathcal{D}$-Properties

A formal declaration of the defender's confidentiality interests is, according to Chapter 2, a requirement to confine the attacker's information gain. Here, agent $\mathcal{D}$ *locally declares* in each point $(r, m) \in \mathcal{PT}(\mathcal{R})$ which information about the global state $r(m)$ agent $\mathcal{A}$ as a potential attacker must consider possible in a *possibility policy*. As a special case, a piece of information about its local state is expressed as a *$\mathcal{D}$-property*. Intuitively, a $\mathcal{D}$-property I describes some property of agent $\mathcal{D}$'s state. In particular, in each point (r, m) of the system agent $\mathcal{D}$ is able to determine whether $(r, m) \in I$ holds or not, given its local state $r_{\mathcal{D}}(m)$ and the definition of I, by checking $\mathcal{K}_{\mathcal{D}}(r, m) \subseteq I$.

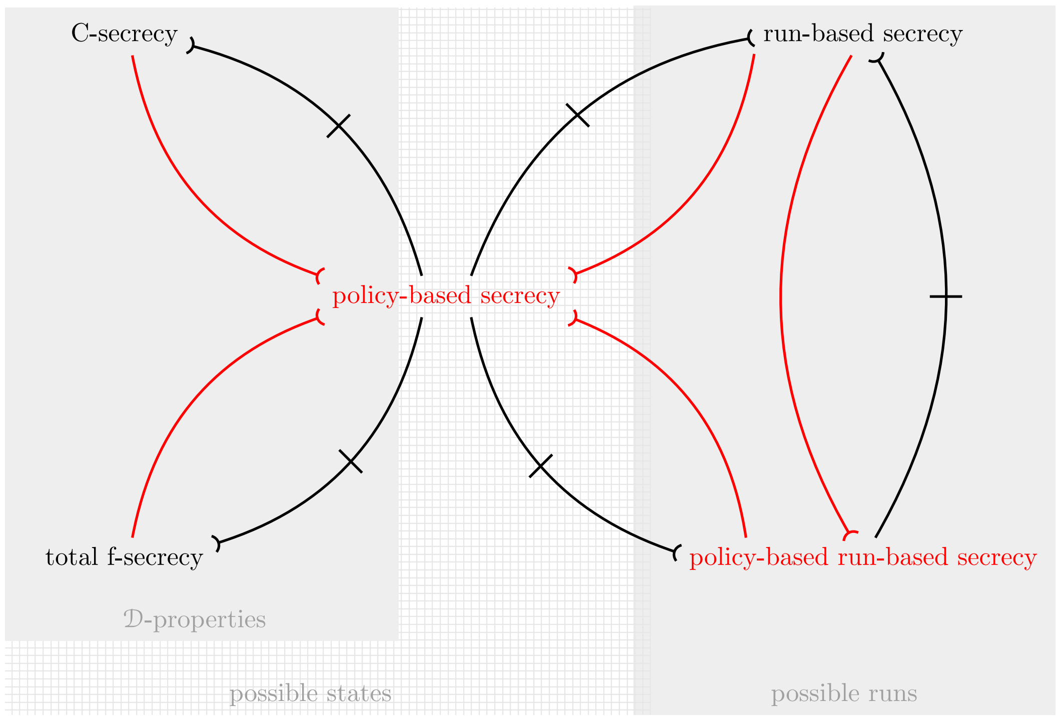
Definition 3.1 ($\mathcal{D}$-Property).

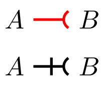
A $\mathcal{D}$-property is a set $I \subseteq \mathcal{PT}(\mathcal{R})$ such that for all $(r, m) \in I$ it follows $\mathcal{K}_{\mathcal{D}}(r, m) \subseteq I$.[1]

Whereas a $\mathcal{D}$-information set (2.3) represents all the information agent $\mathcal{D}$ has in its local state about the system and as such the local state itself, a

[1] The converse implication always holds: If $\mathcal{K}_{\mathcal{D}}(r, m) \subseteq I$ holds, then we obtain $(r, m) \in I$ because $(r, m) \in \mathcal{K}_{\mathcal{D}}(r, m)$ is immediate from the definition of the operator $\mathcal{K}_{\mathcal{D}}$ in (2.3).

Figure 3.1: Expressivity of policy-based secrecy





Property A **can** be expressed in terms of property B

Property A **cannot** be expressed in terms of property B

$\mathcal{D}$-property represents a property agent $\mathcal{D}$ may have in several of its states. All secrecy definitions by Halpern and O'Neill concern only $\mathcal{D}$-properties. They remark that "any 'secret information' that cannot be characterized as a local proposition is not protected" by their definitions. Further, $\mathcal{D}$-properties reappear in Chapter 4 in the declarations of the confidentiality aims of an epistemic agent $\mathcal{D}$.
Throughout this chapter, we will use a running example to illustrate the concepts.

Example 3.1 ($\mathcal{D}$-Properties)**.**

We take up the setting from Example 2.1. There, a reacting agent $\mathcal{D}$ has a local database which may store items (propositions) a and b and responds to queries of another agent $\mathcal{A}$. The setting is visualized in Figure 3.2. In this setting, we consider the following $\mathcal{D}$-properties:

- $I_{(a)} = \mathcal{K}_{\mathcal{D}}(r_2, 0)$*: agent $\mathcal{D}$ is in state (a), visualized by* ⛁*.*
- $I_{(b)} = \mathcal{K}_{\mathcal{D}}(r_3, 0)$*: agent $\mathcal{D}$ is in state (b), visualized by* ⛁*.*
- $I_{(a),(b)} = \mathcal{K}_{\mathcal{D}}(r_2, 0) \cup \mathcal{K}_{\mathcal{D}}(r_3, 0)$*: agent $\mathcal{D}$ is in state (a)* or *in state (b). The global states where $\mathcal{D}$ has this property are drawn with double-line borders in Figure 3.2.*
- $I_{(a,b)} = \mathcal{K}_{\mathcal{D}}(r_4, 0)$*: agent $\mathcal{D}$ is in state (a, b), visualized by* ⛁*.*

Depending on run r and time m, agent $\mathcal{D}$'s possibility policy declares properties of interest attributed to the global state or to $\mathcal{D}$'s local state as a special case. A property of interest might be, like in the previous example, that agent $\mathcal{D}$ stored a relevant data item. In our context, where agent $\mathcal{D}$ has confidentiality interests, a $\mathcal{D}$-property represents no confidential information itself, but is essential for the protection of confidential information which we will shortly illustrate in Example 3.2 below.

Definition 3.2 (Syntax of Possibility Policy and $\mathcal{D}$-Possibility Policy)**.**

A possibility policy is a function `policy` : $\mathcal{PT}(\mathcal{R}) \to \mathcal{P}(\mathcal{P}(\mathcal{PT}(\mathcal{R})))$*, where $\mathcal{P}$ denotes the power set operation. If further for all $(r, m) \in \mathcal{PT}(\mathcal{R})$ the set* `policy`$(r, m) := \{I_1, I_2 \ldots\}$ *contains only $\mathcal{D}$-properties I_k, then* `policy` *is* *called a $\mathcal{D}$-possibility policy and we write* $\texttt{policy}\mathcal{D}$. *Both the sets* $\texttt{policy}(r,m)$ *and their element sets I_k may be infinite.*

Example 3.2 ($\mathcal{D}$-Possibility Policies).

We continue Example 3.1 visualized in Figure 3.2. In this setting, agent $\mathcal{D}$ may declare some $\mathcal{D}$-possibility policies with the $\mathcal{D}$-properties defined in that example. For all $(r,m) \in \mathcal{PT}(\mathcal{R})$ the policies are defined as follows:

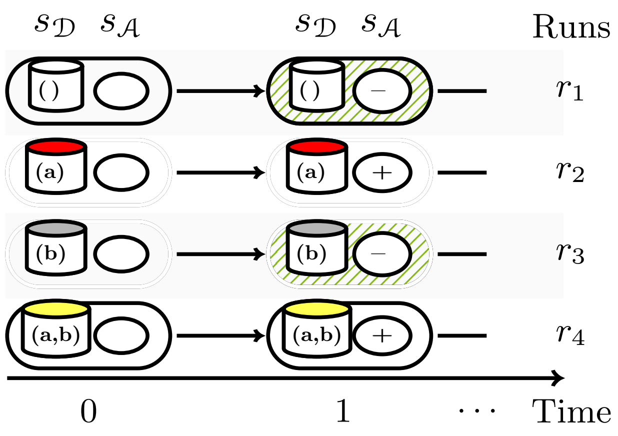
- $\texttt{policy}_1(r,m) := \{I_{(a)}, I_{(b)}\}$,
- $\texttt{policy}_2(r,m) := \{I_{(a),(b)}\}$,
- $\texttt{policy}_3(r,m) := \{I_{(a),(b)}, I_{(a,b)}\}$.

In our example, all policies do not depend on the current run r and time m and are thus rather global than local policies. This is not necessary but simplifies our presentation. The possibility policies get an intuitive meaning if we understand that they state confidentiality interests of agent $\mathcal{D}$:
By $\texttt{policy}_1$, *agent $\mathcal{D}$ expresses the following confidentiality interests*

- *by $I_{(a)}$: If the fact $\neg a \vee b$ holds in agent $\mathcal{D}$'s database (with the usual propositional semantics), this situation must be kept confidential.*
- *by $I_{(b)}$: Analogously for the fact $a \vee \neg b$.*

The semantics of possibility policies will be defined in the next section. Under that semantics, policy $\texttt{policy}_1$ *implicitly does not allow $\mathcal{A}$ to be certain that the system is in one of the points $\mathcal{PT}(\mathcal{R}) \setminus I_{(a)}$. In these points, agent $\mathcal{D}$ has one of the states in the set $\mathcal{St}_{\mathcal{D}} \setminus \{(a)\} = \{(), (b), (a,b)\}$. In these states and only these the fact $\neg a \vee b$ holds in $\mathcal{D}$'s database. Thus, policy* $\texttt{policy}_1$ *implicitly does not allow $\mathcal{A}$ to be certain that the fact $\neg a \vee b$ holds in $\mathcal{D}$'s database. Likewise, the policy states the confidentiality of the fact $a \vee \neg b$ which holds in $\mathcal{D}$'s database in states $\mathcal{St}_{\mathcal{D}} \setminus \{(b)\} = \{(), (a), (a,b)\}$.*
Finally, we list the confidentiality interests expressed by the remaining possibility policies

- $\texttt{policy}_2$ *by $I_{(a),(b)}$: If $a \equiv b$ holds in the database (corresponding to the set of states $\mathcal{St}_{\mathcal{D}} \setminus \{(a),(b)\} = \{(), (a,b)\}$), this situation must be kept confidential.*



$s_{\mathcal{D}}$ $s_{\mathcal{A}}$
$s_{\mathcal{D}}$ $s_{\mathcal{A}}$
Runs
()
()
−
r_1
(a)
(a)
+
r_2
(b)
(b)
−
r_3
(a,b)
(a,b)
+
r_4
0
1
· · ·
Time

Figure 3.2: The graphics shows the $\mathcal{D}$-properties I_a by 🟥, $I_{(b)}$ by ▭, $I_{(a),(b)}$ by ◎ and $I_{(a,b)}$ by 🟨. Agent $\mathcal{A}$'s reasoning in run r_1 at time 1 is about the possible points visualized by ▨.

- `policy`$_3$ *by* $I_{(a),(b)}$*: Analogously.*
- `policy`$_3$ *by* $I_{(a,b)}$*: Analogously for the fact* $\neg a \vee \neg b$ *which holds in* $\mathcal{D}$*'s database in the states* $\mathcal{St}_{\mathcal{D}} \setminus \{(a,b)\} = \{(), (a), (b)\}$.

3.2 Policy-based Secrecy

In this section, we define the semantics of a possibility policy which may be a $\mathcal{D}$-possibility policy as a special case. With this semantics, the security engineer postulates the attacking agent $\mathcal{A}$ to be a reasoner about possible states of the multiagent system and, in particular, about agent $\mathcal{D}$'s state.[2] In this context, a possibility policy describes a set of properties that the attacker $\mathcal{A}$ should not rule out about the system's global state when reasoning with operator $\mathcal{K}_{\mathcal{A}}$ of (2.3). As a special case, a $\mathcal{D}$-possibility policy specifies properties not to be ruled out about $\mathcal{D}$'s local state. Thus, in every point $(r, m) \in \mathcal{PT}(\mathcal{R})$ and for every property I declared in the local policy `policy`(r, m) there is a possible system state $r'(m')$ from agent $\mathcal{A}$'s point of view (that is $(r', m') \in \mathcal{K}_{\mathcal{A}}(r, m)$) such that this state has property I (that is $(r', m') \in I$). This requirement is formalized by policy-based secrecy.

[2] The same postulate is made by total secrecy in Def. 2.1, total f-secrecy in Def. 2.3 and C-secrecy in Def. 2.5.

Definition 3.3 (Semantics of Possibility Policy: Policy-based Secrecy).

Let `policy` *be a possibility policy. Then agent* $\mathcal{D}$ *maintains policy-based secrecy with respect to agent* $\mathcal{A}$ *in* $\mathcal{R}$ *if, for all points* $(r, m) \in \mathcal{PT}(\mathcal{R})$ *and for all* $I \in$ `policy`(r, m)*:*

$$\mathcal{K}_{\mathcal{A}}(r, m) \cap I \neq \emptyset$$

Example 3.3 (Policy-based Secrecy).

We continue Example 3.2 and discuss the preservation of the $\mathcal{D}$*-possibility policies defined in that example under policy-based secrecy. The example should be followed with Figure 3.2.*
The policy $\texttt{policy}_1 = \{I_{(a)}, I_{(b)}\}$ *is violated in point* $(r_1, 1)$ *because*

$$\mathcal{K}_{\mathcal{A}}(r_1, 1) \cap I_{(a)} = \emptyset$$

which says that in $\mathcal{A}$*'s state* $(-)$ *agent* $\mathcal{A}$ *can rule out state* (a) *of agent* $\mathcal{D}$*.*
In contrast, $\texttt{policy}_2 = \{I_{(a),(b)}\}$ *is not violated in point* $(r_1, 1)$ *because*

$$\mathcal{K}_{\mathcal{A}}(r_1, 1) \cap I_{(a),(b)} \supseteq \{(r_3, 1)\} \neq \emptyset.$$

The reason is that agent $\mathcal{A}$ *still considers possible that* $\mathcal{D}$ *is in state* (b)*.*
Policy $\texttt{policy}_3$ *again is not preserved because* $\mathcal{K}_{\mathcal{A}}(r_1, 1) \cap I_{(a,b)} = \emptyset$*.*

3.3 Relation to Total f-Secrecy and C-Secrecy

In this section, we argue and formally prove that the concept of $\mathcal{D}$-possibility policies with the semantics of policy-based secrecy generalizes $\mathcal{D}$-information functions with total f-secrecy (and, thus, total secrecy, cf. [58]) and $\mathcal{A}$-allowability functions with C-secrecy.
Recall from Section 2.3 that $\mathcal{D}$-information functions define pieces of information about $\mathcal{D}$'s local state to be kept secret. Each piece v of information (from an abstract set V of values) attributed to $\mathcal{D}$'s local state by some $\mathcal{D}$-information function can been seen as defining a $\mathcal{D}$-property I_v, but the function defines mutually exclusive properties. More precisely, we translate a $\mathcal{D}$-information function f to a $\mathcal{D}$-possibility policy as follows:

$$\texttt{policy}_f(r, m) = \{I_v \mid v \in V\} \text{ with } I_v = \bigcup_{(r', m') : f(r', m') = v} \mathcal{K}_{\mathcal{D}}(r', m') \tag{3.1}$$

First, note that I_v is a $\mathcal{D}$-property because f is a $\mathcal{D}$-information function (which, by definition, depends only on $\mathcal{D}$'s local state). Second, we observe that $\texttt{policy}_f$ is a *global* policy since it is a constant so that $\texttt{policy}_f$ depends neither on r nor on m. Third, we observe that, because f can be understood as a function on $\mathcal{D}$'s state, the $\mathcal{D}$-properties I_v in $\texttt{policy}_f$ induce a partition of agent $\mathcal{D}$'s states (so that they are mutually exclusive). Vice versa a partition of agent $\mathcal{D}$'s states induces a function on its states. Hence, as concluded by the following Proposition 3.1, total f-secrecy corresponds to policy-based secrecy for the class of all global policies where elements of the policy are $\mathcal{D}$-properties defining a partition on $\mathcal{D}$'s states.

Proposition 3.1 (Total f-Secrecy Is $\texttt{policy}_f$-based Secrecy)**.**

Let f be a $\mathcal{D}$-information function and $\texttt{policy}_f$ the resulting policy given by Equation (3.1)*. Then, agent $\mathcal{D}$ maintains total f-secrecy with respect to agent $\mathcal{A}$ in $\mathcal{R}$ iff $\mathcal{D}$ maintains $\texttt{policy}_f$-based secrecy with respect to $\mathcal{A}$ in $\mathcal{R}$.*

Contrary to total f-secrecy with a global function f, C-secrecy bases on a local declaration called $\mathcal{A}$-allowability function C. Recall from Section 2.3 that an $\mathcal{A}$-allowability function C allows agent $\mathcal{A}$, depending on run r and time m, to rule out all of agent $\mathcal{D}$'s state in the points $\mathcal{PT}(\mathcal{R}) \setminus C(r,m)$. The remaining points may be relevant for the protection of other pieces of information about $\mathcal{D}$'s state. Thus, $\mathcal{D}$'s state in a point in $C(r,m)$ defines a $\mathcal{D}$-property to be included into the local possibility policy in run r at time m. Formally, we translate an $\mathcal{A}$-allowability function to a $\mathcal{D}$-possibility policy as follows:

$$\texttt{policy}_C(r,m) = \{\mathcal{K}_{\mathcal{D}}(r',m') \mid (r',m') \in C(r,m)\} \tag{3.2}$$

First, $\texttt{policy}_C$ needs not be global unlike $\texttt{policy}_f$. Secondly, all properties of $\texttt{policy}_C$ are $\mathcal{D}$-information sets and, thus, are $\mathcal{D}$-properties defining a state of agent $\mathcal{D}$. Vice versa, if a $\mathcal{D}$-possibility policy just contains $\mathcal{D}$-information sets in each point (r,m) such that these sets cover the set $\mathcal{K}_{\mathcal{A}}(r,m)$[3], an $\mathcal{A}$-allowability function C can be defined by selecting at least one point from each $\mathcal{D}$-property in $\texttt{policy}(r,m)$. As a conclusion, C-secrecy corresponds to policy-based secrecy for the class of all policies where the properties are

[3]The covering is a requirement of the $\mathcal{A}$-allowability function C in [58].

$\mathcal{D}$-properties, namely $\mathcal{D}$-informations sets, and cover $\mathcal{K}_{\mathcal{A}}(r,m)$ in each point (r,m).

Proposition 3.2 (C-secrecy Is $\texttt{policy}_C$-based Secrecy).

Let C be an $\mathcal{A}$-allowability function and $\texttt{policy}_C$ *the resulting policy given by Equation* (3.2)*. Then agent $\mathcal{D}$ maintains C-secrecy with respect to agent $\mathcal{A}$ in $\mathcal{R}$ iff $\mathcal{D}$ maintains* $\texttt{policy}_C$*-based secrecy with respect to $\mathcal{A}$ in $\mathcal{R}$.*

The proof is immediate from the definitions and Equation 3.2.
A complete overview over the relationship between Halpern and O'Neill's definitions of secrecy and policy-based secrecy is in Figure 3.1 on page 37. In the next section, we will elaborate on the relationship between policy-based secrecy and run-based secrecy, the last one among the three possibilistic definitions of secrecy by Halpern and O'Neill.

3.4 $\mathcal{R}$-Possibility Policies and Run-based Secrecy

The semantic objects an agent might reason about in the Runs & Systems model are runs, global states and local states. In the previous section, we viewed the attacker $\mathcal{A}$ as a reasoner about global states and, as a special case, about the defender $\mathcal{D}$'s local state. But we might also view the attacker as a reasoner about runs with the operator $\mathcal{R} \circ \mathcal{K}_{\mathcal{A}}$ introduced in Section 2.3. Reasoning about runs resembles reasoning about traces in the trace-based model of the observer's (here: $\mathcal{A}$'s) information gain surveyed in Section 2.2. Following the ideas of that section, it might also be convenient for the security engineer to express security requirements as properties of runs, not as properties of states. However, in Runs & Systems, that way of specifying security requirements is less expressive than possibility policies and policy-based secrecy as we show in this section.
First, will illustrate the main concepts and results of this section with an example which recapitulates run-based secrecy of Section 2.3 in the context of policy-based secrecy. Run-based secrecy considers the attacker $\mathcal{A}$ as a reasoner with operator $\mathcal{R} \circ \mathcal{K}_{\mathcal{A}}$.

Example 3.4 (Policy-Based and Run-Based Secrecy).
Cf. Example A.2 of [58]

Consider the system

$$\begin{aligned}\mathcal{R} := \{r_1 &= \langle (A,X),(A,X),\ldots\rangle,\\ r_2 &= \langle (B,X),(A,Y),\ldots\rangle,\\ r_3 &= \langle (A,Y),(A,Y),\ldots\rangle\}\end{aligned}$$

where $\{X,Y\}$ are the states of agent $\mathcal{A}$ and $\{A,B\}$ those of agent $\mathcal{D}$. Run-based secrecy in this scenario amounts to two requirements:

1. *The first requirement concerns state B of agent $\mathcal{D}$, which is described by the $\mathcal{D}$-property $\mathcal{K}_{\mathcal{D}}(r_2,0)$. It says that for all points $(r,m) \in \mathcal{PT}(\mathcal{R})$ there exists a possible run $r' \in \mathcal{R}(\mathcal{K}_{\mathcal{A}}(r,m))$ (from agent $\mathcal{A}$'s point of view) such that $r' \in \mathcal{R}(\mathcal{K}_{\mathcal{D}}(r_2,0)) = \{r_2\}$. Thus, r_2 must be a possible run.*
   Recall that agent $\mathcal{A}$ reasons about possible runs with operator $\mathcal{R} \circ \mathcal{K}_{\mathcal{A}}$ from its available runtime information in run r at time m represented in its local state $r_{\mathcal{A}}(m)$.[4] *With that information only, a run r' is possible iff agent $\mathcal{A}$ might reach state $r_{\mathcal{A}}(m)$ in that run. This is equivalent to the fact that there exists $n \in \mathbb{N}_0$ such that $(r',n) \in \mathcal{K}_{\mathcal{A}}(r,m)$.*
   Altogether, we may rewrite the first requirement in our example, saying that r_2 must be a possible run, to the following equivalent requirement. For all $(r,m) \in \mathcal{PT}(\mathcal{R})$ it holds $\mathcal{K}_{\mathcal{A}}(r,m) \cap \{(r_2,n) \mid n \in \mathbb{N}_0\} \neq \emptyset$. This is policy-based secrecy for the property $I = \{(r_2,n) \mid n \in \mathbb{N}_0\}$.

2. *Considering state A of agent $\mathcal{D}$, which is described by the $\mathcal{D}$-property $\mathcal{K}_{\mathcal{D}}(r_1,0)$, run-based secrecy requires that there exists a possible run $r' \in \mathcal{R}(\mathcal{K}_{\mathcal{A}}(r,m))$ such that $r' \in \mathcal{R}(\mathcal{K}_{\mathcal{D}}(r_1,0)) = \{r_1,r_2,r_3\} = \mathcal{R}$. Clearly, this requirement is dispensable in the considered system.*

First, for the system $\mathcal{R}$ of this example the requirements of run-based secrecy can be equivalently expressed by the possibility policy `policy`$(r,m) = \{I\}$ *and*

[4] As remarked by Halpern and O'Neill, reasoning from runtime information is different from the trace-based model of $\mathcal{A}$'s reasoning where $\mathcal{A}$ may observe an *infinite* sequence of events, cf. Section 3.5 for further discussion.

policy-based secrecy. Second, note that the property I is not a $\mathcal{D}$-property. Consequently, run-based secrecy in this example cannot be enforced via $\mathcal{D}$-possibility policies.

In the following, we will first generalize run-based secrecy to policy-based run-based secrecy by allowing the declaration of an *$\mathcal{R}$-possibility policy* in place of the fixed policy of run-based secrecy. Then, we show that the generalization still can be expressed by policy-based secrecy as suggested by the previous example.
Run-based secrecy of [58] just considers the property of a run that agent $\mathcal{D}$ might reach in some particular state in that run, cf. Section 2.3. Regarding the agent's state $r_{\mathcal{D}}(m)$ in some point (r, m), the property is specified as the set $\mathcal{R}(\mathcal{K}_{\mathcal{D}}(r, m))$ of runs. In the declaration of an $\mathcal{R}$-possibility policy, general properties of runs may be used which are simply defined as sets $R \subseteq \mathcal{R}$.

Definition 3.4 (Syntax of $\mathcal{R}$-Possibility Policy)**.**

A $\mathcal{R}$-possibility policy is a function $\texttt{policy}\mathcal{R} : \mathcal{PT}(\mathcal{R}) \to \mathcal{P}(\mathcal{P}(\mathcal{R}))$*. Both the sets* $\texttt{policy}\mathcal{R}(r, m)$ *and their element sets* $R \in \texttt{policy}\mathcal{R}(r, m)$ *may be infinite.*

Policy-based run-based secrecy generalizes run-based secrecy in the following way. In any run r at any time m, agent $\mathcal{A}$ should not rule out that the current run has property R if R is declared in the local $\mathcal{R}$-possibility policy $\texttt{policy}\mathcal{R}(r, m)$.

Definition 3.5 (Semantics of of $\mathcal{R}$-Possibility Policy: Policy-Based Run-Based Secrecy)**.**

Let $\texttt{policy}\mathcal{R}$ *be an $\mathcal{R}$-possibility policy. Then agent $\mathcal{D}$ maintains policy-based run-based secrecy with respect to agent $\mathcal{A}$ in $\mathcal{R}$ iff, for all points $(r, m) \in \mathcal{PT}(\mathcal{R})$ and for all* $R \in \texttt{policy}\mathcal{R}(r, m)$*:*

$$\mathcal{R}(\mathcal{K}_{\mathcal{A}}(r, m)) \cap R \neq \emptyset.$$

If we define $\texttt{policy}\mathcal{R}(r, m) = \{\mathcal{R}(\mathcal{K}_{\mathcal{D}}(r', m')) \mid (r', m') \in \mathcal{PT}(\mathcal{R})\}$ for every point (r, m), then we have run-based secrecy.
Almost immediately from the definitions we obtain the following relation between policy-based run-based secrecy and policy-based secrecy.

Proposition 3.3 (Policy-Based Run-Based Secrecy by Policy-Based Secrecy).

Let $\mathcal{R}$ be a system and $\texttt{policy}\mathcal{R}$ *an $\mathcal{R}$-possibility policy defined over $\mathcal{R}$. Based on this policy, we define a possibility policy* $\texttt{policy}$ *of Def. 3.2 by*

$$\texttt{policy}(r,m) = \{\mathcal{PT}(R) \mid R \in \texttt{policy}\mathcal{R}(r,m)\}.$$

Then, agent $\mathcal{D}$ maintains $\texttt{policy}$*-based secrecy towards agent $\mathcal{A}$ iff agent $\mathcal{D}$ maintains* $\texttt{policy}\mathcal{R}$*-based run-based secrecy towards agent $\mathcal{A}$.*

This result is due to the fact that the operator $\mathcal{R} \circ \mathcal{K}_{\mathcal{A}}$ models reasoning from the runtime information $r_{\mathcal{A}}(m)$, not from observing the entire run r. A similar result has been given by Halpern and O'Neill in Proposition 3.10 of [58]: For synchronous multiagent systems, where additionally agents do not forget any runtime information during execution, run-based secrecy is equivalent to C-secrecy with an appropriate allowability function C.
Conversely, we may in general not define an equivalent $\mathcal{R}$-possibility policy from a possibility policy as illustrated in the following example.

Example 3.5 (Expressiveness of Policy-based Run-based Secrecy).

Consider the following simple system taken from the scenario of Example 3.4.

$$\mathcal{R} := \{r = \langle (B,X),(A,Y),(A,Y),\ldots\rangle\}.$$

Further, consider the $\mathcal{D}$-possibility policy $\texttt{policy}\mathcal{D}(r,m) = \{\mathcal{K}_{\mathcal{D}}(r,0)\}$ *for all $m \in \mathbb{N}_0$. Policy-based secrecy is violated for this policy at time 1, but preserved at time 0. Modeling the attacker $\mathcal{A}$ as a reasoner with operator $\mathcal{R} \circ \mathcal{K}_{\mathcal{A}}$, the attacker does not gain any information throughout the execution (because $\mathcal{R} \circ \mathcal{K}_{\mathcal{A}}(r',m') = \{r\}$ for all points (r',m') of the system). Hence, policy-based secrecy cannot be stated as a requirement of confining the attacker's information gain in that model (used by policy-based run-based secrecy).*

Thus, possibility policies and policy-based secrecy are more expressive than $\mathcal{R}$-possibility policies and policy-based run-based secrecy. Finally, we remark that the possibility policy of Proposition 3.3 need not be a $\mathcal{D}$-possibility policy, cf. Example 3.4.

Altogether, possibility policies are a very general means to declare confidentiality aims since the declaration may refer to arbitrary properties of global states, cf. Figure 3.1 on the expressivity of policy-based secrecy. In Chapter 4, we will see that this generality is needed to express an epistemic agent $\mathcal{D}$'s confidentiality aims when sharing information. In particular, $\mathcal{D}$-properties do not suffice.

3.5 Relation to Other Approaches

In this section, we mainly compare policy-based secrecy with the "modular assembly kit for security properties" (MAKS) in Section 2.2. In the preliminary work [28], the reader may find a comparison between policy-based secrecy and function views proposed by Hughes and Shmatikov in [64]. Moreover, in our preliminary work [28], we present a syntactic characterization of policy-based secrecy in the logic of knowledge and time of Section 2.5 following Halpern and O'Neill's approach in [58].

For the comparison with MAKS, we recapitulate the trace-based model of $\mathcal{A}$'s information gain which is used in MAKS. We will do so comparing this model to the Runs & Systems model $\mathcal{R}$ of a multiagent system with two agents, the defender $\mathcal{D}$ and the attacker $\mathcal{A}$. The trace-based model defines a system Sys as a set of traces which are sequences of events. Within Runs & Systems, the events are of the form $s_{\mathcal{X}}$ saying that agent $\mathcal{X}$ enters the local state $s_{\mathcal{X}}$. Thus, we translate a run $r = \langle (s^0_{\mathcal{D}}, s^0_{\mathcal{A}}), (s^1_{\mathcal{D}}, s^1_{\mathcal{A}}), \dots \rangle$ to a trace $t_r = \langle s^0_{\mathcal{D}}, s^0_{\mathcal{A}}, s^1_{\mathcal{D}}, s^1_{\mathcal{A}}, \dots \rangle$. This way, we define the system $Sys_{\mathcal{R}}$ of traces from the system $\mathcal{R}$ of runs.[5] The observer, agent $\mathcal{A}$, may observe every event of the form $s_{\mathcal{A}}$. These events define the set V of visible events of $\mathcal{A}$'s view $\mathcal{V} = (V, N, C)$ in MAKS. Further, as an example we take the set C of confidential events as the set of all events where agent $\mathcal{D}$ enters a local state with some $\mathcal{D}$-property I.[6]

[5] Differently, Halpern and O'Neill in [58, Section 5] translate a system of traces to a system of runs and show an example of a security property in the trace-based model that may be expressed by total secrecy in that translation. We use the converse translation because it is simpler and suffices to illustrate the differences and commonalities of the two models of information gain and its confinement.

[6] Recall that property I may not be confidential itself, but relevant for the protection of

As a first aspect, we compare the model of agent $\mathcal{A}$'s information gain in the two formalisms. In MAKS, agent $\mathcal{A}$ might gain information by observing some trace t as the sequence $t\,|_V$ of visible events in that trace. Formally, the agent cannot distinguish all traces in the set $K(Sys_{\mathcal{R}}, V, t)$ of (2.1) from t. For comparing MAKS with policy-based secrecy, certainly, we need an equal understanding of $\mathcal{A}$'s information gain. Formally, we require that for all runs $r \in \mathcal{R}$, which each define a trace t_r, for every time $m \in \mathbb{N}_0$, for all runs $r' \in \mathcal{R}$ it holds that

$$t_{r'} \in K(Sys_{\mathcal{R}}, V, t_r) \qquad \text{iff} \qquad r' \in \mathcal{R}(\mathcal{K}_{\mathcal{A}}(r, m)). \tag{3.3}$$

In words, after observing trace $t_r\,|_V$, trace $t_{r'}$ translated from run r' is a possible trace iff r' is a possible run at point (r, m). First, necessarily, the equivalence may only hold at time m if agent $\mathcal{A}$ has acquired all the information at that time it would obtain when observing the entire run r. Second, in system $\mathcal{R}$, agent $\mathcal{A}$ might have forgotten prior observations at time m in which case the equivalence would neither hold. These differences have already been remarked by Halpern and O'Neill in [58]. They point out that the operator $\mathcal{K}_{\mathcal{A}}$ corresponds to reasoning about *prefixes of traces*, at least, if agent $\mathcal{A}$ does not forget (that is has *perfect recall*). In particular, in [58, Section 5.1] the reader may find an example where the slight difference in the model of information gain matters.

As another aspect, we compare the expressivity of the two formalisms to specify confinements of agent $\mathcal{A}$'s information gain. To this end, we assume that (3.3) holds for run $r \in \mathcal{R}$ at time m. In MAKS, confinement of $\mathcal{A}$'s information gain is declared by a security predicate, that is a boolean function $\mathtt{SP}_{\mathcal{V}}$ on sets of traces depending on $\mathcal{A}$'s view $\mathcal{V}$. Modeling the attacker $\mathcal{A}$ as a reasoner about runs by (3.3), we confine the agent's information gain by an $\mathcal{R}$-possibility policy. In Section 3.4, we argued that these policies are less expressive than possibility-policies.

In the following, we compare security predicates and $\mathcal{R}$-possibility policies. A security predicate is a closure property of a set of traces, here the system $Sys_{\mathcal{R}}$, which means that it requires the existence of traces of a specific form in the system. Likewise, an $\mathcal{R}$-possibility policy with policy-based run-based

confidential properties, cf. Example 3.2.

secrecy semantics requires the existence of particular runs in the system. To recall this requirement, for any property R declared in the local policy in point (r, m) there must exist some run r' with $r' \in \mathcal{R}(\mathcal{K}_{\mathcal{A}}(r, m)) \cap R$. Thus, intuitively, $\mathcal{R}$-possibility policies and policy-based run-based secrecy should essentially be as expressive as security predicates and views in MAKS if we assume the equivalence in (3.3). For example, in the same way that non-interference has been equivalently specified in MAKS, cf. Section 2.2, we may express noninterference by an appropriate $\mathcal{R}$-possibility policy. To this end, we consider that events where $\mathcal{D}$ entered a state with property I should not interfere with events where $\mathcal{A}$ entered any state. This is declared in the following policy

$$\texttt{policy}\mathcal{R}(r, m) = \{\{r' \in \mathcal{R} \mid \text{not exists } n \in \mathbb{N}_0 \text{ such that } (r', n) \in I\}\}.$$

The reader may compare this policy with policy-based run-based secrecy semantics to the basic security predicate of (2.2). A formal comparison is beyond the scope of this thesis.

With the above considerations, we may suppose that declaring confinement of information gain in MAKS is less expressive than declaring possibility policies. Possibility policies refer to properties of states and, thus, allow to declare a finer-grained policy.

Chapter 4

Epistemic Agents in Runs & Systems

In this chapter, we outline and model a scenario of two interacting epistemic agents and based on this model formalize the confidentiality aims of one of the agents. In Section 1.2, we have already sketched an example of this scenario in the context of a supply chain application. In our focused scenario, an agent $\mathcal{D}$ reacts to the requests of another agent $\mathcal{A}$ while, in terms of security, the reacting agent $\mathcal{D}$ is seen as a defender of its confidentiality interests towards the requesting agent $\mathcal{A}$ as a potential attacker of these interests.
In the beginning of this chapter, we will generally specify the system with the two agents $\mathcal{D}$ and $\mathcal{A}$ as part of the security engineering task of the formal specification. We will complete the specification with a model of the multiagent system in Runs & Systems. With that model, we will relate the formal specification of confidentiality defined in this chapter to policy-based secrecy and this way also to other confidentiality properties in the literature. Our intention is to give an abstract specification of the interaction interface and the functionality of the reacting agent $\mathcal{D}$ as a defender as a part of the security engineering task, but in this chapter we will not give examples of types of interactions or implementations of $\mathcal{D}$'s functionality. Instead we will treat these aspects in Part II in depth.

The organization of this chapter is as follows. Section 4.1 provides an overview of the multiagent system from the perspective of the security engineer. Then, the remaining sections include the activities of the first part of the security

engineering task, the formal specification, cf. the introduction to Part I on page 17. In the first activity, the functionality of the reacting agent $\mathcal{D}$ to be constructed is specified in Section 4.2.
In the second activity, the means of the requesting agent $\mathcal{A}$ as an attacker against $\mathcal{D}$'s interests are postulated and modeled in Section 4.3.
In the third activity, the multiagent system as seen by the security engineer is specified in Runs & Systems in Section 4.4. Then, in Section 4.5, the semantics of the confidentiality policy is formalized. In Section 4.6, the semantics of the confidentiality policy is related to the framework of Section 3.

4.1 Two Interacting Epistemic Agents

The focused scenario is visualized in Figure 4.1. In this scenario, an agent $\mathcal{A}$ iteratively sends requests for offering or demanding information to another agent $\mathcal{D}$ who reacts to these requests in return. To this end, the two agents have agreed on propositional logic $\mathcal{L}_{PL}$ over a countably infinite alphabet $\mathcal{At}$ as a *base language* for their communication. The *interaction interface* provided by the reacting agent $\mathcal{D}$ to the requesting agent $\mathcal{A}$ is defined by pairs of valid request messages from the set $\mathcal{Req}$ and respective valid reaction messages from the set $\mathcal{Rea}$. Formally, we may define an interaction interface as a subset of $\mathcal{Req} \times \mathcal{Rea}$, but will never use this explicit way of defining the interface in the following. Recall that at this place we will not give specific types of requests and reactions, but leave this to Part II.
Both agents are epistemic: generally, an epistemic agent should be capable to reason about its environment and itself and the other agent from its available information. Here, we focus on specific epistemic behavior of the two agents as seen by the security engineer which we explain in the following.
First, the reacting agent $\mathcal{D}$ has the usual functionality of an interacting, intelligent and autonomous agent. This functionality includes and bases on $\mathcal{D}$'s reasoning by means of which $\mathcal{D}$ forms its belief about its environment. More precisely, in the process of reasoning, agent $\mathcal{D}$ applies a *belief operator* $\mathsf{Bel}_{\mathcal{D}}$ on its *epistemic state* comprising its *background knowledge* $bknowl$ and its *evidential knowledge* $eknowl$. The evidential knowledge is specific information $\mathcal{D}$ gathered about the case it reasons about. In contrast, its background

knowledge represents its general expertise used in its process of reasoning. The resulting belief is represented by the set $\mathsf{Bel}_{\mathcal{D}}(bknowl, eknowl) \subseteq \mathcal{L}_{PL}$ using the common base language $\mathcal{L}_{PL}$. Due to obligations or own interests, agent $\mathcal{D}$ aims at keeping certain parts of its belief confidential and specifies these confidential parts by a *confidentiality policy* $conf$. The security engineer sees $\mathcal{D}$ as *defending* its confidentiality aims towards the requesting agent $\mathcal{A}$ and knows and considers $\mathcal{D}$'s functionality, here mainly its belief component, when designing agent $\mathcal{D}$ to enforce these aims.
Second, the security engineer sees the requesting agent $\mathcal{A}$ as attacking $\mathcal{D}$'s confidentiality aims regarding $\mathcal{D}$'s belief and, in this respect, postulates agent $\mathcal{A}$ to be a *skeptical reasoner* about $\mathcal{D}$'s belief. Skeptical reasoning means that $\mathcal{A}$ aims to become certain about $\mathcal{D}$'s belief in the sense that any conclusion about $\mathcal{D}$'s belief should indeed be true. This reasoning is in spirit of the reasoning operators $\mathcal{K}$ and $\mathcal{R} \circ \mathcal{K}$ used by Halpern and O'Neill [58] to model the attacker's reasoning because with those operators the attacker considers the defender $\mathcal{D}$'s state in *all* possible states of the multiagent system, respectively in *all* possible runs of the multiagent system, cf. Section 2.3. In the artificial intelligence community, the term "skeptical reasoning" is used similarly, but in the context of nonmonotonic (defeasible) reasoning [6]. Agent $\mathcal{A}$'s means of reasoning are postulated more specifically as follows. From its observations, agent $\mathcal{A}$ (is assumed to) try to approximate background and evidential knowledge of $\mathcal{D}$'s epistemic state exploiting its awareness of $\mathcal{D}$'s functionality. The observations are represented as follows.

Definition 4.1 (History $hist$ of an Interaction)**.**

A history $hist$ of an interaction is a finite sequence of pairs of request $\theta \in \mathcal{Req}$ and respective reaction $react \in \mathcal{Rea}$.

Its approximations and the history are the basis for agent $\mathcal{A}$'s skeptical reasoning about $\mathcal{D}$'s current or previous belief.
Third, to enforce its confidentiality aims, the defending agent $\mathcal{D}$ *simulates* the attacking agent $\mathcal{A}$'s postulated reasoning about $\mathcal{D}$'s belief. As a basis for this simulation, agent $\mathcal{D}$ maintains an *attacker view* on agent $\mathcal{A}$'s approximations and tracks $\mathcal{A}$'s observations in the history of an interaction. Agent $\mathcal{D}$ might have an *a priori view* $view_0$ on agent $\mathcal{A}$'s initial approximations.

Example 4.1 (A Priori View)**.**

In the application scenario of Section 1.2, the supplier $\mathcal{A}$ may have some expertise in the design of the manufacturer $\mathcal{D}$'s product and, moreover, may know that the manufacturer $\mathcal{D}$ shares this expertise while, in turn, $\mathcal{D}$ may be aware about that knowledge. This awareness is represented by $\mathcal{D}$'s view on $\mathcal{A}$'s initial approximations of $\mathcal{D}$'s background knowledge.

Finally, we terminate the agents' process of mutually reasoning about one another with the following assumption.

Assumption 1 ($\mathcal{D}$'s Complete Knowledge of $\mathcal{A}$'s Approximations)**.**

Agent $\mathcal{D}$'s view and agent $\mathcal{A}$'s approximations coincide.

This assumption is the best case from the defender $\mathcal{D}$'s perspective: $\mathcal{D}$ is completely aware of what $\mathcal{A}$ knows about $\mathcal{D}$'s epistemic state. Surely, this assumption makes it necessary to carefully specify $\mathcal{D}$'s initial view in practice. We will discuss the assumption in Section 5.2.

4.2 Confidentiality Policy and its Enforcement by Defender $\mathcal{D}$

4.2.1 Policy Declaration

Under the confidentiality aspect, agent $\mathcal{D}$ declares a fixed *confidentiality policy conf* that expresses its confidentiality interests or obligations towards agent $\mathcal{A}$. This policy *explicitly* declares confidential pieces of information in contrast to the possibility policies of Chapter 3 which may only implicitly define the confidentiality of properties of the system, cf. Example 3.2. The declaration of the confidentiality policy comprises two *disjoint finite* sets $conf(TCP)$ and $conf(CCP)$ of propositional formulas whose semantics is roughly explained as follows.
On the one hand, some parts of $\mathcal{D}$'s belief may only be sensitive and confidential at the time agent $\mathcal{D}$ holds that belief, but may be disclosed after $\mathcal{D}$ has given up that belief. An example is the manufacturer's result of the

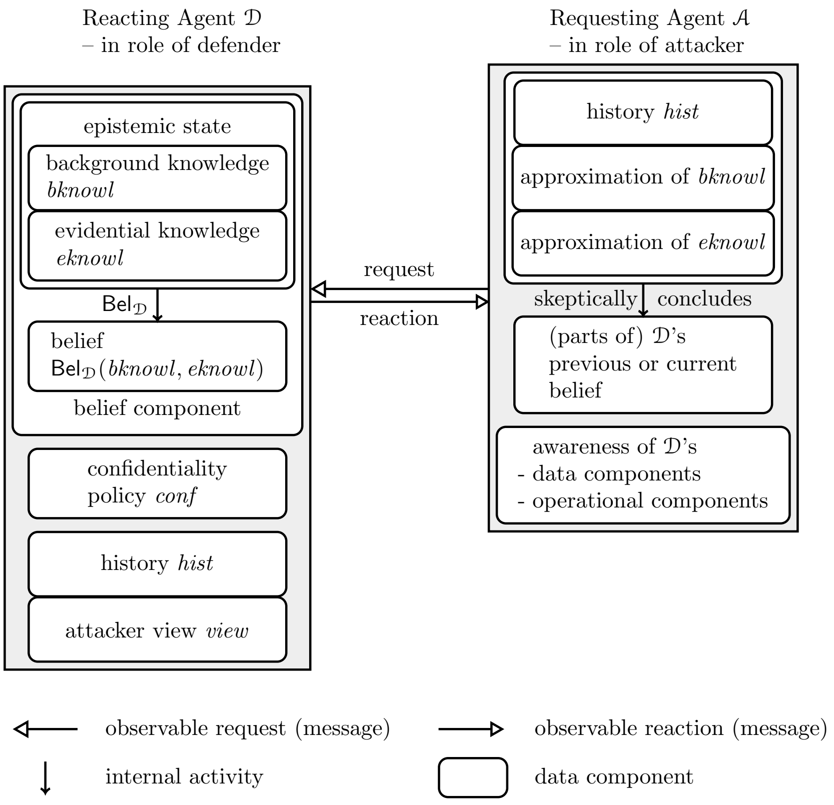

Reacting Agent $\mathcal{D}$
– in role of defender
epistemic state
background knowledge *bknowl*
evidential knowledge *eknowl*
$\mathsf{Bel}_\mathcal{D}$
belief $\mathsf{Bel}_\mathcal{D}(bknowl, eknowl)$
belief component
confidentiality policy *conf*
history *hist*
attacker view *view*
request
reaction
Requesting Agent $\mathcal{A}$
– in role of attacker
history *hist*
approximation of *bknowl*
approximation of *eknowl*
skeptically concludes
(parts of) $\mathcal{D}$'s previous or current belief
awareness of $\mathcal{D}$'s
- data components
- operational components
observable request (message)
observable reaction (message)
internal activity
data component

Figure 4.1: The multiagent system specified by the security engineer: the agent $\mathcal{D}$ whose interface and functionality have to specified by the security engineer and the agent $\mathcal{A}$ whose reasoning capabilities have to be postulated by the security engineer

feasibility study of a product development proposal in the application scenario of Section 1.2. Such confidentiality interests are expressed by the part $conf(TCP) \subset_{fin} \mathcal{L}_{PL}$ of the confidentiality policy, the set of *temporary confidential belief*. Roughly, this part of the policy has the following semantics: If $\mathcal{D}$ *currently believes* formula B with $B \in conf(TCP)$, then agent $\mathcal{D}$ aims to hide this fact from agent $\mathcal{A}$. Note that each formula $B \in conf(TCP)$ is called a temporary confidential belief regardless of whether or not agent $\mathcal{D}$ actually currently believes B.

Example 4.2 (Confidentiality Policy).

In our example from Section 1.2, in a feasibility study agent $\mathcal{D}$ reasons about whether the proposal for development of a particular product is realizable. The *atom r represents the fact that the product proposal is realizable. The manufacturer's confidentiality interest to hide a current positive result of the feasibility study from supplier agent $\mathcal{A}$ is expressed by the confidentiality policy $conf(TCP) = \{r\}$. More precisely, with this policy agent $\mathcal{D}$ aims to conceal the fact $r \in \mathsf{Bel}_{\mathcal{D}}(bknowl, eknowl)$ from agent $\mathcal{A}$ whereas the fact $r \notin \mathsf{Bel}_{\mathcal{D}}(bknowl, eknowl)$ and the fact $\neg r \in \mathsf{Bel}_{\mathcal{D}}(bknowl, eknowl)$ may be revealed.*

On the other hand, some parts of $\mathcal{D}$'s belief might be sensitive even at the time when $\mathcal{D}$ has already given up that belief such as the physician's diagnosis of cancer. Such confidentiality interests are expressed by the part $conf(CCP) \subset_{fin} \mathcal{L}_{PL}$ of the confidentiality policy, the set of *continuous confidential belief*, disjoint with the other part $conf(TCP)$. In this case, if $\mathcal{D}$ *has believed or currently believes* formula B with $B \in conf(CCP)$, then agent $\mathcal{D}$ aims to hide this fact from agent $\mathcal{A}$. Note that each formula $B \in conf(CCP)$ is called a continuous confidential belief regardless of whether or not agent $\mathcal{D}$ has believed or currently believes B. Each formula in the set $conf(TCP) \cup conf(CCP)$ is called a *confidential belief*.
First, we remark that in the policy *conf* agent $\mathcal{D}$ cannot express a purpose of hiding that it *does not believe* something, for example, hiding the fact that it does not have a result of the feasibility study yet (that is the fact that neither $r \in \mathsf{Bel}_{\mathcal{D}}(bknowl, eknowl)$ nor $\neg r \in \mathsf{Bel}_{\mathcal{D}}(bknowl, eknowl)$ holds).
Further, note that we assume agent $\mathcal{D}$ to aim at hiding its belief only. For example, although the manufacturer agent $\mathcal{D}$ might desire to hide the *fact* that the product proposal is realizable, $\mathcal{D}$ only aims at hiding its own belief in the realizability of the proposal. We argue that it is the minimum responsibility of an agent to ensure the confidentiality of pieces of information it has already acquired, thus, of its own belief.

4.2.2 Outline of a Confidentiality Preserving Agent $\mathcal{D}$

The main objective of security engineering in this thesis is constructing agent $\mathcal{D}$ to effectively enforce its confidentiality interests expressed by policy *conf* towards agent $\mathcal{A}$. The overall construction is outlined by Figure 4.2:
The *censor* with the *control function* cexec separates the interaction interface

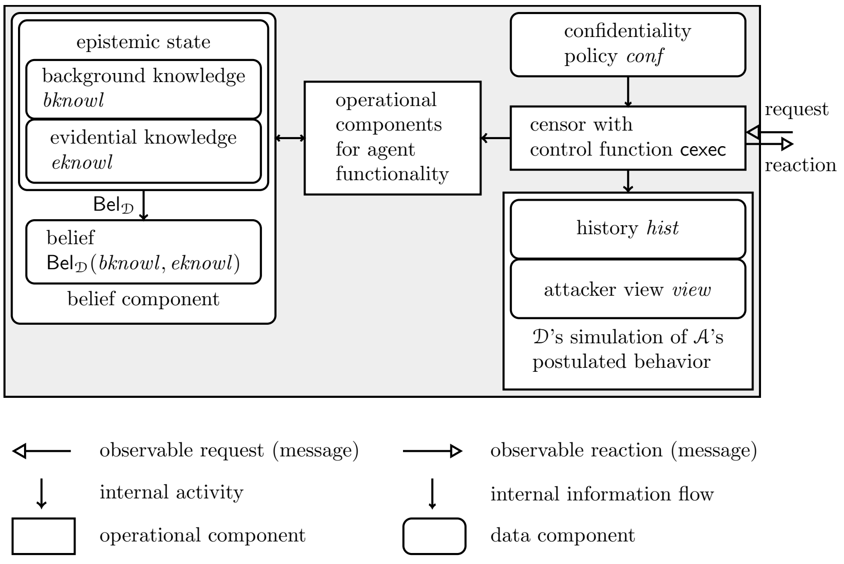

epistemic state
background knowledge *bknowl*
evidential knowledge *eknowl*
$\mathsf{Bel}_{\mathcal{D}}$
belief $\mathsf{Bel}_{\mathcal{D}}(bknowl, eknowl)$
belief component
operational components for agent functionality
confidentiality policy *conf*
censor with control function cexec
request
reaction
history *hist*
attacker view *view*
$\mathcal{D}$'s simulation of $\mathcal{A}$'s postulated behavior
observable request (message)
observable reaction (message)
internal activity
internal information flow
operational component
data component

Figure 4.2: Design of a confidentiality preserving agent $\mathcal{D}$ as defender against a requesting agent $\mathcal{A}$ as attacker

provided to agent $\mathcal{A}$ from agent $\mathcal{D}$'s internal data and functionality. The essential task of the censor's control function is to check whether a planned reaction to a request would enable $\mathcal{A}$ to skeptically conclude a current or previous belief of agent $\mathcal{D}$ if this belief is declared confidential in the policy. To this end, the control function cexec *simulates* agent $\mathcal{A}$'s postulated reasoning about $\mathcal{D}$'s belief. The simulation is based on the attacker view on agent $\mathcal{A}$'s approximations and the history of the interaction with $\mathcal{A}$'s observations.
As part of this simulation, the control function incorporates what $\mathcal{A}$ has learned from outgoing reaction messages into the attacker view and tracks request and respective reaction in the history. Note that the simulation of agent $\mathcal{A}$ is separate from $\mathcal{D}$'s reasoning and resulting belief about its environment.
Security engineering involves postulating the attacking agent $\mathcal{A}$'s awareness about $\mathcal{D}$'s data and operational components. To this end, we first define $\mathcal{D}$'s data components via an *abstract state* and then define its operational components via the control function cexec. These definitions provide a specification for the implementation of agent $\mathcal{D}$ in Part II.

Definition 4.2 (Abstract State of the Defender $\mathcal{D}$).

The set $\mathcal{St}$ of agent $\mathcal{D}$'s abstract local states consists of tuples with the following components:

1.	*bknowl:*	*background knowledge,*
2.	*eknowl:*	*evidential (or contextual) knowledge,*
3.	$\mathsf{Bel}_{\mathcal{D}}$*:*	*belief operator,*
4.	*conf:*	*confidentiality policy,*
5.	*hist:*	*history of the interaction,*
6.	*view:*	*attacker view.*

Generally, agent $\mathcal{D}$ has several operational components within its usual BDI-agent functionality (shown as a black box in Figure 4.2). However, since the censor is between the interaction interface to agent $\mathcal{A}$ and $\mathcal{D}$'s other operational components, agent $\mathcal{A}$ may well view $\mathcal{D}$'s operational components as a single black box the inputs and outputs of which agent $\mathcal{A}$ may only observe through inputs and outputs of the censor.[1] For conciseness of the presentation, we simplified the processing of the censor as follows. At an incoming request, the censor calls its control function cexec. Beside other computations, the control function writes the reaction to the request into $\mathcal{D}$'s history component. After the control function terminates, the censor reads the respective reaction and sends it to the requesting agent $\mathcal{A}$.
The interface of the censor's control function is defined as follows.

Definition 4.3 (Control Function cexec).

A control function $\mathsf{cexec} : \mathcal{St} \times \mathcal{Req} \to \mathcal{St}$ *takes as input an abstract state of $\mathcal{D}$ and a request. Its output is a subsequent abstract state of $\mathcal{D}$. The function always tracks request and respective reaction in the history component of the abstract state, formally,*

$$\mathsf{cexec}((\ldots, hist, \ldots), \theta) = (\ldots, hist \centerdot (\theta, react), \ldots) \text{ with } \theta \in \mathcal{Req}, react \in \mathcal{Rea}.$$

Here, the operator $\centerdot$ denotes the concatenation of sequences.

[1] If agent $\mathcal{D}$ was designed with a BDI architecture, the control function might not be between the communication interface and the agent functionality. We will discuss this matter in Section 5.2.

Note that the set $\mathcal{St}$ implicitly defines invariants of the control function.
We illustrate the abstract specification of agent $\mathcal{D}$ with a simple implementation.

Example 4.3 (Database Queries).

We take up the setting from Example 2.1. There, an reacting agent $\mathcal{D}$ has a local database $\mathcal{db}$ which may store items (propositions) a and b and responds to queries of another agent $\mathcal{A}$. In this setting, the state of agent $\mathcal{D}$ of Definition 4.2 is very simple: here we consider that $\mathcal{D}$'s evidential knowledge is the database instance without any background knowledge. Based thereupon, $\mathcal{D}$'s belief are just all propositional formulas $F \in \mathcal{L}_{PL}$ over the alphabet $\mathcal{At} = \{a, b\}$ that are true in the database (in the usual sense) denoted by $\mathcal{db} \models_{PL} F$. In the scope of our example, queries are expressed in $\mathcal{L}_{PL}$ as well as the confidentiality policy $conf(TCP) \subset_{fin} \mathcal{L}_{PL}$ (we set $conf(CCP) = \emptyset$) and the attacker's view $view \subset_{fin} \mathcal{L}_{PL}$ in which query answers should be tracked. Agent $\mathcal{D}$'s control function computes the simulation of $\mathcal{A}$'s skeptical reasoning about whether $\mathcal{D}$ currently believes formula $B \in \mathcal{L}_{PL}$ by deciding propositional entailment $view \vdash_{pl} B$.
Agent $\mathcal{D}$ implemented its control function as the following modified version of the refusal censor from [19] which on our purpose opens opportunities for agent $\mathcal{A}$ to infer confidential belief.

- *Request: query A*
- *If $\mathcal{db} \models_{PL} A$ and $view \cup \{A\} \not\vdash_{pl} S$ for all $S \in conf(TCP)$, then append (query A, A) to hist and set $view := view \cup \{A\}$.*
- *If $\mathcal{db} \models_{PL} \neg A$ and $view \cup \{\neg A\} \not\vdash_{pl} S$ for all $S \in conf(TCP)$, then append (query A, $\neg A$) to hist and set $view := view \cup \{\neg A\}$.*
- *Otherwise, append (query A, refuse) to hist.*

4.3 Postulates about the Requesting Agent $\mathcal{A}$ as Attacker

As a part of the security engineering task, we need to be more precise about $\mathcal{D}$'s aim of concealing belief from $\mathcal{A}$, that is, the formal semantics of the confidentiality policy *conf*. In order to do that, we first must postulate what $\mathcal{A}$'s means are to reason about $\mathcal{D}$ and $\mathcal{D}$'s belief, especially, to what extent $\mathcal{A}$ is aware of $\mathcal{D}$'s internal functionality, and what are $\mathcal{A}$'s computational capabilities.
The following *open design assumption* is essential for understanding agent $\mathcal{A}$'s means to reason about agent $\mathcal{D}$'s belief. This assumption follows the common rationale in computer security of "no security by obscurity". That means that confidentiality should not rely on $\mathcal{A}$'s ignorance of $\mathcal{D}$'s design.

Assumption 2 (Open Design for Attacker).
Cf. [15] Section 1.5.2

Agent $\mathcal{A}$ is aware of agent $\mathcal{D}$'s design.
More precisely, first, regarding the data components of agent $\mathcal{D}$ subsumed in its local state (Def. 4.2), agent $\mathcal{A}$ may read the components conf, hist and view any time during the interaction. Moreover, agent $\mathcal{A}$ is aware that $\mathcal{D}$'s initial state is within the set $\mathcal{St}_{Init} \subseteq \mathcal{St}$.
Second, regarding agent $\mathcal{D}$'s operational components (Def. 4.3), $\mathcal{A}$ knows the definition of the control function cexec *in the following sense. First, it knows the interface* $\mathsf{cexec} : \mathcal{St} \times \mathcal{Req} \to \mathcal{St}$ *of the function. Second, for each input $s \in \mathcal{St}$ and $\theta \in \mathcal{Req}$, agent $\mathcal{A}$ is able to determine* $\mathsf{cexec}(s, \theta)$.

Essentially, the effect of an open definition of the control function is that agent $\mathcal{A}$ could simulate $\mathcal{D}$'s processing of an interaction request if it further had access to $\mathcal{D}$'s background knowledge *bknowl* and $\mathcal{D}$'s evidential knowledge *eknowl*. Although these two components are *not* accessible to agent $\mathcal{A}$, in principle, $\mathcal{A}$ is able to test by way of simulation whether some instantiation of *bknowl* and some instantiation of *eknowl* together with the known policy *conf* and its observations in history *hist* and view *view* may produce an observed reaction to an interaction request. By observation and such testing, finally agent $\mathcal{A}$ might find out more about $\mathcal{D}$'s belief.

We assume an open policy *conf* as a matter of precaution, because agent $\mathcal{A}$ could guess or even know (parts of) the confidentiality policy (cf. [18, 19] for approaches that hide the policy). Likewise, assuming that the view is visible to $\mathcal{A}$ is also a matter of precaution.
In general, we define what agent $\mathcal{A}$ *a priori knows* about a specific implementation of agent $\mathcal{D}$ by defining the domain and the input-output behavior of the control function cexec. With the set $\mathcal{S}t_{Init}$, we may further define what agent $\mathcal{A}$ knows about the preconditions of an implemented control function. Moreover, with the set $\mathcal{S}t_{Init}$ we may specify what parts of particular instantiations of $\mathcal{D}$'s data components *bknowl* and *eknowl* agent $\mathcal{A}$ a priori knows. We will do the above formalization of $\mathcal{A}$'s awareness about $\mathcal{D}$'s implementation in Part II.
Finally, the security engineer has to make assumptions about the attacking agent's computational resources and capabilities which are no practical assumptions about $\mathcal{A}$'s actual computational resources, but also a matter of precaution in security engineering.
First, the attacker agent $\mathcal{A}$ is assumed not to forget about its previous states so that it is able to reconstruct the whole sequence of its previous states. Preparing for the next section, we precisely formulate this ability in Assumption 3 using the property of perfect recall of an agent in the Runs & Systems framework. In the next section, we will formalize the complete scenario of $\mathcal{D}$ and $\mathcal{A}$ outlined so far in this chapter in Runs & Systems.

Definition 4.4 (Perfect Recall).
Cf. Section 2 of [58]

Let $\mathcal{R}$ be a system and $\mathcal{X}$ an agent in $\mathcal{R}$. Then agent $\mathcal{X}$ has perfect recall iff for all points $(r, m) \in \mathcal{PT}(\mathcal{R})$, for all points $(r', m') \in \mathcal{K}_{\mathcal{X}}(r, m)$ agent $\mathcal{X}$ has the same local-state sequence at both (r, m) and (r', m'). A local-state sequence in a point (r, m) is the result of removing all successive duplicates from the sequence $\langle r_{\mathcal{X}}(0), \ldots, r_{\mathcal{X}}(m)\rangle$.

Assumption 3 (No Loss of Information).

Agent $\mathcal{A}$ has perfect recall.

As anticipated, we assume that $\mathcal{A}$ is a skeptical reasoner, that is, it aims to become certain about $\mathcal{D}$'s belief.

Assumption 4 (Attacker's Reasoning Capabilities).

Agent $\mathcal{A}$ is a skeptical reasoner who has unlimited computational power.

We will discuss the assumptions in Section 5.2.

4.4 Model of Defender $\mathcal{D}$ and Attacker $\mathcal{A}$ and Their Interaction in Runs & Systems

The main contribution of this section is a Runs & Systems model of our scenario of the two interacting epistemic agents $\mathcal{D}$ and $\mathcal{A}$ from Section 4.1. With this model, we will compare the semantics of $\mathcal{D}$'s confidentiality policy with policy-based secrecy of Chapter 3. From that section, we know how policy-based secrecy relates to several other confidentiality properties from the literature.

Essentially, our agent model in Runs & Systems is an abstract agent model from the security engineering perspective and, mainly, should capture all the postulated means of the attacking agent $\mathcal{A}$ for reasoning about $\mathcal{D}$'s confidential belief as outlined in Section 4.1 and further detailed in Section 4.3. Thus, importantly, the agent model does not represent the multiagent system as it is actually implemented; first, the attacker $\mathcal{A}$ is modeled as postulated by the security engineer and, further, the defender $\mathcal{D}$ is modeled as seen by the attacker in the way postulated by the security engineer. However, abstracting away details from the actual implementation of agent $\mathcal{D}$ in the model must be carefully reviewed under the principle of "no security by obscurity", cf. Assumption 2.

In this thesis, as introduced in Section 4.2, agent $\mathcal{D}$'s confidentiality interests concern just parts of its current or previous belief and hiding them from agent $\mathcal{A}$. Thus, from the security engineering point of view, the attacker $\mathcal{A}$ reasons only about $\mathcal{D}$'s local state and therefore only the information $\mathcal{A}$ has about $\mathcal{D}$'s local state is represented in $\mathcal{A}$'s local state in Runs & Systems. (Recall that in Runs & Systems an agent $\mathcal{X}$'s local state $r_{\mathcal{X}}(m)$ encodes the information accessible to the agent at runtime in run r at time m, cf. Section 2.3.)

The basis for the attacker $\mathcal{A}$'s reasoning about the defender $\mathcal{D}$'s local state is $\mathcal{D}$'s open design for the attacker in Assumption 2. By that assumption, agent $\mathcal{A}$ understands how agent $\mathcal{D}$ might modify its local state when processing agent $\mathcal{A}$'s request with a call to the control function cexec. Further, agent $\mathcal{A}$ may partially observe the actual modifications by receiving the respective reaction and by accessing parts of $\mathcal{D}$'s data components as permitted by Assumption 2. For these reasons, the security engineer models the system of defender $\mathcal{D}$ and attacker $\mathcal{A}$ on the basis of the openly defined control function cexec. Recall that the domain of this function already defines the set of $\mathcal{D}$'s local states (Definition 4.2) and the form of requests, cf. Definition 4.3.

Definition 4.5 (System $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of Defender $\mathcal{D}$ and Attacker $\mathcal{A}$)**.**

Let $\mathsf{cexec} : \mathcal{St} \times \mathcal{Req} \to \mathcal{St}$ *be a control function. Then, the global states of the system* $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ *have the form of pairs* $(s_\mathcal{D}, s_\mathcal{A})$ *where*

$$\begin{aligned} s_\mathcal{D} &= (\mathit{bknowl}, \mathit{eknowl}, \mathsf{Bel}_\mathcal{D}, \mathit{conf}, \mathit{hist}, \mathit{view}) \in \mathcal{St} \\ s_\mathcal{A} &= (\mathsf{Bel}_\mathcal{D}, \mathit{conf}, \mathit{hist}, \mathit{view}). \end{aligned}$$

Thus, $\mathcal{A}$'s abstract state $s_\mathcal{A}$ is defined as the projection of $\mathcal{D}$'s abstract state $s_\mathcal{D}$ of Definition 4.2 to the last four components.
The runs of the system are defined as follows. Let $\mathit{Int} = \langle \theta_1, \ldots, \theta_n \rangle \in \mathcal{Req}^n$ *be a finite sequence of requests and* $s_\mathcal{D} \in \mathcal{St}_{\mathit{Init}}$ *an initial abstract state of $\mathcal{D}$. The run* $r^{\mathit{Int},s_\mathcal{D}}$ *is inductively defined by the following definition of $\mathcal{D}$'s abstract state* $r_\mathcal{D}^{\mathit{Int},s_\mathcal{D}}(k)$ *at time k:*

$$\begin{aligned} r_\mathcal{D}^{\mathit{Int},s_\mathcal{D}}(0) &= s_\mathcal{D} \\ r_\mathcal{D}^{\mathit{Int},s_\mathcal{D}}(k) &= \begin{cases} \mathsf{cexec}(r_\mathcal{D}^{\mathit{Int},s_\mathcal{D}}(k-1), \theta_k) & \text{if } k \le n \\ r_\mathcal{D}^{\mathit{Int},s_\mathcal{D}}(n) & \text{if } k > n.^2 \end{cases} \end{aligned}$$

Beside Assumption 2 about the attacker's awareness, the definition reflects also Assumption 1 about the defender's knowledge about the attacker. Since in the defined system $\mathcal{A}$'s local state is a projection of $\mathcal{D}$'s local state, agent $\mathcal{D}$ knows at runtime all the information $\mathcal{A}$ has about $\mathcal{D}$'s local state. That is, in

[2] That is after the last request θ_n agent $\mathcal{D}$'s state (and thus, the state of the system) does not change in the considered run.

each point (r, m) agent $\mathcal{D}$ is able to determine with $\mathcal{K}_{\mathcal{D}}(r, m)$ in which local state agent $\mathcal{A}$ is.
In this thesis, we make the two following simplifying assumptions about agent $\mathcal{D}$'s functionality.

Assumption 5 (No Policy Modification).

Agent $\mathcal{D}$ does not modify its confidentiality policy throughout a run. Formally, we assume $conf(r, n) = conf(r, 0)$ for all $n \in \mathbb{N}$ and $r \in \mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$.

Assumption 6 (Fixed Belief Operator).

Agent $\mathcal{D}$'s belief operator $\mathsf{Bel}_{\mathcal{D}}$ is fixed in a system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$.

Because agent $\mathcal{D}$'s belief operator is fixed in an implementation of a system, we usually omit the operator from the two agents' local states in the remainder.

Example 4.4 (The Attacker's Reasoning in the Database Example).

We resume Example 4.3 and assume a fixed confidentiality policy $conf(TCP) = \{a\}$ (while $conf(CCP) = \emptyset$). The excerpt of the system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ is illustrated in Figure 4.3. Our illustrations in the figure focus on agent $\mathcal{D}$ trying to protect its temporary confidential belief a with the control function of Example 4.3. In the figure, only $\mathcal{D}$'s state is shown with the components: history, attacker's view and database instance as component eknowl from top to bottom and left to right. The figure illustrates agent $\mathcal{A}$'s reasoning from the visible components policy, history and view as postulated by the model of Definition 4.5. Until time 1, agent $\mathcal{A}$ cannot distinguish the execution of run r_1 from that of run r_2 because the visible components history and view are equal. The states $\mathcal{A}$ considers possible in point $(r_1, 1)$ are all the states of the points in $\mathcal{K}_{\mathcal{A}}(r_1, 1)$. In Figure 4.3, these points are drawn with a single frame. But at time 2, agent $\mathcal{A}$ with the observed answers is able to single out $\mathcal{D}$'s current state: The set $\mathcal{K}_{\mathcal{A}}(r_1, 2)$ is a singleton. The point within this set is drawn with a double frame. Note, that $\mathcal{A}$'s information gain from time 1 to time 2 in run r_1 is due to the observed reaction refuse. This kind of information gain is called meta-inference *in the line of research of Controlled Interaction Execution [16] because the information gain is not explicit in the reaction refuse.*

How can meta-inferences be characterized formally on the basis of the agent model $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$? The following example discusses possible ways of characterization in the context of the running example.

Example 4.5 (Meta-Inference)**.**

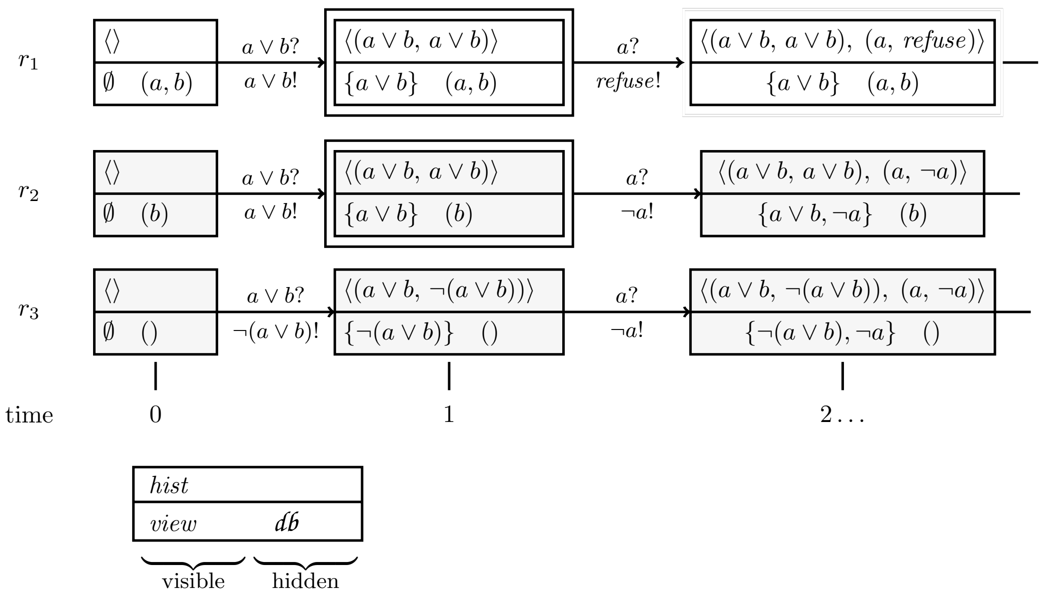
In Fig. 4.3, we notice that although with reasoning $\mathcal{K}_{\mathcal{A}}(r_1, 2)$ agent $\mathcal{A}$ has found out that $\mathcal{D}$ believes a, the view in that situation does not imply formula a with propositional entailment $\vdash_{pl}$, that is, $view(r_1, 2) = \{a \vee b\} \nvdash_{pl} a$. However, the censor of Ex. 4.3, used in Fig. 4.3, simulates the reasoning $\mathcal{K}_{\mathcal{A}}$ by propositional entailment from the view. Thus, the censor of Ex. 4.3 does not correctly simulate operator $\mathcal{K}_{\mathcal{A}}$. How to define a correct simulation formally? In a first attempt, we may stipulate that the censor incorporates everything $\mathcal{A}$ has learned into the view. We may formalize this with the operator $\mathcal{K}_{view}(r, m) = \{(r', m') \in \mathcal{PT}(\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}) \mid view(r, m) = view(r', m')\}$ by requiring $\mathcal{K}_{view}(r, m) \subseteq \mathcal{K}_{\mathcal{A}}(r, m)$ for all points (r, m). But the requirement does not suffice to expose the flaw of the censor's simulation since it holds in the situation of Fig. 4.3. Even more, if $\mathcal{A}$ reasoned with $\mathcal{K}_{view}$, it would find out that $\mathcal{D}$ believes formula a in run r_1 at time 2.
By these considerations, we see the correctness of simulation cannot be formalized in the model $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ only. Rather the correctness should relate the reasoning in the simulation, here $\vdash_{pl}$, to the reasoning $\mathcal{K}_{\mathcal{A}}$ in the model. Thus, we defer a formalization to Part II of this thesis where the simulation of $\mathcal{A}$'s reasoning is detailed for exemplary implementations of agent $\mathcal{D}$.

4.5 Semantics of the Confidentiality Policy

Interacting with an attacking agent $\mathcal{A}$ that has means to reason about $\mathcal{D}$'s belief as outlined in the previous sections, the defending agent $\mathcal{D}$ should guarantee confidentiality as the property of Definition 4.6 below. This property gives a precise semantics to the confidentiality policy *conf* and is summarized as follows.

> Starting with any abstract state of $\mathcal{D}$
> that is an initial state defined by $\mathcal{S}t_{Init}$,

Figure 4.3: The attacking agent $\mathcal{A}$'s reasoning about the defending agent $\mathcal{D}$ of Example 4.4 as postulated by Definition 4.5



r_1
$\langle\rangle$
$\emptyset \quad (a,b)$
$a \vee b?$
$a \vee b!$
$\langle(a \vee b,\, a \vee b)\rangle$
$\{a \vee b\} \quad (a,b)$
$a?$
$refuse!$
$\langle(a \vee b,\, a \vee b),\ (a,\, refuse)\rangle$
$\{a \vee b\} \quad (a,b)$
r_2
$\langle\rangle$
$\emptyset \quad (b)$
$a \vee b?$
$a \vee b!$
$\langle(a \vee b,\, a \vee b)\rangle$
$\{a \vee b\} \quad (b)$
$a?$
$\neg a!$
$\langle(a \vee b,\, a \vee b),\ (a,\, \neg a)\rangle$
$\{a \vee b, \neg a\} \quad (b)$
r_3
$\langle\rangle$
$\emptyset \quad ()$
$a \vee b?$
$\neg(a \vee b)!$
$\langle(a \vee b,\, \neg(a \vee b))\rangle$
$\{\neg(a \vee b)\} \quad ()$
$a?$
$\neg a!$
$\langle(a \vee b,\, \neg(a \vee b)),\ (a,\, \neg a)\rangle$
$\{\neg(a \vee b), \neg a\} \quad ()$
time
0
1
2...
hist
view
db
visible
hidden

after any finite interaction between $\mathcal{A}$ and $\mathcal{D}$,
for each confidential belief $S \in conf(TCP) \cup conf(CCP)$,
by accessing $\mathcal{D}$'s open data components,
including $\mathcal{A}$'s observations of $\mathcal{D}$'s reactions,
$\mathcal{A}$ cannot distinguish the actual abstract state
from an alternative abstract state
in which $\mathcal{D}$ does currently not believe S in case $S \in conf(TCP)$,
or in which $\mathcal{D}$ has never believed S throughout the interaction in
case $S \in conf(CCP)$.

$\mathcal{D}$'s opponent agent $\mathcal{A}$ is assumed to aim at actually determining what $\mathcal{D}$ has believed or currently believes in each of these indistinguishable states or, in other words, at being certain that $\mathcal{D}$ believes some formula $B \in conf(TCP) \cup conf(CCP)$ or has believed some formula $B \in conf(CCP)$. Or to put it into another perspective, while agent $\mathcal{A}$ knows the declaration of the policy, that is the sets $conf(TCP)$ and $conf(CCP)$, it tries to determine the intersections $(conf(TCP) \cup conf(CCP)) \cap \mathsf{Bel}_{\mathcal{D}}(bknowl_n, eknowl_n)$ for the current time n and $conf(CCP) \cap \mathsf{Bel}_{\mathcal{D}}(bknowl_k, eknowl_k)$ for any previous time k. Altogether, the confidentiality property considers agent $\mathcal{A}$ to be a *skeptical reasoner* about $\mathcal{D}$'s belief because agent $\mathcal{A}$ is supposed to accept only certain conclusions about that belief.

Formally, we define the confidentiality property as the following requirement on $\mathcal{D}$'s control function cexec based on the explicit confidentiality policy *conf*. By way of contrast, possibility policies of Chapter 3 may implicitly express confidentiality interests of an agent and their semantics of policy-based secrecy imposes a requirement on the entire system $\mathcal{R}$.

Definition 4.6 (Continuous/Temporary Confidentiality Preservation).

Agent $\mathcal{D}$ with control function $\mathsf{cexec} : \mathcal{St} \times \mathcal{Req} \to \mathcal{St}$ of Definition 4.3 preserves continuous/temporary confidentiality with respect to agent $\mathcal{A}$ iff

for every initial abstract state of the form
$(bknowl_0, eknowl_0, conf_0, hist_0, view_0) \in \mathcal{St}_{Init}$
for all finite sequences $\mathcal{Int} = \langle \theta_1, \ldots, \theta_n \rangle \in \mathcal{Req}^n$ of requests,
for all $S \in conf_0(CCP) \cup conf_0(TCP)$:
there exists alternative initial background knowledge $bknowl'_0$ and evidential knowledge $eknowl'_0$ such that
the abstract state $(bknowl'_0, eknowl'_0, conf_0, hist_0, view_0)$ is in $\mathcal{St}_{Init}$

and the following two properties hold:

1. *Indistinguishability: same open data components*
   for all $1 \leq k \leq n$ it holds

$$conf'_k = conf_k \qquad hist'_k = hist_k \qquad view'_k = view_k$$

where

$$(bknowl'_k, eknowl'_k, conf'_k, hist'_k, view'_k) = \mathsf{cexec}((bknowl'_{k-1}, eknowl'_{k-1}, conf_{k-1}, hist_{k-1}, view_{k-1}), \theta_k)$$

$$(bknowl_k, eknowl_k, conf_k, hist_k, view_k) = \mathsf{cexec}((bknowl_{k-1}, eknowl_{k-1}, conf_{k-1}, hist_{k-1}, view_{k-1}), \theta_k)$$

2. (a) *Continuous preservation: sequence of safe belief*
   If $S \in conf(CCP)$ then
   for all $0 \leq k \leq n$ it holds $S \notin \mathsf{Bel}_{\mathcal{D}}(bknowl'_k, eknowl'_k)$.

   (b) *Temporary preservation: safe current belief*
   If $S \in conf(TCP)$ then it holds $S \notin \mathsf{Bel}_{\mathcal{D}}(bknowl'_n, eknowl'_n)$.

Clearly, continuous confidentiality is stricter than temporary so we expect the control function to be more restrictive and to release less information to agent $\mathcal{A}$. Conversely, by declaring a belief a temporary confidential belief instead of a continuous confidential belief, agent $\mathcal{D}$ may usually share more information with $\mathcal{A}$.
We call any state of $\mathcal{D}$ that $\mathcal{A}$ is not able to distinguish from $\mathcal{D}$'s actual state an *alternative state* (according to property 1 of Definition 4.6 for the initial states and all subsequent states in the interaction).

Example 4.6 (Confidentiality Violation in the Database Example)**.**

We resume Example 4.4 and consider the protection of agent $\mathcal{D}$'s temporary confidential belief $a \in conf(TCP)$. The example should be followed with Figure 4.3 which illustrates the attacker's reasoning during run r_1. The states indistinguishable to $\mathcal{A}$ at time 1 in run r_1 are drawn with a single frame. In this situation, agent $\mathcal{D}$'s state in $(r_2, 0)$ is a possible initial state with safe current belief at times 0 and 1 regarding the temporary confidential belief a. But at time 2, agent $\mathcal{A}$ with the observed answers is able to single out $\mathcal{D}$'s current state (double framed state in Figure 4.3) where $\mathcal{D}$'s belief is not safe, but contains a. Therefore, confidentiality is violated in run r_1 at time 2.

In our running example so far the control function never modifies $\mathcal{D}$'s evidential knowledge, in the example the database instance $\mathfrak{db}$. In Part II, however, our major research interest will be the design of control functions that may modify $\mathcal{D}$'s evidential knowledge after receiving information from $\mathcal{A}$.

4.6 Confidentiality as Policy-Based Secrecy

In this section, we base on our agent model in Runs & Systems from Section 4.4 to relate the semantics of the confidentiality policy from Section 4.5 to policy-based secrecy of Chapter 3. In Chapter 3, policy-based secrecy has been compared to other requirements of information flow confinement in the literature, in particular, to the definitions of secrecy by Halpern and O'Neill in [58].

For comparing confidentiality to policy-based secrecy and related notions, the basic step is translating local confidentiality policies in the agent system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ to one single possibility policy of Definition 3.2. Recall that at each point $(r, m) \in \mathcal{PT}(\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}})$ a possibility policy declares a set of relevant properties of the global state that agent $\mathcal{A}$ should not be able to rule out with its available information in its local state $r_{\mathcal{A}}(m)$. As a special case, the properties may be properties of $\mathcal{D}$'s local state and then are called $\mathcal{D}$-properties of Definition 3.1. The reader may find the idea of the translation in Example 3.2. The example explains how possibility policies and their semantics by policy-based secrecy may express confidentiality interests of agent $\mathcal{D}$. Generally, in order to ensure the confidentiality of the fact that $\mathcal{D}$ *currently believes* $S \in \mathcal{L}_{PL}$[3], the complement fact that $\mathcal{D}$ *does not currently believe* S must be a target of the $\mathcal{D}$-possibility policy. Thus, for each temporary confidential belief S we define the following $\mathcal{D}$-property which is relevant for the protection of S:

$$I_S^{TCP} = \{(r', m') \mid S \notin \mathsf{Bel}_{\mathcal{D}}(\mathit{bknowl}(r', m'), \mathit{eknowl}(r', m'))\}. \tag{4.1}$$

Here, $\mathit{bknowl}(r', m')$ denotes the defender $\mathcal{D}$'s local background knowledge in run r' at time m' of a system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of Definition 4.5. In the same way, we refer to other data components of $\mathcal{D}$'s local state in run r' at time m'. The set I_S^{TCP} is a $\mathcal{D}$-property of Definition 3.1 since $\mathcal{D}$ is able to verify this property in point (r, m) at runtime by checking $S \notin \mathsf{Bel}_{\mathcal{D}}(\mathit{bknowl}(r, m), \mathit{eknowl}(r, m))$, or equivalently, $\mathcal{K}_{\mathcal{D}}(r, m) \subseteq I_S^{TCP}$ as required by Definition 3.1.

Likewise, to ensure the confidentiality of the fact that $\mathcal{D}$ *has believed or currently believes* $S \in \mathcal{L}_{PL}$[4], the complement fact that $\mathcal{D}$ *never has believed* S,

[3] That is, S is a temporary confidential belief of $\mathit{conf}(TCP)$.

[4] That is, S is a continuous confidential belief of $\mathit{conf}(CCP)$.

neither previously nor currently, must be target of the $\mathcal{D}$-possibility policy. This fact need not define a $\mathcal{D}$ property, but for each continuous confidential belief S defines the following property of runs at time m:

$$R^{CCP}_{S,m} = \{r' \in \mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}} \mid \text{for all } n \leq m : \\ S \notin \mathsf{Bel}_{\mathcal{D}}(bknowl(r', n), eknowl(r', n))\}. \quad (4.2)$$

Following Section 3.4, an attacker reasoning about runs and their properties can be equivalently handled by policy-based secrecy, representing a property R of runs as the property $\mathcal{PT}(R)$ of states. The main argument in Section 3.4 was that $\mathcal{A}$'s reasoning about runs with operator $\mathcal{R} \circ \mathcal{K}_{\mathcal{A}}$ bases on the information $r_{\mathcal{A}}(m)$ in the agent's local state in the current run r at time m. Therefore, run-based reasoning is not more powerful than state-based reasoning with $\mathcal{K}_{\mathcal{A}}$.
These are the key ideas of the translation from local confidentiality policies to one system-wide possibility policy in the following definition.

Definition 4.7 (Confidentiality Policies as Possibility Policies)**.**

Let $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ *be as system of defender* $\mathcal{D}$ *and attacker* $\mathcal{A}$*. Then, agent* $\mathcal{D}$*'s local confidentiality policy* $conf(r, m)$ *defines the possibility policy* $\mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}} : \mathcal{PT}(\mathcal{R}) \rightarrow \mathcal{P}(\mathcal{P}(\mathcal{PT}(\mathcal{R})))$ *as follows:*

$$\begin{array}{rlll} \mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}(r, m) = & \{I^{TCP}_S & \mid S \in conf(TCP)(r, m)\} & \cup \\ & \{\mathcal{PT}(R^{CCP}_{S,m}) & \mid S \in conf(CCP)(r, m)\}. & \end{array}$$

The following theorem shows that we can adequately express the requirements of Definition 4.6 in the Runs & Systems framework using $\mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}$-based secrecy of Definition 3.3.

Theorem 4.1 (Equivalence of Confidentiality Preservation and $\mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}$-based Secrecy)**.**

Let $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ *be a system of defender* $\mathcal{D}$ *and attacker* $\mathcal{A}$*. Then, agent* $\mathcal{D}$ *maintains* $\mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}$*-based secrecy with respect to agent* $\mathcal{A}$ *according to Def. 3.3 iff* cexec *preserves continuous/temporary confidentiality according to Def. 4.6.*

The proof of the theorem bases on Assumption 3, Assumption 5 and Definition 4.5 which reflects Assumption 2.

Corollary 4.1.

Let $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ be a system of defender $\mathcal{D}$ and attacker $\mathcal{A}$ with continuous confidentiality policy only (that is $conf(TCP) = \emptyset$). Further, we define $\mathtt{policy}\mathcal{R}(r,m) = \{R^{CCP}_{S,m} \mid S \in conf(CCP)(r,m)\}$. Then, agent $\mathcal{D}$ maintains $\mathtt{policy}\mathcal{R}$-based run-based secrecy with respect to agent $\mathcal{A}$ according to Definition 3.5 iff cexec *preserves continuous confidentiality according to Definition 4.6.*

Example 4.7 (Confidentiality Properties, Total f-Secrecy and C-Secrecy)**.**

Consider a system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of defender $\mathcal{D}$ and attacker $\mathcal{A}$. Further assume that agent $\mathcal{D}$'s belief may contain either combination of the atoms $a, b \in \mathcal{L}_{PL}$. Consider now a temporary confidentiality policy $conf(TCP)(r,m) = \{a,b\}$ for all system states (r,m). According to Theorem 4.1 this confidentiality policy is equivalently enforced by policy-based secrecy with $\mathcal{D}$-possibility policy $\mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}(r,m) = \{I^{TCP}_a, I^{TCP}_b\}$ of Definition 4.7. But it cannot be enforced by total f-secrecy since there is no $\mathcal{D}$-information function f with co-domain $V = \{v_1, v_2\}$ and equivalent $\mathcal{D}$-possibility policy

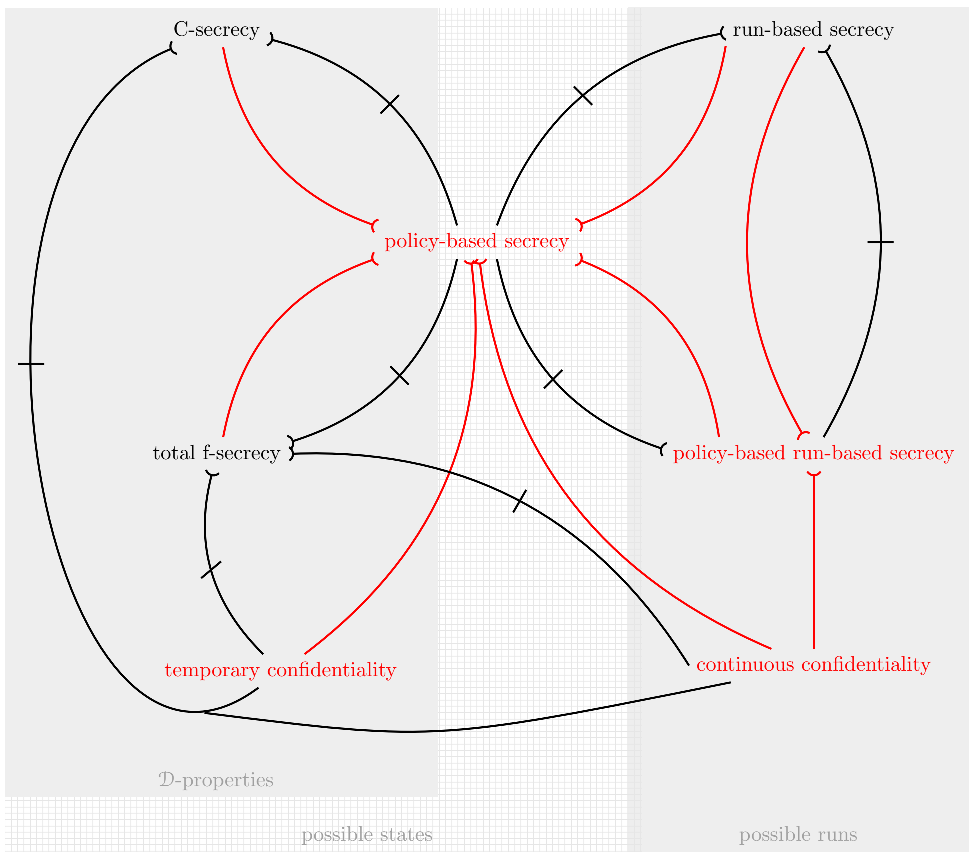
$$\begin{aligned}\mathtt{policy}_f(r,m) &= \{I_{v_1}, I_{v_2}\}\\ &= \{I^{TCP}_a, I^{TCP}_b\} = \mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}(r,m).\end{aligned}$$

The reason is that $\mathcal{D}$-information functions define disjoint $\mathcal{D}$-properties so that $I_{v_1} \cap I_{v_2} = \emptyset$ whereas $I^{TCP}_a \cap I^{TCP}_b \neq \emptyset$ and $I^{TCP}_a \neq I^{TCP}_b$ as we have assumed that $\mathcal{D}$'s belief might contain either combination of the atoms a and b. Thus, by Proposition 3.1 the confidentiality policy cannot be equivalently enforced by total f-secrecy.
The confidentiality policy also cannot be enforced by C-secrecy since there is no $\mathcal{A}$-allowability function C and equivalent $\mathcal{D}$-possibility policy

$$\begin{aligned}\mathtt{policy}_C(r,m) &= \{\mathcal{K}_\mathcal{D}(r_1,m_1), \mathcal{K}_\mathcal{D}(r_2,m_2)\}\\ &= \{I^{TCP}_a, I^{TCP}_b\} = \mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}(r,m).\end{aligned}$$

The reason is that I^{TCP}_a contains at least two different $\mathcal{D}$-information sets where either b is an element of $\mathcal{D}$'s belief or not. Thus, by Proposition 3.2 the confidentiality policy cannot be equivalently enforced by C-secrecy.

Figure 4.4: Relationship among secrecy notions and confidentiality properties



A — B	Property A **can** be expressed in terms of property B
A —+ B	Property A **cannot** be expressed in terms of property B

When assuming that agent 𝒟's belief does not change during a run, this example works for continuous confidentiality policies as well.

An overview of the relationship between secrecy notions and confidentiality properties can be found in Figure 4.4.

Chapter 5

Intermediate Conclusion

5.1 Summary

In Part I, we completed the formal specification of the security engineering task that provides an agent with means to declare its confidentiality interests with a well-defined semantics. In the formal specification, we focused on a system of two epistemic agents sharing information – a requesting agent $\mathcal{A}$ as a potential attacker and a reacting agent $\mathcal{D}$ as a defender of its confidentiality interests against $\mathcal{A}$. As the basic framework of the formal specification we have chosen the Runs & Systems framework following Halpern and O'Neill's formalization of secrecy in multiagent systems in [58]. On this basis, we could relate the model of the attacker $\mathcal{A}$'s information gain and its confinement to other definitions in the literature.

The model of information gain was adopted from Halpern and O'Neill's definition of secrecy and agrees with other possibilistic models discussed in Section 3.5. Possibility policies confine the attacker's information gain as defined by the policies' formal semantics of policy-based secrecy or policy-based run-based secrecy respectively. These definitions of secrecy generalize Halpern and O'Neill's definitions, cf. Figure 3.1. The generalization was proven in Chapter 3 and motivated in Chapter 4 as an appropriate means for declaring an agent's confidentiality interests, cf. Figure 4.4.

Complementing the formal specification, we discussed the relation of policy-based secrecy to other definitions in the literature in Section 3.5. This discussion gives examples how via policy-based secrecy we may understand the confidentiality property of Chapter 4 in terms of other security requirements in the literature or at least compare them.

5.2 Discussion and Future Work

Agent Model As Figure 4.2 shows, in this thesis, we use the control function to separate the defending agent's functionality and the interaction interface provided to the attacker which eases the modeling of the agent system from the security engineering point of view in Section 4. Recalling Section 1, a BDI-agent is an autonomous computing system that interacts with other agents in its environment directed by internal goals [95]. As a BDI-agent the defending agent should autonomously consider the decision of the control function when planning its action or deciding on its options. The reader may find a proposal of a BDI-agent which tries to preserve its secrets by choosing appropriate plans in [75]. In contrast, in this thesis we do not consider a BDI-agent's full internal functionality. Instead, we will design the agent to control the information flow during its belief operations. This control will turn out to be involved, so that information flow control within a BDI-agent with its full complexity is beyond the scope of this work.
In the design of agent $\mathcal{D}$ we decided that $\mathcal{D}$'s simulation of $\mathcal{A}$'s reasoning is separate from $\mathcal{D}$'s process of reasoning about the world in general. In the following, we explain this decision. The motivation behind the decision is to strictly separate the agent's functionality in order to reduce the complexity of designing the agent. The separation is justified by the assumptions underlying the focused scenario of agents. In the focused scenario of $\mathcal{D}$ and $\mathcal{A}$, both agents do not acquire information in any other way but through the interaction between $\mathcal{D}$ and $\mathcal{A}$. Thus, it is possible and straightforward for $\mathcal{D}$ to reason about what $\mathcal{A}$ might conclude about $\mathcal{D}$'s belief without any consideration of the world or other agents in their common environment. Nevertheless, the assumptions justifying the separation may be argued which we will do in the next paragraph.

Postulates About Defender and Attacker We might reconsider several of the assumptions about the agent's capabilities in Section 4.1 and Section 4.3. Especially, Assumption. 1 and Assumption 4 can be argued in many application scenarios.

By Assumption. 1, agent $\mathcal{D}$ *a priori knows* all sources of information that $\mathcal{A}$ can access before the interaction and that are relevant for protecting $\mathcal{D}$'s belief. This assumption must be reconsidered if there are many agents in the environment that contribute information to $\mathcal{D}$'s belief and to which $\mathcal{D}$ discloses parts of its belief. Then, $\mathcal{D}$ might need to protect its confidential belief against a group of agents that might by purpose or accidentally share information and this way harm $\mathcal{D}$'s confidentiality interests. If $\mathcal{D}$ assumed every other agent to freely distribute any information $\mathcal{D}$ disclosed to that agent, $\mathcal{D}$ would rather cautiously control its disclosure of information to other agents and this way disregard its interest of cooperatively sharing information.

Further, since simulating $\mathcal{A}$'s postulated reasoning is supposed to be generally computationally expensive, in place of Assumption 4, restrictions on agent $\mathcal{A}$'s assumed computational capability should be taken into account. A starting point might be the model of resource-bounded reasoning of an agent in Runs & Systems proposed by Halpern and Pucella in [59]. They introduce an *algorithmic knowledge* operator $\mathbf{X}_{\mathcal{Y}}$ which models the computational capabilities of agent $\mathcal{Y}$.
Moreover, Assumption 4 postulates that $\mathcal{A}$ is a *skeptical reasoner* which thus aims to become certain about $\mathcal{D}$'s belief and in this sense is rather cautious with its conclusions about that belief. Postulating a less cautious attacker would require more effort in the enforcement of confidentiality, cf. [76] for a discussion and a different approach.

If we consider a scenario of more than two agents, then we might assume that communication acts are visible to all agents (but, of course, not the communicated data). With this assumption, we make sure that agent $\mathcal{A}$ is aware of the fact that, during its interaction with agent $\mathcal{D}$, agent $\mathcal{D}$ does not communicate privately with other agents. In this sense, we may view the focused scenario of this thesis as a private conversion between $\mathcal{A}$ and $\mathcal{D}$ among a group of several agents.

Policy Declaration In this thesis, the defending agent $\mathcal{D}$ can only specify its confidentiality interests within the propositional language $\mathcal{L}_{PL}$ with the further option to declare either a temporary or continuous confidential belief. However, within the language $\mathcal{L}_{PL}$, agent $\mathcal{D}$ might only adequately express its confidentiality interests with an extensive or even infinite confidentiality policy. For example, an interest of the manufacturer in the scenario of Section 1.2 might be that a supplier does not know any agreement of the manufacturer with another supplier on the deployment of some part. However, there might be a large (or infinite) number of possible agreements. To make the policy declaration concise, the authors of [25] introduce a richer language for confidentiality policies than that of Section 4.2 by means of which the security administrator can declare a combination of attributes (of a relation) sensitive. For example, the manufacturer's interest towards supplier #1 may be declared by the formula $\mathit{deploy}(X, Y) \wedge (Y \neq \mathit{supplier}\#1)$ where X is a variable ranging over parts and Y over suppliers.

Moreover, agent $\mathcal{D}$ might be interested to hide that it does not believe something from $\mathcal{A}$ or other properties of its belief. For example, the manufacturer may want to hide from his competitors that he is not able to reach a conclusion about the realizability of a proposed product yet. In the context of incomplete databases (essentially, a satisfiable set of propositional formulas) Biskup and Weibert in [33] use *modal logic* to express what a database user should not know about what the database knows or *does not know*. In future work, we might follow their approach and use a modal language for the declaration of the confidentiality policy.

At last, a richer language of the confidentiality policy might enable agent $\mathcal{D}$ to express more accurately which information should be hidden. This has an impact on how *cooperatively* $\mathcal{D}$ may share information with $\mathcal{A}$.

Declassification Declassification allows to relax the requirements of information flow confinement such as non-interference of the trace-based model in Section 2.2. There are several ways to declassify information about the values of hidden program variables, called dimensions in [85]. Generally, a property of the value of a hidden variable may be declassified which means that the attacker is allowed to learn this property, but *non-interference modulo* *declassification* still requires that the attacker is not able to distinguish between values that have the declassified property (or those that do not have the property respectively). The declassification may depend on a condition: The program must reach a release point to declassify a particular property; or another property must become true for the declassification of some property. The security model of the agent system in Chapter 4 allows the attacker $\mathcal{A}$ to access the view and the confidentiality policy by Assumption 2. Apart from that $\mathcal{A}$ should not learn anything about $\mathcal{D}$'s belief. This requirement is similar to non-interference modulo declassification, where the view and the policy are declassified. Yet, it is not straightforward to formalize the requirement that $\mathcal{A}$ should not learn anything about $\mathcal{D}$'s state except for the accessible components, at least in the model of the agent system of Chapter 4. We discussed this matter in Example 4.5. By way of contrast, Balliu et al in [8] formalize declassification for sequential while programs in the logic of knowledge and time (Section 2.5) and in subsequent work [7] present a method for automated verification.

Part II

Enforcing Confidentiality in Agent Interactions

In Part I of this thesis, we focused on *postulates* about the reasoning capabilities of the requesting agent $\mathcal{A}$ as an attacker of the reacting agent $\mathcal{D}$'s interests in the confidentiality of its belief. In Part II, now, we focus on the *simulation* and computation of $\mathcal{A}$'s postulated reasoning as a means to enforce $\mathcal{D}$'s confidentiality interests. Moreover, the contributions of this part will be centered around the second part of the security engineering task, the *implementation and verification* of a confidentiality preserving agent $\mathcal{D}$. In this task, we mainly investigate $\mathcal{D}$'s *controlled* processing of information received from $\mathcal{A}$ by means of *belief change operators of update and revision* as illustrated in the dialog in Section 1.2. Our contributions are motivated by the research questions raised in that section.
Above all, we will outline the second task of security engineering. Afterwards, we will give an overview of our general approach and the organization of Part II.
Essentially, the implementation and verification of agent $\mathcal{D}$ must be based on the postulates about the agents' capabilities in Part I. Of course, the attacking agent $\mathcal{A}$ further might exploit details about $\mathcal{D}$'s implementation to reason about $\mathcal{D}$'s belief. These further means of attack must be accounted for in the second task of security engineering. This task comprises the following activities:

1. The defender $\mathcal{D}$'s implemented functionality must be outlined.

2. $\mathcal{A}$'s awareness of $\mathcal{D}$'s implementation may be postulated by defining the set $\mathcal{S}t_{Init}$ of possible initial states of the system model $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of Definition 4.5 (in addition to $\mathcal{A}$'s awareness of the implementation of the control function by Assumption 2).

3. The implementation of the control function must be verified to enforce the confidentiality property of Definition 4.6.

In the construction of agent $\mathcal{D}$, we design agent $\mathcal{D}$'s control function cexec in the line with research of Controlled Interaction Execution (CIE) [16, 18]. At a potential violation of $\mathcal{D}$'s confidentiality policy, the proposed control functions *refuse* $\mathcal{A}$'s request. From the research of CIE it is well-known that control functions with refusal in particular have two major challenges: First, they must simulate the attacker's postulated reasoning about $\mathcal{D}$'s current belief and, more intricately, $\mathcal{D}$'s previous belief. Second, by means of an appropriate strategy of generating refusals, the control functions must prevent harmful *meta-inferences* the attacker $\mathcal{A}$ bases on refusal notifications, cf. Example 4.5. Yet, this strategy must still respect agent $\mathcal{D}$'s *interest of sharing information as cooperatively* as possible with $\mathcal{A}$.

The control function cexec in this thesis simulates $\mathcal{A}$'s reasoning $\mathcal{K}_{\mathcal{A}}$ about the system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ from its runtime information $r_{\mathcal{A}}(m)$ by an operator $\mathsf{skeptical}$ on the attacker view $view$. As common in CIE, we design control functions that preserve the invariant $S \notin \mathsf{skeptical}(view)$ for all pieces S of information whose disclosure in the view reveals some current or previous confidential belief to $\mathcal{A}$. This invariant implies the enforcement of the confidentiality property if the control function *correctly simulates* $\mathcal{A}$'s reasoning, cf. Example 4.5.

Finally, we outline the organization of this part. In Chapter 6, we present and motivate two example implementations of $\mathcal{D}$'s belief component (as Activity 1 of the second task of security engineering) used in the following chapters. At the end of the chapter, we indicate a possible combination and enhancements of the exemplary implementations.

Chapter 7 focuses on the *controlled processing of update requests* while $\mathcal{D}$'s belief bases on a simple logic-oriented database model. We exploit its simplicity to investigate more intricate problems, namely, the correctness of the simulation of $\mathcal{A}$'s postulated reasoning, the protection of continuous confidential belief and cooperative information sharing.

Chapter 8 focuses on the *controlled processing of revision requests* while $\mathcal{D}$'s belief bases on incomplete evidential knowledge and nonmonotonic reasoning. This implementation of $\mathcal{D}$'s belief is standard in artificial intelligence for epistemic agents in inaccessible environments. Due to the complexity of this implementation, we mainly investigate the computation of the simulation of $\mathcal{A}$'s postulated reasoning. Problems as studied in Chapter 7 are left for future work.

In both chapters, we will present control functions for handling $\mathcal{A}$'s requests and prove their effectiveness in enforcing confidentiality.

Chapter 6

Implementing the Defender in the Scenario of Two Agents

In this chapter, we present two implementation of $\mathcal{D}$'s belief component. The first implementation is a simple logic-oriented database model. With this implementation we study several aspects brought up by the research community on database security in Chapter 7. A major aspect is the conflict between enforcing confidentiality and preserving integrity constraints of the database schema when updating a database. A further motivation for this implementation is that its simplicity makes several tasks of security engineering easier. Yet, some particularities of the database model are inadequate to implement the belief component of an intelligent agent $\mathcal{D}$ which reasons about an inaccessible and dynamic environment. We will discuss this matter in Section 6.4. Therefore, the second implementation bases agent $\mathcal{D}$'s belief on a nonmonotonic consequence relation. Nonmonotonic consequence relations have been introduced as a model of rational reasoning in the artificial intelligence research community. In Section 6.4, we hint at how both implementations might be combined in future work. Figure 6.1 and Figure 6.2 visualize the two exemplary implementations of $\mathcal{D}$'s functionality as seen by the security engineer. In the following, we describe the major aspects of the two implementations as shown by the figures. Then, we finish the introduction with an outline of the organization of this chapter.

In the scenario of two epistemic agents that we consider in this thesis, cf. Chapter 4, the defending agent $\mathcal{D}$ interacts with agent $\mathcal{A}$ for the purpose of information sharing, but only reacts to $\mathcal{A}$'s requests. The basis for information sharing is an agreement on a propositional base language $\mathcal{L}_{PL}$ over a countably infinite alphabet $\mathcal{At}$.

One core of $\mathcal{D}$'s functionality is its belief component. The *belief* comprises all formulas of the common propositional language that the agent accepts to be true in its environment. In this thesis, we only consider $\mathcal{D}$'s processing of the information shared with agent $\mathcal{A}$ and the resulting effect on $\mathcal{D}$'s belief instead of the full functionality of a BDI-agent, cf. Section 1.1.

For defining the semantics of the formulas of the base language $\mathcal{L}_{PL}$, essentially, the environment may be ideally seen as a truth-value assignment of all atomic propositions in the alphabet $\mathcal{At}$. Each truth-value assignment thus describes some state of the environment and is viewed as a *world*. Thus, the base language $\mathcal{L}_{PL}$ establishes a common vocabulary to exchange information about the agents' environment. This vocabulary is also the basis for the two implementations of $\mathcal{D}$'s belief component which we outline in the following.

In the first implementation of Figure 6.1, agent $\mathcal{D}$ acquired *complete information* about its environment stored in a *complete propositional database instance* $db \subset_{fin} \mathcal{At}$. By the *closed world assumption (CWA)* [1], agent $\mathcal{D}$ assumes that all atoms a it has not found to be true (that is, $a \notin db$) are actually false in the environment. This way, the agent models its environment by a unique truth-value assignment. The agent's background knowledge defines which database instances are reasonable models of the environment and is applied during a *database update*. This knowledge is represented as a set IC of *integrity constraints* of the database instance.

In the second implementation of Figure 6.2, the agents are situated in an *inaccessible and dynamic environment* [94] where the closed world assumption is not applicable. Thus, agent $\mathcal{D}$ faces generally *incomplete information* gathered in a finite set $as \subset_{fin} \mathcal{L}_{PL}$ of *current assertions*. From these assertions, agent $\mathcal{D}$ *defeasibly reasons* about which formulas it accepts as true with its background knowledge and, in this process, forms its belief. The background knowledge is represented as a *fixed nonmonotonic consequence relation* $\mid\!\sim_{\mathcal{D}}$.

Although agent $\mathcal{A}$ is part of the environment, for simplicity, agent $\mathcal{D}$ does not consider $\mathcal{A}$'s reasoning in its belief, but simulates $\mathcal{A}$'s reasoning separately, cf. the discussion in Section 5.2.

In both implementations, in principle, we consider the following types of valid requests agent $\mathcal{A}$ may send to agent $\mathcal{D}$: Let $A \in \mathcal{L}_{PL}$

- `que`(A) (query about current belief of A):
  reply to "$A \in \mathsf{Bel}_{\mathcal{D}}(\mathit{bknowl}, \mathit{eknowl})$?".
- `rev`(A) (belief revision by A):
  add A to the evidential knowledge *eknowl* or merge A with *eknowl*.
- `up`(A) (belief update by A):
  update the evidential knowledge *eknowl* with A.

In the application scenario of Section 1.2, we have presented a dialog with these types of requests in an informal way. Whereas via a belief revision request agent $\mathcal{A}$ *informs* $\mathcal{D}$ about the current situation, via an update request $\mathcal{A}$ *informs* $\mathcal{D}$ about a change in the current situation. In particular, agent $\mathcal{A}$ indicates to $\mathcal{D}$ whether to process the information A by an update operator or by a revision operator.[1]
Agent $\mathcal{D}$'s processing of information from agent $\mathcal{A}$ and its subsequent reactions to $\mathcal{A}$ might reveal confidential belief to $\mathcal{A}$. Such a disclosure of confidential pieces of information has rarely been investigated in computer security research, but will be a major issue of this part of the thesis.
As Figure 6.1 and Figure 6.2 indicate, we will restrict to certain types of requests depending on the different representations of agent $\mathcal{D}$'s epistemic state. Moreover, in the database implementation we will not use the elementary type of update request from the above list, but *view update transaction requests*. However, a transaction is comprised of a series of elementary updates.
Finally, Figure 6.1 and Figure 6.2 show the requesting agent $\mathcal{A}$'s skeptical reasoning about $\mathcal{D}$'s belief as postulated by the security engineer in Chapter 4. The postulated reasoning has to be simulated by agent $\mathcal{D}$ to achieve its confidentiality aims. The general tasks within the simulation have been

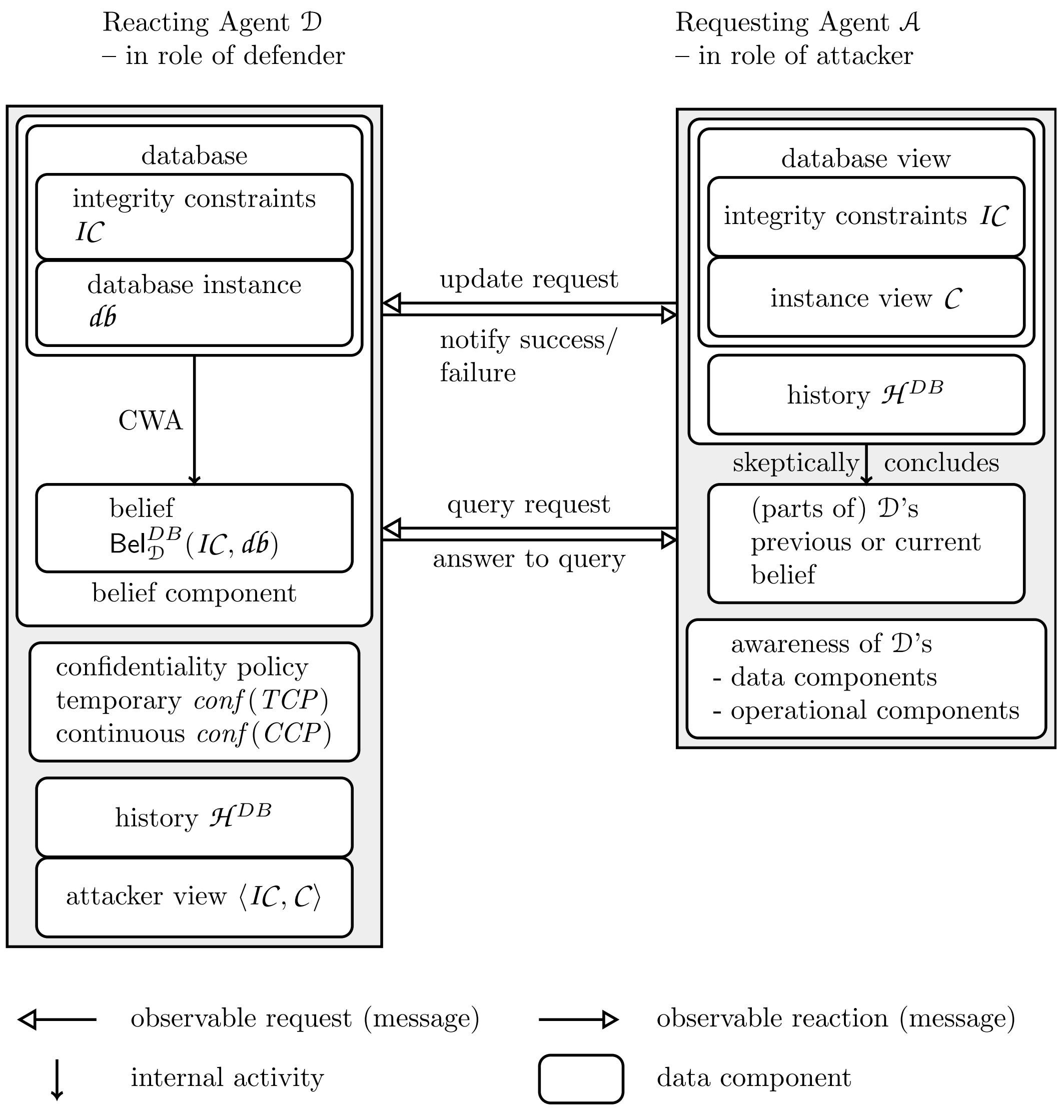
[1] In the informal dialog of Section 1.2 the communication between the two agents is more realistic. There, agent $\mathcal{A}$ indicates a change of the situation by saying "the incompatibility has been resolved" which suggests to $\mathcal{D}$ to process the information with an update operator. By making this explicit in the type of request as we do in this thesis, we simplify the agents' communication language and its interpretation by $\mathcal{D}$.

described in Section 4.2.2. A major contribution of this thesis will be the simulation of $\mathcal{A}$'s postulated skeptical reasoning and its computation for both implementations which is treated in Chapter 7 and Chapter 8 respectively.
The remainder of this chapter is organized as follows. Section 6.1 and Section 6.2 detail the implementation of the belief component and the processing of requests (without a control function for confidentiality enforcement). Section 6.3 presents a selection of related work with approaches to extend the functionality of agent $\mathcal{D}$ of the exemplary implementation. Finally, in Section 6.4 we outline a possible extension of $\mathcal{D}$'s functionality.

6.1 Implementation with Complete Propositional Databases

6.1.1 Agent $\mathcal{D}$'s Belief

As an exemplary implementation, we deploy agent $\mathcal{D}$ with a complete propositional database as a basis for its belief. The implementation is outlined by Figure 6.1. The notion of complete propositional databases and the related terminology originates from the database community. The terminology and concepts from the database area are surveyed in [1] for advanced data models like the relational model. In the outline of the implementation, we will adapt the database community terminology, but we relate each concept to the multiagent scenario of Chapter 4.
When a database is set up, usually by a database administrator, the administrator defines the structure of the data in the *database schema* within a data definition language. The database schema of propositional complete databases simply comprises an alphabet $\mathcal{At}$ of propositional atoms and a finite set $\mathcal{IC} \subset_{fin} \mathcal{L}_{PL}$ of formulas called *integrity constraints*. In this thesis, we assume that the alphabet of the schema is the alphabet of the agents' base language.
Data is stored as a finite set $\mathcal{db} \subset_{fin} \mathcal{At}$ of atoms taken from the alphabet $\mathcal{At}$ of the schema. This set is called the *database instance* and has to further comply with the integrity constraints of the schema in the following sense. Each database instance $\mathcal{db}$ inductively defines a propositional interpretation



Reacting Agent $\mathcal{D}$
– in role of defender
Requesting Agent $\mathcal{A}$
– in role of attacker
database
integrity constraints $\mathcal{IC}$
database instance db
CWA
belief $\mathsf{Bel}_{\mathcal{D}}^{DB}(\mathcal{IC}, db)$
belief component
confidentiality policy
temporary $conf(TCP)$
continuous $conf(CCP)$
history $\mathcal{H}^{DB}$
attacker view $\langle \mathcal{IC}, \mathcal{C} \rangle$
update request
notify success/ failure
query request
answer to query
database view
integrity constraints $\mathcal{IC}$
instance view $\mathcal{C}$
history $\mathcal{H}^{DB}$
skeptically concludes
(parts of) $\mathcal{D}$'s previous or current belief
awareness of $\mathcal{D}$'s
- data components
- operational components
observable request (message)
observable reaction (message)
internal activity
data component

Figure 6.1: Implemented scenario of Figure 4.1 as seen by the security engineer basing $\mathcal{D}$'s functionality on a complete propositional database instance: agent $\mathcal{D}$ to be constructed by the security engineer and agent $\mathcal{A}$ as postulated by the security engineer to be simulated by $\mathcal{D}$

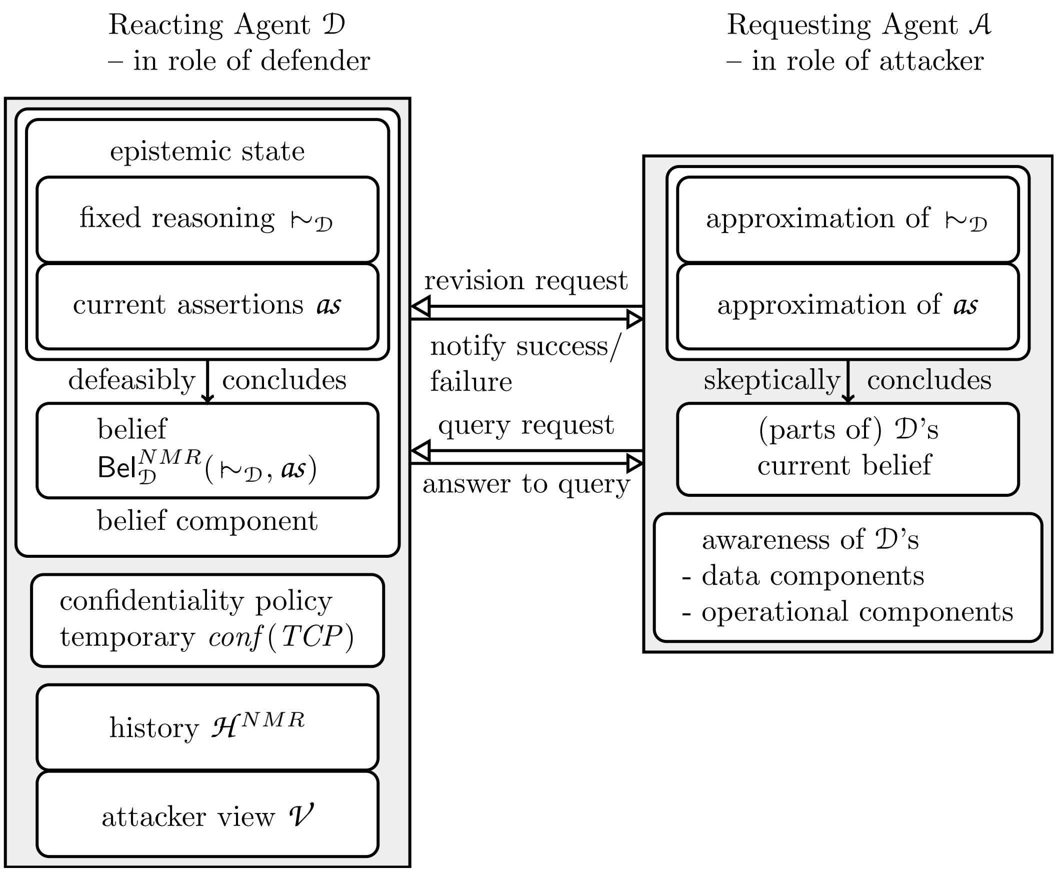

Reacting Agent $\mathcal{D}$
– in role of defender
Requesting Agent $\mathcal{A}$
– in role of attacker
epistemic state
fixed reasoning $\mid\!\sim_{\mathcal{D}}$
current assertions as
defeasibly concludes
belief
$\mathsf{Bel}_{\mathcal{D}}^{NMR}(\mid\!\sim_{\mathcal{D}}, as)$
belief component
confidentiality policy
temporary $conf(TCP)$
history $\mathcal{H}^{NMR}$
attacker view $\mathcal{V}$
revision request
notify success/ failure
query request
answer to query
approximation of $\mid\!\sim_{\mathcal{D}}$
approximation of as
skeptically concludes
(parts of) $\mathcal{D}$'s current belief
awareness of $\mathcal{D}$'s
- data components
- operational components

Figure 6.2: Implemented scenario of Figure 4.1 as seen by the security engineer basing $\mathcal{D}$'s functionality on nonmonotonic reasoning: agent $\mathcal{D}$ to be constructed by the security engineer and agent $\mathcal{A}$ as postulated by the security engineer to be simulated by $\mathcal{D}$

$I^{\mathcal{db}}$, as usual for the propositional connectives in $\mathcal{L}_{PL}$, and for all atoms $a \in \mathcal{At}$ via $I^{\mathcal{db}} \models_{PL} a$ iff $a \in \mathcal{db}$. Then, the interpretation defined by an instance $\mathcal{db}$ is required to satisfy all integrity constraints $\mathcal{IC}$ of the schema. In the following, we use the notation $\mathcal{db} \models_{PL} F$ to abbreviate $\mathcal{Int}^{\mathcal{db}} \models_{PL} F$ with $F \in \mathcal{L}_{PL}$. The operator $\models_{PL}$ bases on the *closed world assumption* (CWA) [1] since all atoms not included in the database instance $\mathcal{db}$ are set to false by the operator.

Definition 6.1 (Database)**.**

Let $\mathcal{IC}$ be integrity constraints and $\mathcal{db}$ a database instance such that $\mathcal{db} \models_{PL} \mathcal{IC}$. Then, the pair $\langle \mathcal{IC}, \mathcal{db} \rangle$ is called a database.[2]

In the considered implementation of agent $\mathcal{D}$, the agent's belief are all formulas that are true in the database instance $\mathcal{db}$ under the closed world assumption:

$$\mathsf{Bel}^{DB}_{\mathcal{D}}(\langle \mathcal{IC}, \mathcal{db} \rangle) = \{B \in \mathcal{L}_{PL} \mid \mathcal{db} \models_{PL} B\}. \tag{6.1}$$

Commonly, a database user only acquires information from the database instance as needed within his duties. Thus, the information frequently needed by the user is maintained in a *database view.* A general purpose of a database view is to hide irrelevant information and to restructure data and, this way, to assist the database user in the access and maintenance of information [1]. The restructuring makes sense in the context of relational databases where the design of the database schema defining the structure of the data aims at optimal performance of the database operations and minimal redundancy of the stored data. In the context of relational databases, a materialized database view (the stored result of a relational query) avoids more costly access to the database instance itself. In the context of information security, a database user may only be granted access or update rights on some database view, not on the full database instance, cf. [15, Chapter 17.2].
In the context of our agent scenario, the *database view* consists of two parts. The first part is the set $\mathcal{IC}$ of integrity constraints of the database schema. These constraints are useful to agent $\mathcal{A}$ to find out the reasons why updating the database instance with additional information does not comply with the

[2] Commonly, a database would contain the whole definition of the schema. But to shorten the definition, we omitted the alphabet $\mathcal{At}$ since it is not needed for presenting the other concepts clearly.

integrity constraints of the schema and why the updating thus fails. Knowing the reasons for the failure of an update helps $\mathcal{A}$ to take appropriate actions. Therefore, the integrity constraints are included in a database view.
The second part is the set $\mathcal{C}$ of all formulas that agent $\mathcal{A}$ knows to be true in the database instance $\mathcal{db}$ and thus is called the *instance view*.

Definition 6.2 (Database View).

Let $\langle \mathcal{IC}, \mathcal{db} \rangle$ be a database and $\mathcal{C} \subset_{fin} \mathcal{L}_{PL}$ a finite set of formulas such that $\mathcal{db} \models_{PL} \mathcal{C}$. Then, the set $\mathcal{C}$ is called a instance view of instance $\mathcal{db}$ and the pair $\langle \mathcal{IC}, \mathcal{C} \rangle$ a database view.

6.1.2 The Agents' Interaction

In the exemplary implementation with propositional complete databases, cf. Figure 6.1, in this thesis, we consider a restricted interaction interface of the reacting agent $\mathcal{D}$ to the requesting agent $\mathcal{A}$. Via this interface, agent $\mathcal{A}$ may send *query* and *view update transaction requests* to agent $\mathcal{D}$ defined by the set $\mathcal{Req}^{DB}$ of valid request messages

$$\begin{aligned} \mathcal{Req}^{DB} &= \{\texttt{que}(A) \mid A \in \mathcal{L}_{PL}\} \cup \\ &\{\texttt{up}(\{L_1, \ldots, L_n\}) \mid L_1, \ldots, L_n \in \mathcal{L}_{PL} \text{ literals over pairwise distinct atoms}\} \end{aligned} \tag{6.2}$$

An extension of this interface is discussed in Section 6.4. In the remainder of this subsection, we detail how agent $\mathcal{D}$ reacts to agent $\mathcal{A}$'s requests without controlling its reactions for the sake of confidentiality.
On query request $\texttt{que}(A)$, agent $\mathcal{D}$ returns the result of the following query evaluation function.

Definition 6.3 (Query Evaluation).

Let $\langle \mathcal{IC}, \mathcal{db} \rangle$ be a database and $A \in \mathcal{L}_{PL}$. Then, the query evaluation function is defined by

$$\texttt{eval}(A)(\mathcal{IC}, \mathcal{db}) := \begin{cases} \textit{true} & \textit{if } A \in \mathsf{Bel}_{\mathcal{D}}^{DB}(\mathcal{IC}, \mathcal{db}), \\ \textit{false} & \textit{otherwise.} \end{cases}$$

Commonly, in database applications, a database user may contribute or modify some information in a database view instead of directly accessing the database instance, but then the contributed or modified information should be made valid also in the underlying database instance. To this end, the modifications in the database view must be *translated* to the database instance which is involved for relational databases, cf. [1, 9]. Moreover, a common database functionality is ensuring that a view update transaction adheres to the *ACID principles*. These principles require that either a transaction is committed as a whole or the previous database instance is restored (**A**tomicity); and further, after committing, the integrity constraints are preserved (**C**onsistency preservation). Here, we neglect the further requirements of **I**solation and **D**urability.
In our implementation of the multiagent scenario of Chapter 4, agent $\mathcal{A}$ may send a view update transaction request $\mathtt{up}(\mathcal{L})$ to $\mathcal{D}$ with literals $\mathcal{L} = \{L_1, \ldots, L_l\}$. A positive literal $a \in \mathcal{L}_{PL}$ requires the insertion of atom a into the database instance and a negative literal $\neg a \in \mathcal{L}_{PL}$ the deletion of atom a from the database instance, respectively. This means that $\mathcal{A}$ informs $\mathcal{D}$ that the value of the atom a has changed either to true or to false, respectively. Then, in return, agent $\mathcal{D}$ processes the request. In the following, we outline the processing for the case that agent $\mathcal{D}$ does not control its reaction for the sake of confidentiality:

1. filter out void update requests from the input literals in $\mathcal{L}$ and keep the *outstanding updates* in $\mathcal{U}$, that is, set $\mathcal{U} = \mathcal{L} \setminus \mathsf{Bel}_{\mathcal{D}}^{DB}(\langle \mathcal{IC}, \mathcal{db} \rangle)$;

2. check whether $\mathcal{U} = \emptyset$;

3. if not, check whether updating the database instance by $\mathcal{U}$ violates some integrity constraints;

4. in case of violation, notify agent $\mathcal{A}$;

5. else actually update the database instance and the database view $\langle \mathcal{IC}, \mathcal{C} \rangle$ by $\mathcal{U}$ and notify agent $\mathcal{A}$.

To implement the processing of a transaction, we introduce the database update operator $\bullet$ which translates the modifications of the view update operator $\odot$ on the database view to the instance. Both operators are defined in the sequel.

Definition 6.4 (Database Update $\bullet$).

Let $\langle \mathcal{IC}, \mathcal{db} \rangle$ be a database and $\mathcal{L}$ a set of literals over pairwise distinct atoms. Then, update of the database instance $\mathcal{db}$ by $\mathcal{L}$ is defined as

$$\mathcal{db} \bullet \mathcal{L} = (\mathcal{At} \cap \mathcal{L}) \cup (\mathcal{db} \setminus \{a \mid \neg a \in \mathcal{L}\}).$$

Further, update of the database $\langle \mathcal{IC}, \mathcal{db} \rangle$ by $\mathcal{L}$ is defined as

$$\langle \mathcal{IC}, \mathcal{db} \rangle \bullet \mathcal{L} = \begin{cases} \langle \mathcal{IC}, \mathcal{db} \bullet \mathcal{L} \rangle & \text{if } \mathcal{db} \bullet \mathcal{L} \models_{PL} \mathcal{IC}, \\ \langle \mathcal{IC}, \mathcal{db} \rangle & \text{otherwise.} \end{cases}$$

Usually, it should be possible to undo a previous update operation on the database or on the database view, cf. [9, 27]. However, in this thesis we will not treat this matter further, except for the basic Lemma 6.1 below which is needed for some of our results. For undoing updates by the set of literals $\mathcal{L}$, we invert these updates by updating with the set of literals $\neg(\mathcal{L})$. This set is formally defined, in general, for a set of formulas $\mathcal{M} \subseteq \mathcal{L}_{PL}$ as follows.

$$\begin{aligned} \neg(\mathcal{M}) :=& \{a \mid \neg a \in \mathcal{M}, a \in \mathcal{At}\} \cup \\ & \{\neg F \mid F \in \mathcal{M} \text{ and } F \text{ is not of the form } \neg a, a \in \mathcal{At}\} \end{aligned} \tag{6.3}$$

Note that, by definition, the set $\neg(\mathcal{L})$ is a set of literals.

Lemma 6.1 (Reversing Database Updates).

Let $\mathcal{U}$ be a set of literals over pairwise distinct atoms. Further let $\mathcal{db}$ be a database instance such that $\mathcal{db} \not\models_{PL} L$ for all $L \in \mathcal{U}$. Then, it holds that $\mathcal{db} = (\mathcal{db} \bullet \mathcal{U}) \bullet \neg(\mathcal{U})$.

A successful view update transaction may modify formulas in the instance view by means of the view update operator $\odot$ which bases on the operation of *variable negation* neg defined in the following. Essentially, the variable negation operator ensures that all formulas in the instance view become valid in the modified database instance and no information is lost. The correctness of the operator in this respect will be subject of Lemma 6.3 below.

Definition 6.5 (Variable Negation neg).
Cf. [22]

Let $\mathcal{M} \subseteq \mathcal{L}_{PL}$ be a set of formulas and $\mathcal{L}$ a finite set of literals over pairwise distinct atoms. Then, variable negation on $\mathcal{M}$ by $\mathcal{L}$ is defined as
$\mathsf{neg}(\mathcal{M}, \mathcal{L}) := \{\mathsf{neg}(F, \mathcal{L}) \mid F \in \mathcal{M}\}$ *where* $\mathsf{neg}(F, \mathcal{L})$ *is the result of replacing each occurrence of an atom $a \in \mathsf{At}(\mathcal{L})$ in F by $\neg a$ (here $\mathsf{At}(\mathcal{L})$ is the set of atoms occurring in L).*

We required literals in $\mathcal{L}$ instead of atoms for later convenience which by the following lemma makes no difference. The lemma shows two very basic properties of variable negation. The second property is important for properties of iterated database updates discussed in Section 7.3.

Lemma 6.2 (Properties of Variable Negation neg).

Let F be a formula and $\mathcal{X}$ and $\mathcal{Y}$ sets of literals each over pairwise distinct atoms. Then, it holds

1. $\mathsf{neg}(F, \mathcal{X}) = \mathsf{neg}(F, \mathsf{At}(\mathcal{X}))$,
2. $\mathsf{neg}(\mathsf{neg}(F, \mathcal{X}), \mathcal{Y}) = \mathsf{neg}(F, (\mathcal{X} \setminus \mathcal{Y}) \cup (\mathcal{Y} \setminus \mathcal{X}))$.

The view update operator $\odot$ introduced in the next definition adheres to the following requirements. First, necessarily, the instance view must be modified such that all formulas in the instance view become valid in the modified database instance, cf. Definition 6.2 of database views. Second, after these modifications no information in the database view must be lost. Third, as all literals in $\mathcal{L}$ become valid in the database instance after $\mathsf{up}(\mathcal{L})$, they must be added to the instance view.

Definition 6.6 (View Update $\odot$).

Let $\langle \mathcal{IC}, \mathcal{C} \rangle$ be a database view and $\mathcal{L}$ a finite set of literals over pairwise distinct atoms. Then, view update of the instance view $\mathcal{C}$ by $\mathcal{L}$ is defined as

$$\mathcal{C} \odot \mathcal{L} = \mathsf{neg}(\mathcal{C}, \mathcal{L}) \cup \mathcal{L} \qquad \textit{(1. and 3. requirement).}$$

Further, view update of the database view $\langle \mathcal{IC}, \mathcal{C} \rangle$ by $\mathcal{L}$ is defined as

$$\langle \mathcal{IC}, \mathcal{C} \rangle \odot \mathcal{L} = \langle \mathcal{IC}, (\mathcal{C} \cup \mathcal{IC}) \odot \mathcal{L} \rangle \qquad \textit{(2. requirement).}$$

The set $\mathcal{IC} \odot \mathcal{L}$ represents the information that the integrity constraints have been valid in the database instance before the update by $\mathcal{L}$.
That the above definition of the operator $\odot$ indeed satisfies the first and second requirement is verified by the following lemma. The lemma relates modifications of the database view by means of the view update operator $\odot$ via variable negation neg and modifications of the database instance by means of the database update operator $\bullet$.

Lemma 6.3 (Negation Equivalence).
Cf.[22]

Let db be a database instance, $\mathcal{M} \subseteq \mathcal{L}_{PL}$ a set of formulas, and $\mathcal{U}$ a set of literals over pairwise distinct variables such that $db \not\models_{PL} L$ for all $L \in \mathcal{U}$. Then, it holds that

1. $db \models_{PL} \mathcal{M}$ *iff* $db \bullet \mathcal{U} \models_{PL} \mathsf{neg}(\mathcal{M}, \mathcal{U})$, *and*

2. $db \models_{PL} \mathsf{neg}(\mathcal{M}, \mathcal{U})$ *iff* $db \bullet \mathcal{U} \models_{PL} \mathcal{M}$.

Especially, by definition of the database update $\bullet$ and the view update $\odot$ it follows that $db \models_{PL} \mathcal{M}$ iff $db \bullet \mathcal{U} \models_{PL} \mathcal{M} \odot \mathcal{U}$.[3]

6.2 Implementation with Nonmonotonic Reasoning

6.2.1 Agent $\mathcal{D}$'s Belief

The second exemplary implementation of agent $\mathcal{D}$ is outlined by Figure 6.2. Further, we illustrate the implementation in Section 6.2.2 with a running example. In inaccessible environments, the defender $\mathcal{D}$ faces generally incomplete information (a *finite* set of *current assertions* $as \subset_{fin} \mathcal{L}_{PL}$) about its environment. To guide its decisions, $\mathcal{D}$ must be capable to draw reasonable conclusions from the available information as. This capability is modeled by a *consequence relation* $\mid\!\sim \subseteq \mathcal{L}_{PL} \times \mathcal{L}_{PL}$ that maps assertions to conclusions

[3] The definition of operator $\odot$ may well be applied to arbitrary sets $\mathcal{M} \subseteq \mathcal{L}_{PL}$ without any change.

where we read $A \mathrel{|\!\sim} B$ as "B is a *defeasible conclusion* from A". Here, we focus on *nonmonotonic* consequence relations that *lack* the following property of *monotonicity*: For all $A, B, C \in \mathcal{L}_{PL}$ it holds that if $A \mathrel{|\!\sim} C$ then $A \wedge B \mathrel{|\!\sim} C$ (cf., e.g., [38] for a comprehensive introduction to nonmonotonic consequence relations). The consequence relation agent $\mathcal{D}$ uses for reasoning is a *fixed* instance of a class $\mathcal{C}$ of consequence relations and denoted by $\mathrel{|\!\sim}_{\mathcal{D}}$. Examples of such a class are consequence relations defined from plausibility structures such as preference orderings, ordinal conditional functions, possibility or plausibility spaces [50].
Gathering all conclusions from its current assertions, $\mathcal{D}$ forms its *belief* about the environment as follows:

$$\mathsf{Bel}^{NMR}_{\mathcal{D}}(\mathrel{|\!\sim}_{\mathcal{D}}, as) := \{B \in \mathcal{L}_{PL} \mid \bigwedge(as) \mathrel{|\!\sim}_{\mathcal{D}} B\}, \tag{6.4}$$

$$\text{where } \bigwedge(as) := \begin{cases} \bigwedge_{F \in as} F & \text{if } as \neq \emptyset, \\ \top & \text{otherwise.} \end{cases} \tag{6.5}$$

Here, the symbol $\top$ refers to a tautology whereas the symbol $\bot$ refers to a contradiction. Among $\mathcal{D}$'s belief, there is *unquestionable belief* that $\mathcal{D}$ is not willing to give up independently of the assertions as at hand so that unquestionable belief cannot be revised. In line with [77, 50], we represent the fact that $\mathcal{D}$ unquestionably believes A by $\neg A \mathrel{|\!\sim}_{\mathcal{D}} \bot$ and also say that $\neg A$ is *contradictory under* $\mathrel{|\!\sim}_{\mathcal{D}}$ or $\mathcal{D}$ *considers* $\neg A$ *contradictory*. Agent $\mathcal{D}$ should never hold assertions that conflict with unquestionable belief, or in other words, that $\mathcal{D}$ considers contradictory.[4] Thus, the pair $\langle \mathrel{|\!\sim}_{\mathcal{D}}, as\rangle$ should be an epistemic state according to the following definition.

Definition 6.7 (Epistemic State $\langle \mathrel{|\!\sim}, as\rangle$).
Cf. [72]

Let $\mathrel{|\!\sim}$ *be a consequence relation and* $as \subset_{fin} \mathcal{L}_{PL}$ *a finite set of formulas such that* $\bigwedge(as) \mathrel{|\!\not\sim} \bot$. *Then, the pair* $\langle \mathrel{|\!\sim}, as\rangle$ *is called an epistemic state.*

Further, we follow a common approach in the literature [74] and only study consequence relations that extend propositional entailment from the assertions as ((6.6) in the following list) and that do not depend on the syntax of

[4] Otherwise the agent might come to strange conclusions, cf. a discussion in [56].

premises ((6.7) in the list). These properties can be expressed in terms of the belief (6.4) induced by the consequence relation $\mid\!\sim_{\mathcal{D}}$ given assertions as as follows.

$$\text{If } \mathit{as} \vdash_{pl} F, \qquad \text{then } F \in \mathsf{Bel}_{\mathcal{D}}^{NMR}(\mid\!\sim_{\mathcal{D}}, \mathit{as}). \tag{6.6}$$

$$\text{If } \vdash_{pl} \bigwedge(\mathit{as}_1) \Leftrightarrow \bigwedge(\mathit{as}_2), \quad \text{then } \mathsf{Bel}_{\mathcal{D}}^{NMR}(\mid\!\sim_{\mathcal{D}}, \mathit{as}_1) = \mathsf{Bel}_{\mathcal{D}}^{NMR}(\mid\!\sim_{\mathcal{D}}, \mathit{as}_2). \tag{6.7}$$

6.2.2 Running Example

Overview of Supply Chain Scenario

Above all, we recall the application scenario from Section 1.2: Agent $\mathcal{D}$ on behalf of a manufacturer and agent $\mathcal{A}$ on behalf of supplier #1 are negotiating for supply contracts. The manufacturer has confidentiality interests towards supplier #1 which $\mathcal{D}$ is responsible to enforce. We will focus on the aspect of confidentiality in Section 8. Moreover, $\mathcal{D}$ is informed by $\mathcal{A}$ about the supply and possible deployment of needed parts. With this information, agent $\mathcal{D}$ reasons about the realizability of the manufacturer's product proposal. In this subsection, to illustrate this reasoning, we will represent $\mathcal{D}$'s fixed nonmonotonic consequence relation $\mid\!\sim_{\mathcal{D}}$ by an *ordinal conditional function* (OCF) [89]. To this end, we will simplify the supply chain scenario as follows. First, we focus on only one product proposal and assume that part #1 and part #2 and only those are needed for the product and there are only the two possible suppliers #1 and #2 for these parts. We consider the countably infinite alphabet $\mathcal{A}t = \{r, s11, d11, i11\_22, \ldots\}$ where r stands for "the product proposal is realizable", sij stands for "part i can be supplied by supplier j", dij for "part i from supplier j may be deployed in the product" and ikl_mn for "part k from supplier l and part m from supplier n are incompatible".[5] Especially, we consider that a supplier might be the manufacturer of the supplied part. Thus, the incompatibility of two parts may depend on the suppliers of those parts.
In the simplified scenario, agent $\mathcal{D}$ unquestionably believes $r \Leftrightarrow (d11 \vee d12) \wedge (d21 \vee d22)$ (abbreviated by U) which will be represented in

[5] To make the representation of $\mathcal{D}$'s reasoning concise in the following, we will not use a proposition saying that a part is needed for the proposed product.

its fixed reasoning as $\neg U \mathrel{|\!\sim}_{\mathcal{D}} \bot$. Finally, we consider that before the negotiation agent $\mathcal{D}$ agreed with the other part supplier #2 on the deployment of part #1. Thus, $\mathcal{D}$'s initial assertions are $\mathit{as} = \{d12, s12\}$.

Defining $\mathcal{D}$'s Reasoning by an Ordinal Conditional Function

Agent $\mathcal{D}$'s reasoning about the realizability of the product proposal should reflect the following two defeasible rules:

1. "Normally, if a part i is needed for the product and can be supplied by some supplier j then i from j may be deployed in the product" and
2. "Normally, if a part i from some supplier j and a part k from some supplier l are incompatible then part i from supplier j or part k from supplier l may not be deployed in the product".

In the simplified scenario, we use the following three simplified instantiations of these two rules:

1. "Normally, if part #2 can be supplied by supplier #1, then part #2 from supplier #1 may be deployed"; and likewise
2. "Normally, if part #2 can be supplied by supplier #2, then part #2 from supplier #2 may be deployed";
3. "Normally, if part #1 from supplier #2 and part #2 from supplier #1 are incompatible, then part #2 from supplier #1 may not be deployed".[6]

On the basis of these three instantiated rules, we will now construct $\mathcal{D}$'s nonmonotonic consequence relations by an ordinal conditional function (OCF). An OCF κ is a function $\kappa : \Omega \to \mathbb{N}_0 \cup \{\infty\}$ with $\kappa^{-1}(0) \neq \emptyset$ and defines *plausibility* of worlds (truth-value assignments) ranging from 0 (most plausible) to ∞ (totally implausible). The *rank* $\kappa(A)$ of a propositional formula A is defined by

$$\kappa(A) := \begin{cases} \underset{\omega\in\Omega:\omega\models_{PL}A}{min} \kappa(\omega) & \text{if } A \text{ is satisfiable,} \\ \infty & \text{otherwise.} \end{cases}$$

[6] Because the manufacturer has already agreed on the deployment $d12$ of part #1 from supplier #2.

rank	world						id	world						id
∞	$*$	$*$	$*$	$\neg d21$	$\neg d22$	r	$l_{\infty 1}$	$*$	$*$	$*$	$d21$	$\neg d22$	$\neg r$	$l_{\infty 2}$
	$*$	$*$	$*$	$\neg d21$	$d22$	$\neg r$	$l_{\infty 3}$	$*$	$*$	$*$	$d21$	$d22$	$\neg r$	$l_{\infty 4}$
3	$*$	$*$	$s22$	$d21$	$\neg d22$	r	l_{31}	$*$	$*$	$s22$	$\neg d21$	$\neg d22$	$\neg r$	l_{32}
2	$i12\_21$	$*$	$\neg s22$	$d21$	$*$	r	l_{21}	$i12\_21$	$*$	$s22$	$d21$	$d22$	r	l_{22}
1	$i12\_21$	$s21$	$\neg s22$	$\neg d21$	$d22$	r	l_{11}	$*$	$s21$	$s22$	$\neg d21$	$d22$	r	l_{12}
	$i12\_21$	$s21$	$\neg s22$	$\neg d21$	$\neg d22$	$\neg r$	l_{13}							
	$\neg i12\_21$	$s21$	$\neg s22$	$\neg d21$	$d22$	r	l_{14}	$\neg i12\_21$	$s21$	$\neg s22$	$\neg d21$	$\neg d22$	$\neg r$	l_{15}
0	$i12\_21$	$\neg s21$	$\neg s22$	$\neg d21$	$d22$	r	l_{01}	$i12\_21$	$\neg s21$	$s22$	$\neg d21$	$d22$	r	l_{02}
	$i12\_21$	$\neg s21$	$\neg s22$	$\neg d21$	$\neg d22$	$\neg r$	l_{03}							
	$\neg i12\_21$	$s21$	$\neg s22$	$d21$	$*$	r	l_{04}	$\neg i12\_21$	$s21$	$s22$	$d21$	$d22$	r	l_{05}
	$\neg i12\_21$	$\neg s21$	$\neg s22$	$*$	$d22$	r	l_{06}	$\neg i12\_21$	$\neg s21$	$s22$	$*$	$d22$	r	l_{07}
	$\neg i12\_21$	$\neg s21$	$\neg s22$	$d21$	$*$	r	l_{08}							
	$\neg i12\_21$	$\neg s21$	$\neg s22$	$\neg d21$	$\neg d22$	$\neg r$	l_{09}							

Table 6.1: The OCF κ_1 on the relevant atoms for illustrating $\mathfrak{D}$'s fixed reasoning $\mid\!\sim_{\mathfrak{D}}^{\kappa_1}$ (only specified for the worlds satisfying $d12 \wedge s12$)

In the running example, we define the consequence relation $\mid\!\sim_{\mathfrak{D}}$ by the OCF κ_1 in Table 6.1. For each world, we only write down the truth-value assignment of atoms as needed for illustrating $\mathfrak{D}$'s defeasible reasoning about r in the simplified scenario. An assignment of atom x to *true*/*false* is denoted by x and $\neg x$ respectively. The symbol $*$ stands for an arbitrary assignment of the corresponding atom and each (set of) worlds has an identifier l_{ij} on its right for later reference where i corresponds to the rank and j is a consecutive number within rank i. The OCF κ_1 represents $\mathfrak{D}$'s unquestionable belief of $r \Leftrightarrow (d11 \vee d12) \wedge (d21 \vee d22)$ by ranking the models of its negation as totally implausible (∞).
With the plausibility of worlds given by an OCF κ in mind, essentially, $\mathfrak{D}$ defeasibly concludes a formula B from a formula A iff any world that *falsifies the conclusion* of B from A (that is, a model of $A \wedge \neg B$) is less plausible than some world that *justifies the conclusion* (that is, a model of $A \wedge B$):

$$A \mid\!\sim^{\kappa} B \text{ iff } \kappa(A) = \infty \text{ or } \kappa(A \wedge B) < \kappa(A \wedge \neg B). \tag{6.8}$$

We denote $\mathfrak{D}$'s fixed reasoning defined by κ_1 in Table 6.1 by $\mid\!\sim_{\mathfrak{D}}^{\kappa_1}$. Table 6.1 is presented in the way that worlds with equal rank which either falsify or justify the conclusion of one of the three instantiated rules or do neither, are grouped together as closely as possible. Considering these rules, we see that in its reasoning $\mid\!\sim_{\mathcal{D}}^{\kappa_1}$ agent $\mathcal{D}$ applies these rules to the three situations $s21$, $s22$ and $i12\_21$ that are explicitly the premises of these rules:

$$s21 \mid\!\sim_{\mathcal{D}}^{\kappa_1} d21 \qquad s22 \mid\!\sim_{\mathcal{D}}^{\kappa_1} d22 \qquad i12\_21 \mid\!\sim_{\mathcal{D}}^{\kappa_1} \neg d21. \tag{6.9}$$

In the following, we will explain why $s21 \mid\!\sim_{\mathcal{D}}^{\kappa_1} d21$ holds indeed. To draw conclusions from the assertion $s21$, $\mathcal{D}$ only inspects the most plausible models of $s21$ (which are given by l_{04} and l_{05}). By (6.8), $\mathcal{D}$ concludes that part #2 from supplier #1 can be deployed ($d21$) from $s21$ if and only if $s21$ has rank ∞ or for all most plausible models ω of $s21$ it holds $\omega \models_{PL} d21$ which is the case for both l_{04} and l_{05} indeed. The remaining relations of (6.9) can be verified the same way.

Finally, we explain how $\mathcal{D}$ generally applies the three rules to arbitrary assertions by describing the construction of the OCF κ_1. Altogether, we determined a *C-representation*[7] [70] from the three instantiated rules. But since the C-representation is not uniquely defined by these rules, we defined the rank of worlds with equal *conditional structure*[7] [70] in the following way. Above all, we of course ensure Equation 6.9 like in [70]. But further, we ensure that an instantiated rule could be applied to the current assertions as long as its conclusion does not contradict the conclusion of another applicable rule. For example, that way, we obtain that $s22 \wedge i12\_21 \mid\!\sim^{\kappa_1} d22 \wedge \neg d21$ holds (see l_{02}). If $s21$ and $i12\_21$ are among the current assertions, then the two rules "Normally, if $s21$ then $d21$" and "Normally, if $i12\_21$ then $\neg d21$" might be applied, but their conclusions contradict each other. In this case, we give priority to the latter rule because for deployment of parts their incompatibility must be accounted for, despite the availability of the parts

[7] We summarize the idea behind a C-representation in simple terms. We can describe the influence of a defeasible rule "Normally, if A then B" with $A, B \in \mathcal{L}_{PL}$ on a world ω. First, the rule renders ω less plausible if the rule is falsified by ω, that is $\omega \models_{PL} A \wedge \neg B$. Second, the rule renders ω more plausible if the rule is verified by ω, that is $\omega \models_{PL} A \wedge B$. At last, the rule does not influence ω at all if ω is indifferent towards the rule, that is $\omega \models_{PL} \neg A$. Given a set of defeasible rules, a C-representation of these rules assigns the same rank to every two worlds that are influenced by these rules in the same way (in formal terms, that have the same *conditional structure*).

from suppliers. With this choice, the following relations hold in $\mathcal{D}$'s consequence relation $\mid\!\sim_{\mathcal{D}}^{\kappa_1}$: $s22 \wedge s21 \wedge i12\_21 \mid\!\sim^{\kappa_1} d22 \wedge \neg d21$ (see l_{12}) and $s22 \wedge s21 \wedge i12\_21 \not\mid\!\sim^{\kappa_1} d21$ (see l_{12}).

6.2.3 The Agents' Interaction

Implementing agent $\mathcal{D}$ with a nonmonotonic consequence relation, cf. Figure 6.2, we restrict the interaction interface agent $\mathcal{D}$ provides to the requesting agent $\mathcal{A}$. The restriction should simplify our study of confidentiality. Via the interface, agent $\mathcal{A}$ may send *query* and *revision requests* to agent $\mathcal{D}$ defined by the set $\mathcal{R}eq^{NMR}$ of valid request messages:

$$\mathcal{R}eq^{NMR} = \{\texttt{que}(A) \mid A \in \mathcal{L}_{PL}\} \cup \{\texttt{rev}(A) \mid A \in \mathcal{L}_{PL}\}. \tag{6.10}$$

We discuss an extension of this interface in Section 6.4. In the remainder of this subsection, we detail how agent $\mathcal{D}$ reacts to agent $\mathcal{A}$'s requests without controlling its reactions for the sake of confidentiality.
On a query request $\texttt{que}(A)$, agent $\mathcal{D}$ returns the result of the ordinary query evaluation $\texttt{eval}(A)(\mid\!\sim_{\mathcal{D}}, \mathit{as})$, generally defined as follows.

Definition 6.8 (Query Evaluation).

Let $\langle \mid\!\sim_{\mathcal{D}}, \mathit{as}\rangle$ be an epistemic state and $A \in \mathcal{L}_{PL}$. Then, the query evaluation function is defined by

$$\texttt{eval}(A)(\mid\!\sim_{\mathcal{D}}, \mathit{as}) := \begin{cases} \mathit{true} & \mathit{if}\ A \in \mathsf{Bel}_{\mathcal{D}}^{NMR}(\mid\!\sim_{\mathcal{D}}, \mathit{as}), \\ \mathit{false} & \mathit{if}\ \neg A \in \mathsf{Bel}_{\mathcal{D}}^{NMR}(\mid\!\sim_{\mathcal{D}}, \mathit{as}), \\ \mathit{undef} & \mathit{otherwise.} \end{cases}$$

Via a revision request $\texttt{rev}(A)$ with formula $A \in \mathcal{L}_{PL}$, agent $\mathcal{A}$ provides $\mathcal{D}$ with information about an environment that does not evolve during the interaction [69]. Without controlling its reaction for the sake of confidentiality, on the request $\texttt{rev}(A)$ agent $\mathcal{D}$ first checks whether it unquestionably believes that the assertions A and as cannot hold at the same time, that is, whether the conjunction of these assertions is contradictory under $\mid\!\sim_{\mathcal{D}}$. Only if the conjunction of the assertions is not contradictory under $\mid\!\sim_{\mathcal{D}}$, agent $\mathcal{D}$ sets its epistemic state to the result of the revision operation $\langle \mid\!\sim_{\mathcal{D}}, \mathit{as}\rangle \circ A$ as follows.

Definition 6.9 (Revision $\circ$).
Cf. [12]

Let $\langle \mathrel{|\!\!\sim}_{\mathcal{D}}, as\rangle$ be an epistemic state and $A \in \mathcal{L}_{PL}$. Then, revision of $\langle \mathrel{|\!\!\sim}_{\mathcal{D}}, as\rangle$ by A is defined as

$$\langle \mathrel{|\!\!\sim}_{\mathcal{D}}, as\rangle \circ A = \begin{cases} \langle \mathrel{|\!\!\sim}_{\mathcal{D}}, as \cup \{A\}\rangle & \textit{if } \bigwedge(as) \wedge A \mathrel{|\!\!\not\sim}_{\mathcal{D}} \bot, \\ \langle \mathrel{|\!\!\sim}_{\mathcal{D}}, as\rangle & \textit{otherwise.} \end{cases}$$

Finally, $\mathcal{D}$ reacts by notifying $\mathcal{A}$ whether $\mathcal{D}$'s epistemic state has been revised as follows:

$$\texttt{notify}(A)(\mathrel{|\!\!\sim}_{\mathcal{D}}, as) := \begin{cases} \textit{success} & \text{if } \bigwedge(as) \wedge A \mathrel{|\!\!\not\sim}_{\mathcal{D}} \bot, \\ \textit{failure} & \text{otherwise.} \end{cases}$$

Here, we chose to let $\mathcal{D}$ throw away the new information A in case of a failure. With this approach, $\mathcal{D}$'s notification to $\mathcal{A}$ after a revision can only reveal whether $\mathcal{D}$ considers $\bigwedge(as) \wedge A$ contradictory or not. Generally, $\mathcal{D}$ should merge the new information with the previously gathered information but then it must control the merging process for preserving confidentiality. In Section 6.4, we discuss how to extend agent $\mathcal{D}$'s functionality including merging and updating evidential knowledge (here as) with additional information.

6.3 Related Work

This section should put the two exemplary implementations of agent $\mathcal{D}$ into the broader context of related work. Several of these related works suggest how to extend agent $\mathcal{D}$'s functionality under various aspects. Others indicate changes to be made in the design of agent $\mathcal{D}$. These aspects are summarized by Figure 6.3. Several of these aspects are both treated by the database and artificial intelligence research community. Thus, we focus on differences and commonalities in these two research areas so that in Section 6.4 we may sketch first steps towards a combination of the two exemplary implementations. However, implementing these steps is still future work.
This section is organized along the aspects shown in Figure 6.3. Note that this

section should not give an overview over the mentioned research areas, but rather point out relevant aspects for an extended implementation of agent $\mathcal{D}$.

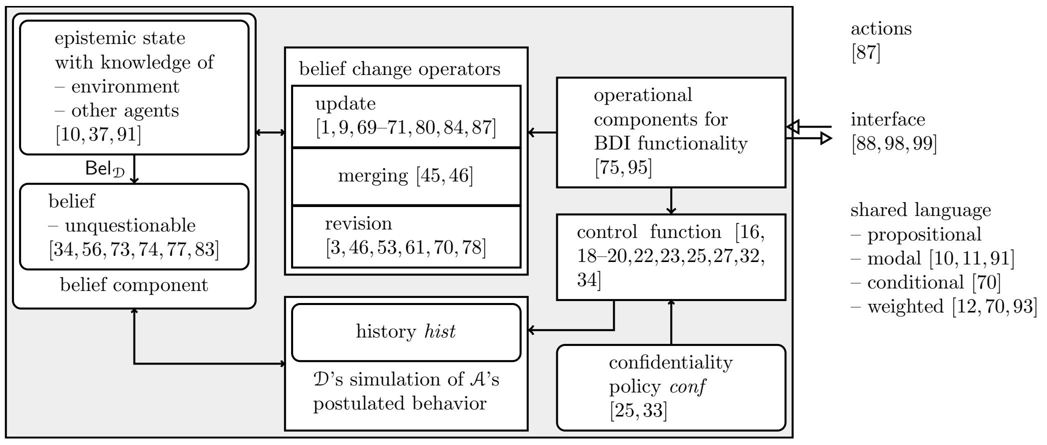

epistemic state with knowledge of
– environment
– other agents
[10, 37, 91]
$\mathsf{Bel}_{\mathcal{D}}$
belief
– unquestionable
[34, 56, 73, 74, 77, 83]
belief component
belief change operators
update [1, 9, 69–71, 80, 84, 87]
merging [45, 46]
revision [3, 46, 53, 61, 70, 78]
operational components for BDI functionality [75, 95]
control function [16, 18–20, 22, 23, 25, 27, 32, 34]
history *hist*
$\mathcal{D}$'s simulation of $\mathcal{A}$'s postulated behavior
confidentiality policy *conf* [25, 33]
actions [87]
interface [88, 98, 99]
shared language
– propositional
– modal [10, 11, 91]
– conditional [70]
– weighted [12, 70, 93]

Figure 6.3: Design of a BDI-agent $\mathcal{D}$ as defender with extended functionality

Interface Generally, a BDI-agent may not only share information with other agents explicitly like in thesis, but also perceive the action performed by another agent or perceive its environment by sensors. Moreover, the agent initiates actions itself. Perceiving the performance of an action makes the agent update its belief. We illustrate this aspect by examples from the literature in the subsequent paragraph about updating belief.

In order to share information with other agents, agent $\mathcal{D}$ may offer various types of interaction and associate a specific communication language to each type. The interaction interface also has relevance to confidentiality. On the one hand, it is the attacker $\mathcal{A}$'s means for actively probing agent $\mathcal{D}$'s epistemic state. Thus, restricting the types of interaction or the communication language implies restricting agent $\mathcal{A}$'s means as an attacker. On the other hand, agent $\mathcal{D}$ may want to weaken an answer to a request to enforce its confidentiality aims which might be more cooperative than a refusal of the request. Thus, the communication language should provide the agent with means to weaken information appropriately.

In the following, we will survey several examples of communication languages. In a *modal language* [10, 11, 91], agent $\mathcal{D}$ is able to make statements about its belief, for example, $\mathsf{B}r \vee \mathsf{B}\neg r$ expresses that manufacturer $\mathcal{D}$ has a result of the feasibility study, but not whether it is positive (r) or negative ($\neg r$). This way, agent $\mathcal{D}$ may weaken statements about its belief, benefiting from the richer expressivity of the modal language compared to the propositional language $\mathcal{L}_{PL}$. Agent $\mathcal{D}$ may also receive information expressed as sentences in the modal language and change its epistemic state with that information [11].

In a *conditional language* [70], agents may share (defeasible) rules which the receiver may integrate into its background or even its evidential knowledge. In the latter case, the receiver just wants to apply the additional rules in the case it reasons about like the feasibility study or the treatment of a particular patient. By way of contrast, background knowledge includes rules the agent applies in any case it reasons about. The reader may find examples of defeasible rules in Subsection 6.2.2.

In a language with *weights* [12, 70, 93], agent $\mathcal{D}$ may express its confidence in a belief by attributing a weight to this belief. The confidence (or certainty) in a belief of formula B is the *degree of disbelief* in the complement $\neg B$. For example, agent $\mathcal{D}$ as the manufacturer of Section 6.2.2 may express its confidence in the realizability of the product proposal by the rank $\kappa_1(\neg r)$. In general, the higher the rank is the higher the agent's confidence is. In Table 6.1, we see that the rank $\kappa_1(\neg r)$ is 0. This means that without any assertion agent $\mathcal{D}$'s confidence in the realizability of the proposal is low so that $\mathcal{D}$ does not believe that the proposal is realizable: $r \notin \mathsf{Bel}_{\mathcal{D}}^{NMR}(\mid\!\sim_{\mathcal{D}}^{\kappa_1}, \emptyset)$. If agent $\mathcal{D}$ held the assertion $d21$ that part #2 from supplier #1 may be deployed, then its confidence in the realizability of the proposal would be $\kappa_1(\neg r \wedge d21) - \kappa_1(d21) = \infty - 0 = \infty$.[8] Wiese in [93] defines confidentiality policies as a set of possibilistic formulas (S, α) with $S \in \mathcal{L}_{PL}$ and $\alpha \in (0, 1]$ for query-answering to a *possibilistic belief base*. Here, confidence in a belief B is the necessity degree α that is the maximal possibility of a model of $\neg B$. A formula (S, α) in the policy means that the attacker should not know that S is believed with confidence greater than α. Confidentiality is preserved by weakening answers: An answer (B, α) means that α is a lower bound to the necessity degree of B in the belief base.

[8] In Table 6.1, the models of $\neg r \wedge d21$ are given by $l_{\infty 2}$ and $l_{\infty 4}$.

BDI Functionality Since BDI-agents are *autonomous* in their decision-making, the control function should be used by agent $\mathcal{D}$ during its process of decision-making, cf. the discussion in Section 5.2 on the basis of [75, 95].

Control Function The line of work of Controlled Interaction Execution (CIE), cf. a survey in [16, 18], presents control functions to compute reactions of a server with a logic-oriented information system to requests of a client under confidentiality requirements like those of Chapter 4. The work deals with different parameters of the server some of which are part of the aspects of the defender's functionality in Figure 6.3. First, the *interface* of the server to the client may offer different kind of query requests like open or closed queries in an appropriate language, for example, open or closed formulas of *first order logic* [20]. Further, the server may offer elementary view updates [27] or view update transactions [22] in an appropriate language which to-date allows only propositional literals. The server may further restrict the first order language for query requests in order to reduce the costly simulation of the attacker (in CIE, the client) to simple pattern matching between the query and sentences of the confidentiality policy [25, 23]. To this end, the data model must also be restricted sufficiently. Second, the information system of the server may be based on different logic-oriented *data models* including complete propositional databases [19], incomplete propositional databases [34], relational databases [20, 23], that is, complete first order databases and incomplete first order databases [32]. Third, the control function may have different *options to enforce confidentiality*. Detecting a potential of confidentiality violation, the control function may *lie* by deviating from the functionally expected reaction, for example, by returning the negation of a query evaluation result instead of the correct answer. It is, however, crucial that the client cannot distinguish between a lie and a correct reaction. Another option at the potential of confidentiality violation is that the control function *refuses* a reaction to the request altogether by returning a refusal notification. The third option is combining lying and refusal. The differences between lying and refusal is outlined in [16, 18]. The surveys [16, 18] explain further parameters of CIE.

Confidentiality Policy See the discussion in Section 5.2.

Belief Update Updates have both been studied in the artificial intelligence research community, for example in [69, 70, 12, 65, 87, 84, 63], and in the database research community, for example in [1, 9, 84]. The work we present here is only a small selection of the work in both communities. This selection should make clear the relation between revision and update and between belief update in AI research and traditional database update.

In the seminal work of [69], Katsuno and Mendelzon highlight the fundamental differences between belief updates and belief revision. They consider an agent $\mathcal{X}$ that is in the following situation. The agent has a *knowledge base*[9] $\mathcal{kb} \subseteq \mathcal{L}_{PL}$ and receives an additional piece $A \in \mathcal{L}_{PL}$ of information that is inconsistent with $\mathcal{kb}$. Its knowledge base enables the agent to distinguish between *possible worlds*, all models of $\mathcal{kb}$, and impossible worlds. In the considered situation, the agent $\mathcal{X}$ may interpret the piece A of information differently depending on the context. First, $\mathcal{X}$ may interpret the formula A as evidence conflicting with what the agent learnt so far about a static environment. In this case, the evidence A indicates "the retroactive impossibility of [some world]" [69] and the agent would revise $\mathcal{kb}$ with A. Katsuno and Mendelzon explain the concept of revision as follows. "Since the real world has not changed, and that fact [A] has to be true in all the new possible worlds, we can forget some of the old possible worlds on the grounds that they are too different from what we now know to be the case."

Second, the agent $\mathcal{X}$ may interpret the formula A as information about a change in its environment. In this case, "the fact that the real world has changed gives us no grounds to conclude that some of the old worlds where actually not possible." Therefore, the agent should "find a minimal way of changing each [of the old possible worlds] so that it becomes a model of that new fact [A]" [69]. Thus, the agent updates $\mathcal{kb}$ with A. As a seminal work, Katsuno and Mendelzon give requirements for update operators as rationality postulates complementing the well-known *AGM postulates* for belief revision operators [3].

In the work by Katsuno and Mendelzon an agent processes propositional

[9] A knowledge base is a consistent set of propositional formulas.

information only. Kern-Isberner in [70, 71] considers that an agent may incorporate both propositions and additional (defeasible) rules into its background and evidential knowledge. The essential difference between update and revision pointed out by Kern-Isberner is as follows. Iterated revision, where the agent receives a sequence of pieces of additional information (propositions or rules), may be processed in a single step revision by collecting all the information received sequentially. In particular, the agent does not discard a piece of information received in that sequence. In contrast, the agent may not process iterated updates in a single step. The reason is that a piece of information interpreted as an update may override other information received so far.

Another line of research in artificial intelligence focuses on how an agent should update its epistemic state due to some specific *action* the agent perceived or performed itself, for example, in [80, 87, 84]. How the action affects the environment is well-known to the agent. For example, an action might be to open a door. Thus, the ramifications of the action are specified as well as facts that are invariant under that action. Then, the agent uses the specification to reason about actions. The formalism for the specification is the *situation calculus* of [80]. In this thesis, we study a different kind of an agent's perceptions so that this line of research is beyond the scope of this thesis. However, an agent should have the ability to process both kinds of perceptions. The following two examples of research give ideas how to do that. Shapiro et al in [87] extend the situation calculus with a representation of an agent's belief similar to the representation of Section 6.2. With that extension, the effects of an action on the agents's belief can be specified. The authors distinguish between actions that do not affect the environment (like a sensing action) and actions that do so presenting specific properties for each kind of action in the extended situation calculus. The effects of actions of the former kind on the agents' belief satisfy the AGM postulates for belief revision, whereas the latter satisfy the Katsuno/Mendelzon postulates for belief update. Reiter in [84] uses the situation calculus for specifying update transaction of databases. In his work, a database essentially is a first order theory where each formula is parameterized with the database state in which it holds. A transaction is an action in the sense of the situation calculus and leads to a transition of the database state adhering to axioms defining effects and invariants with the usual formalism of the situation calculus.[10] Thus, the database supports only some predefined set of transactions.

The formal semantics of view update transactions have further been studied in other research from the database community, cf. [1] for an introduction. A seminal work in this community is by Bancilhon and Spyratos in [9]. Their work presents a solution to the "view update problem" in relational databases: The definition of a view (that is a database query) does generally not suffice to translate a view update to the database instance uniquely. The authors propose to add a *complement view* to the view definition. The complement view has to be invariant when an update of the view is translated to the database instance. This means that the evaluation of the complement view in the instance before and after the translation is the same. Then, this requirement uniquely defines the translation of the view update.

Belief Revision Pioneering the work of Katsuno and Mendelzon [69], the seminal works on belief revision study how an agent $\mathcal{X}$ with a consistent knowledge base $\mathcal{kb} \subseteq \mathcal{L}_{PL}$ may incorporate an additional piece $A \in \mathcal{L}_{PL}$ of information into its knowledge base with a *revision operator* $\star$ (cf. the surveys in [61] and in [70, Chapter 2]). In this situation, agent $\mathcal{X}$ might have to solve the problem that the new evidence A contradicts its knowledge base, that is, $\mathcal{kb} \vdash_{pl} \neg A$ holds, but – for some reasons – the agent does not want to drop the evidence A. Then, the agent has to give up some pieces of information included in $\mathcal{kb}$ so that afterwards it might add the new evidence to its remaining knowledge base $\mathcal{kb}'$ consistently (that is, its knowledge base becomes $\mathcal{kb}' \cup \{A\}$). In this process, agent $\mathcal{X}$ has to decide which information should be given up. The agent commits to these decisions at the time it chooses its revision operator $\star$. The literature defines postulates for properties of revision operators which follow a *principle of minimal change* of the agent's knowledge base, cf. the surveys in [61, 70] and the fundamental postulates of AGM in [3]. Further, revision operators may be constructed on the basis of other information like an *entrenchment order* on formulas where the agent

[10] These axioms make use of second order logic. Because the database includes these axioms, it is not a pure first order theory.

prefers to give up a formula over another higher entrenched formula [61].

In Section 6.2, we viewed belief revision as the agent's process of nonmonotonic reasoning from its current assertions (that is, its evidential knowledge). The literature brings together the two lines of research on belief revision as characterized by the AGM postulates and related work, and nonmonotonic reasoning. The reader may find references of this literature in the survey of [70, Chapter 2]. The postulates refer to revision operators that process a knowledge base and an additional piece of information $A \in \mathcal{L}_{PL}$. Within Section 6.2, agent $\mathcal{D}$'s knowledge base $\mathcal{kb}$ is the set $\mathsf{Bel}_{\mathcal{D}}^{NMR}(\mathrel{|\!\sim}_{\mathcal{D}}, \mathcal{as})$. Agent $\mathcal{D}$'s processing of new evidence $A \in \mathcal{L}_{PL}$ by the revision operator $\circ$ of Definition 6.9 defines a revision operator $\star$ on its knowledge base. The operator outputs the knowledge base $\mathcal{kb} \star A$ defined by the relation $B \in \mathcal{kb} \star A = \mathsf{Bel}_{\mathcal{D}}^{NMR}(\langle \mathrel{|\!\sim}_{\mathcal{D}}, \mathcal{as}\rangle \circ A)$ iff $\mathcal{as} \wedge A \mathrel{|\!\sim}_{\mathcal{D}} B$ (given that $\mathcal{as} \wedge A$ does not contradict $\mathcal{D}$'s unquestionable belief). This relation may be understood as follows [46] "The AGM postulates of belief revision are in a sense written from a purely external point of view, as if an observer had access to the agent's belief set [here: knowledge base[11]] from outside, would notice its evolution under input information viewed as stimuli, and describe its evolution laws." Here, these evolution laws are mainly determined by $\mathcal{D}$'s fixed nonmonotonic reasoning $\mathrel{|\!\sim}_{\mathcal{D}}$.

The *Ramsey test* further highlights the connection between nonmonotonic reasoning and belief revision, cf. the survey in [61]. It gives semantics to rules like "If $\neg i12\_21$ was true (the incompatibility had been resolved), then r would be true (the proposal would be realizable)" in the agents' dialog of Section 1.2 called counterfactual conditionals in the literature. The semantics of the latter rule according to the Ramsey test is that after the agent revised its epistemic state with $\neg i12\_21$ it would believe r.

Beyond these two views on an agent's belief revision process where "belief revision and nonmonotonic reasoning are two sides of the same coin" [53, 78], Dubois in [46] points out yet another two different interpretations of a belief revision process from the literature. First, the evidence acquired by an agent might be uncertain or unreliable to different degrees. In this situation, the agent may merge all pieces of evidence according to their reliability and

[11]A belief set is a set of propositional formulas closed under propositional entailment.

certainty and, this way, remove contradicting evidence. The agent may merge its evidence first and then reason from the merged evidence with its nonmonotonic consequence relation. In the subsequent paragraph about merging evidence, we will briefly outline the difference between belief revision as merging and belief revision as nonmonotonic reasoning.
Second, background knowledge given by a consequence relation (or its defining structure like an OCF as in Subsection 6.2.2) may be revised by additional defeasible rules, cf. the approach of Kern-Isberner in [70].

Merging Evidence Delgrande et al in [45] propose that an agent orders its evidential knowledge, for example, by the credibility of the sources of information or by the recency of the information. The order determines the priority that a piece of evidence takes over another one when the agent must decide between those two pieces of evidence to resolve an inconsistency. The evidential knowledge thus is represented as a list of multisets so that a multiset with a lower index in the list has less priority. The agent processes its evidential knowledge with a merging operator $\circledast$ which removes inconsistencies from the evidence and outputs a propositional formula. Delgrande et al propose postulates characterizing merging operators. One intuitive postulate says that the processing of pieces of evidence is not influenced by evidence with lower priority. Another intuitive postulate is that consistent evidence is merged to the conjunction of all pieces of evidence.

Other Types of Interaction The types of interactions that we have surveyed so far serve the purpose of information sharing in general. Those general types of interactions are processed by respective belief operators that we have partly illustrated in the previous paragraphs. However, several applications of intelligent agents might require more specific types of interaction. For example, before the agents may collaborate pursuing a joint goal, the agents first have to agree which goal to achieve. To this end, the agents are initially involved in a *negotiation*. As another example, the agents may settle other agreements in a negotiation like a meeting [24]. We selected two examples [98, 99] to illustrate research on negotiation in the context of our work. In a negotiation, each agent has demands which may be a set of propositional formulas [98] such as its most preferred times for the meeting or even a *preference order* over propositions [99]. The outcome of a negotiation in general is a subset of the agents' demands that is accepted by every agent. The means of negotiation proposed by [98, 99] are belief revision operators. The idea is that an agent revises its own demands with possibly conflicting demands of other agents it accepts in an agreement. Research on negotiation in artificial intelligence is related to other fields such as game theory and argumentation, cf. [98].

In other applications, where intelligent agents should support human decision-makers, the agent should not only automatically suggest a decision, but also give an explanation to be judged by the human agent. *Argumentation systems* [88] present formalisms for providing explanations that are easily followed by humans. According to the overview in [88], argumentation is a "form of reasoning where the conclusion and the way to arrive at it are doubted and effectively challenged." In dialogical argumentation, an agent has to defend its claim with supporting arguments towards an opposing agent that raises counter-arguments against the claim or its support. Complementary, an agent may use internal argumentation to establish its belief. The outcome of the process of argumentation may be defined by different semantics and generally decides which arguments are accepted and which are rejected. A claim is justified and explained to a human agent by a process of argumentation that leads to acceptance of the claim.

Integrity Constraints and Unquestionable Belief In the two exemplary implementations, the concept of integrity constraints on the one hand and the concept of unquestionable belief on the other hand define which evidential knowledge agent $\mathcal{D}$ may accept. In this paragraph, we relate the two concepts by the literature [74, 77, 13, 73, 56, 83]. The concept of unquestionable belief follows the work of Lehmann and Magidor [74, 77] who define a *hard constraint* $A \in \mathcal{L}_{PL}$ of an agent's belief as the constraint $\neg A \mathrel{|\!\!\sim} \bot$ on the agent's consequence relation. The constraint means that the agent only reasons about models of A: The structure defining the consequence relation ranges only over models of A similar to the representation of the constraint by ordinal condition functions in Subsection 6.2.2. Lehmann and

Magidor's semantics of unquestionable belief essentially allows the agent to reason with unquestionable belief by propositional entailment – at least if its consequence relation is defined by Lehmann and Magidor's ranked models or ordinal conditional functions.

Konieczny [73] extends this semantics to several levels of belief where a higher level constitutes the integrity constraints of the lower levels. For example, facts about a patient, medical expertise and arithmetics form a hierarchy of three levels of belief from low to high. Levels of beliefs are represented by transfinite ordinals in the range of an ordinal conditional function where the first transfinite ordinal ∞ marks the boundary between level one and two (in the example, facts about a patient and medical expertise).

In database research, Reiter [83] proposes to formalize integrity constraints for *incomplete first-order databases* in a fragment of first order modal logic. He argues that integrity constraints are statements about the state of the database instance and such statements require a modality saying "the database knows". In contrast, database instances as first order theories makes statements about the state of the world. In the context of incomplete propositional databases, the modal logic S5 might be the right choice to formalize integrity constraints [34, 48]. The semantics of the integrity constraints are S5 Kripke structures which separate the set of propositional interpretations into the set of models of the database instance (a propositional theory[12]) and the complement of this set. This way, the integrity constraints define propositional theories that may be used as the database instance. In a similar way, the hard constraints by Lehmann and Magidor define what assertions (also a propositional theory) the agent may hold.

Finally, we mention the alternative semantics of unquestionable belief by Goldszmidt and Pearl [56]. They argue that a strict rule $A \Rightarrow B$ should not be identified with the hard constraint $\neg A \vee B$ (that is, material implication) as Lehmann and Magidor do. Conversely, the two rules $A \Rightarrow B$ and $A \Rightarrow \neg B$ should be contradictory, not yield the conclusion of $\neg A$. Contradictions according to Goldszmidt and Pearl's semantics thus expose flaws in the modeling of background knowledge by strict and defeasible rules. However, the modeling of background knowledge by rules is not the focus of this thesis.

[12]Here, a propositional theory is a finite consistent set of propositional formulas.

Therefore, we neglect such aspects since we aim at a simple representation of unquestionable belief in a consequence relation $\mid\!\sim \subset \mathcal{L}_{PL} \times \mathcal{L}_{PL}$.

Knowledge and Reasoning About Other Agents In this thesis, $\mathcal{D}$'s simulation of $\mathcal{A}$'s postulated reasoning is the key to $\mathcal{D}$'s enforcements of its confidentiality interests. As an attacker, agent $\mathcal{A}$ reasons about $\mathcal{D}$'s process of forming its belief. $\mathcal{D}$'s simulation of that reasoning might be based on approaches in the literature on an agent's reasoning (here $\mathcal{A}$'s) about other agents (here $\mathcal{D}$).
In the following, we survey selected approaches [91, 10, 37] from the perspective of our focused scenario of the reacting defender $\mathcal{D}$ and the requesting attacker $\mathcal{A}$. Especially, we point out how each approach considers important aspects of the focused scenario, that is, the interaction between the agents (1), the belief change operators applied by $\mathcal{D}$ (2), the structure of $\mathcal{D}$'s epistemic state (3) and the model of $\mathcal{A}$'s reasoning about $\mathcal{D}$ (4). This way, we may judge whether an approach is appropriate for $\mathcal{D}$'s simulation of $\mathcal{A}$ in the context of one of the implementations of agent $\mathcal{D}$.
In the field of *dynamic epistemic logic*, Van Benthem [91] describes how the epistemic state of each agent in a multiagent system evolves after *public announcements* of propositional or epistemic formulas to all agents. The epistemic state of each agent is an ordering of worlds by plausibility and depends on the world the agents are actually situated in. An agent's epistemic state involves belief about other agents and, thus, Van Benthem's work covers the dynamics of such belief under public announcement. Public announcement gives rise to the change of the epistemic states of all agents in the system. Each agent processes the information of the announcement by some particular *belief revision operator* (lexicographic upgrade and elite change), but without any response to other agents. Dynamic epistemic logic in the work [91] by Van Benthem can express propositions about the agent's knowledge and belief like in epistemic logic [48] and propositions about the change of their knowledge and belief after public announcement. As one contribution, the reasoning with such propositions is reduced to reasoning in epistemic logic.
In future work of this thesis, it might be necessary for $\mathcal{D}$ to integrate the simulation of $\mathcal{A}$ in its process of reasoning, cf. Section 5.2. To this end, we

might reconsider Van Benthem's approach because it combines an agent's belief about its environment and other agents.
Banerjee and Dubois [10] study a responder[13] who communicates epistemic formulas like in [91] with a possibility and necessity operator. But the responder does not receive information from the observer and its epistemic state is just a consistent set of propositional formulas. The observer's reasoning about the responder's epistemic state is modeled by *meta-epistemic logic* (MEL) proposed in their work and may be computed by deduction with the presented sound and complete axiomatization of MEL. The authors further discuss to extend their results based on possibility/necessity modalities to degrees of possibility where the responder has weighted belief (cf. the above paragraph about the interface).
Booth and Nittka [37] consider a responder who may receive query and revision requests in a propositional language from an observer. The responder's epistemic state is comprised of a core (information that cannot be revised like unquestionable belief) and a sequence of pieces of information subsequently received in revision requests. The responder processes the sequence and the core by *linear merging* [45]. Conversely, the observer is ignorant about the responder's initial epistemic state. By means of its requests and the observed reactions, the observer gives a *rational explanation* that should reconstruct the responders's initial epistemic state.

6.4 Extending $\mathcal{D}$'s Interaction Interface

This section sketches an extension of the interaction interface of the defender's implementation with nonmonotonic reasoning to include update requests, beside revision and query requests. As anticipated, an intelligent agent in an inaccessible and dynamic environment should rather base its belief on nonmonotonic reasoning than on a database instance. The extension bases on approaches from the literature surveyed in Section 6.3.
Mainly, in the proposed extension agent $\mathcal{D}$ will update its assertions according to their *recency* with a *prioritized merging operator* as proposed by Delgrande

[13] The emitter in [10] may be viewed as a responder, if the observer may issue queries in the epistemic language.

et al in [45]. Moreover, in a revision the agent will resolve conflicts of assertions with its unquestionable belief by means of prioritized merging according to the *credibility* of assertions.
Altogether, in the proposed extension, agent $\mathcal{D}$'s assertions will include recency and credibility information. Further, the belief operator of Equation 6.4 will be adapted to preprocess the assertions with prioritized merging where we define priorities by recency and credibility. After this preprocessing, the belief operator uses nonmonotonic reasoning from the merged assertions.
The purpose of this section is to point out several commonalities and differences of the two exemplary implementations of $\mathcal{D}$ and to outline further functionality of $\mathcal{D}$ to be investigated in future work.

6.4.1 A Simple Update Operator

Before elaborating the extended implementation, we will illustrate that the way integrity constraints are handled by the database update $\bullet$ of Definition 6.4 is not appropriate for the implementation of an intelligent agent. To this end, we will sketch a simple extension of agent $\mathcal{D}$, where the agent processes updates similar to database updates, and contrast this simple extension with our proposed extension later on. Apart from the differences, the simple update operator we will introduce will also make clear similarities between the implementations in Section 6.1 and in Section 6.2.
For the simple extension, we assume that agent $\mathcal{D}$'s current assertions as are represented as a finite set of literals only:

$$\mathit{as} \subset_{\mathit{fin}} \mathcal{A}t \cup \{\neg a \mid a \in \mathcal{A}t\}.$$

The database update of Definition 6.4 may only succeed if it does not conflict with the integrity constraints. As suggested by Section 6.3, unquestionable belief may be understood as integrity constraints. With this understanding, following the concept of a database update, we tentatively define an update operator[14] $\bullet_{simp}$ of the agent's epistemic state as follows. The update operator preprocesses an input literal L and the assertions as with a *flat merging operator* [45] $*_{simp}$ which resolves conflicts of L and as with $\mathcal{D}$'s

[14] The index *simp* stands for simple.

unquestionable belief in analogy to the database update of Definition 6.4:

$$*_{simp}(\mathrel{|\!\sim}_{\mathcal{D}}, L, \mathit{as}) = \begin{cases} \mathit{as} \setminus \neg(\{L\})^{15} \cup \{L\} & \text{if } \bigwedge(\mathit{as} \setminus \neg(\{L\}) \cup \{L\}) \mathrel{|\!\not\sim}_{\mathcal{D}} \bot, \\ \mathit{as} & \text{otherwise.} \end{cases}$$

The operator $*_{simp}$ gives priority to the assertions as over the more recent information L with the exception that L might override $\neg L$ in the processing of $\mathit{as} \setminus \neg(\{L\}) \cup \{L\}$. To give an idea how to interpret inconsistency under $\mathrel{|\!\sim}_{\mathcal{D}}$, we recall the example of the consequence relation $\mathrel{|\!\sim}_{\mathcal{D}}^{\kappa_1}$ defined by the OCF κ_1 in Section 6.2.2. There, concluding an inconsistency from a formula $A \in \mathcal{L}_{PL}$, that is $A \mathrel{|\!\sim}_{\mathcal{D}}^{\kappa_1} \bot$, says that A is propositionally inconsistent with $\mathcal{D}$'s unquestionable belief because the worlds satisfying the unquestionable belief are those with rank less than ∞.
The update of the epistemic state proceeds as follows.

$$\langle \mathrel{|\!\sim}_{\mathcal{D}}, \mathit{as}\rangle \bullet_{simp} L = \langle \mathrel{|\!\sim}_{\mathcal{D}}, *_{simp}(\mathrel{|\!\sim}_{\mathcal{D}}, L, \mathit{as})\rangle \tag{6.11}$$

After processing an update request, agent $\mathcal{D}$ will notify $\mathcal{A}$ either of the success or of the failure of the update.
The following proposition shows that the database implementation of Section 6.1 is a special case of the implementation of Section 6.2 extended with updates by means of operator $\bullet_{simp}$ if we assume a finite alphabet $\mathcal{At}$.

Proposition 6.1 (Embedding of Database Implementation)**.**

Let $\langle \mathcal{IC}, \mathit{db}\rangle$ be a database over a finite alphabet $\mathcal{At}$ and $\bullet_{db}$ the database update[16] of Def. 6.4. Further, we define a transformation $\mathtt{t}$ *of a database into an epistemic state of Def. 6.7 as follows:*

$$\begin{array}{lrcl} & \mathtt{t}(\langle \mathcal{IC}, \mathit{db}\rangle) & = & \langle \mathtt{t}(\mathcal{IC}), \mathtt{t}(\mathit{db})\rangle \\ \textit{with} & (A,B) \in \mathtt{t}(\mathcal{IC}) & \textit{iff} & \{A\} \cup \mathcal{IC} \vdash_{pl} B \\ \textit{and} & \mathtt{t}(\mathit{db}) & = & \mathit{db} \cup \{\neg a \mid a \in \mathcal{At} \setminus \mathit{db}\}. \end{array}$$

Let L be a literal over $\mathcal{At}$. Then, it holds

1. $\mathtt{t}(\langle \mathcal{IC}, \mathit{db}\rangle)$ *is an epistemic state according to Def. 6.7,*

[15] The operator $\neg()$ is defined in (6.3).
[16] In the scope of this proposition and its proof, we use the notation $\bullet_{db}$ for the operator of Def. 6.4 for clarity of notation.

2. $\mathsf{Bel}^{DB}_{\mathcal{D}}(\langle \mathcal{IC}, \mathcal{db}\rangle) = \mathsf{Bel}^{NMR}_{\mathcal{D}}(\mathsf{t}(\langle \mathcal{IC}, \mathcal{db}\rangle))$,

3. $\mathsf{t}(\langle \mathcal{IC}, \mathcal{db}\rangle \bullet_{db} \{L\}) = \mathsf{t}(\langle \mathcal{IC}, \mathcal{db}\rangle) \bullet_{simp} L$.

The first property ensures that the transformation is well-defined while the last two properties enable $\mathcal{D}$ to do the operations in the transformed database as needed to process query and update requests.
Of course, the priority of the assertions over a more recent piece of information in the processing of $\bullet_{simp}$ may be argued because an update operator should give priority to recency. In the next section, we will introduce an update operator $\bullet_{rec}$ using *prioritized merging* by recency.

6.4.2 Update with Prioritized Merging

In this section, we still focus on the situation that agent $\mathcal{D}$ accepts update requests only. Later on, we will treat the situation that $\mathcal{D}$ accepts both update and revision requests. Following Delgrande et al in [45], an update of assertions may be considered as a process of merging assertions ordered by their recency. Thus, for the time being, $\mathcal{D}$'s current assertions are a list of literals ordered from the least to the most recent piece of information:

$$\mathcal{as} = \langle L_1, \ldots, L_n\rangle \text{ with literals } L_i \text{ over } \mathcal{At}.$$

To form the agent's belief, the belief operator processes this list with a *prioritized merging operator*[17] $\circledast_{rec}$ similar to linear merging [45] and then applies the agent's nonmonotonic reasoning $\mathrel{|\!\!\sim}_{\mathcal{D}}$ on the merging result:

$$\mathsf{Bel}^{NMR}_{\mathcal{D}}(\mathrel{|\!\!\sim}_{\mathcal{D}}, \langle L_1, \ldots, L_n\rangle) = \{B \in \mathcal{L}_{PL} \mid \circledast_{rec}(\mathrel{|\!\!\sim}_{\mathcal{D}}, \langle L_1, \ldots, L_n\rangle) \mathrel{|\!\!\sim}_{\mathcal{D}} B\}.$$

Following [45], we define the prioritized merging operator $\circledast_{rec}$ as an iterative application of a flat merging operator $*_{rec}$ as follows. Let $F, G \in \mathcal{L}_{PL}$ and L_i be literals over $\mathcal{At}$:

$$*_{rec}(\mathrel{|\!\!\sim}_{\mathcal{D}}, F, G) = \begin{cases} G \wedge F & \text{if } G \wedge F \mathrel{|\!\!\not\sim}_{\mathcal{D}} \bot \\ G & \text{otherwise,} \end{cases}$$

$$\circledast_{rec}(\mathrel{|\!\!\sim}_{\mathcal{D}}, \langle\rangle) = \top$$

$$\circledast_{rec}(\mathrel{|\!\!\sim}_{\mathcal{D}}, \langle L_1, \ldots, L_n\rangle) = *_{rec}(\mathrel{|\!\!\sim}_{\mathcal{D}}, L_1, \circledast_{rec}(\mathrel{|\!\!\sim}_{\mathcal{D}}, \langle L_2, \ldots, L_n\rangle))$$

[17] The subscript annotates that the priority is according to recency.

The operator $\circledast_{rec}$ prefers more recent information L_i over less recent information L_j $(1 \leq j < i \leq n)$. Whereas the linear merging operator in [45] checks for propositional contradiction, that is, $G \wedge F \vdash_{pl} \bot$, the operator $*_{rec}$ checks for conflicts with $\mathcal{D}$'s unquestionable belief, that is, $G \wedge F \mathrel{|\!\!\sim}_{\mathcal{D}} \bot$.
When agent $\mathcal{D}$ receives a literal L via an update request from $\mathcal{A}$, it updates its epistemic state with operator $\bullet_{rec}$ and L by appending L to its current assertions:

$$\langle \mathrel{|\!\!\sim}_{\mathcal{D}}, \mathit{as} \rangle \bullet_{rec} L = \langle \mathrel{|\!\!\sim}_{\mathcal{D}}, \mathit{as} \centerdot \langle L \rangle \rangle. \tag{6.12}$$

In the end of this subsection, we highlight two differences between the operator $\bullet_{rec}$ and the operator $\bullet_{simp}$ caused by the inverse priority to recency of information. First, in contrast to the operator $\bullet_{rec}$, the simple update operator $\bullet_{simp}$ does not need the record of recency information since it never overrides less recent information by more recent information due to the lower priority (except for the possible explicit overriding of literals with their negation). Therefore, the operator $\bullet_{simp}$ need not be based on prioritized merging, but flat merging suffices.
Second, the operators $\bullet_{simp}$ and $\bullet_{rec}$ differ in the way of interpreting and handling unquestionable belief. We illustrate this difference with a small example.

Example 6.1 (The Effect of Unquestionable Belief during Updates)**.**

Assume that agent $\mathcal{D}$ considers $a \wedge b$ contradictory[18] *and has no assertions in the beginning. Moreover, its reasoning $\mathrel{|\!\!\sim}^{\kappa}_{\mathcal{D}}$ is implemented by some OCF κ, cf. the example of Sect. 6.2.2. Then, it receives the update requests* $\mathtt{up}(a)$ *and* $\mathtt{up}(b)$ *from agent $\mathcal{A}$ in sequence. With the simple update operator of* (6.11)*, the second update fails due to a conflict with $\mathcal{D}$'s unquestionable belief of $\neg a \vee \neg b$ and the assertion a, that is, $*_{simp}(\mathrel{|\!\!\sim}^{\kappa}_{\mathcal{D}}, b, \{a\}) = \{a\}$. Hence, by the property in* (6.6) *of $\mathcal{D}$'s nonmonotonic reasoning, agent $\mathcal{D}$ believes formula a after the two updates. Further, $\mathcal{D}$ believes $\neg b$ because it reasons with $\mathrel{|\!\!\sim}^{\kappa}_{\mathcal{D}}$ from the unquestionable belief of $\neg a \vee \neg b$ and the assertion a as with propositional entailment like in Sect. 6.2.2.*
After processing the sequence of requests with the update operator $\bullet_{rec}$ of (6.12)*, agent $\mathcal{D}$ has the assertions $\langle a, b \rangle$. The belief operator merges this list to*

[18] That is, it holds $a \wedge b \mathrel{|\!\!\sim}_{\mathcal{D}} \bot$ so that $\mathcal{D}$ unquestionably believes $\neg a \vee \neg b$.

the formula $\circledast_{rec}(\mathrel{|\!\sim}^{\kappa}_{\mathcal{D}}, \langle a, b\rangle) = b$. *Then, after nonmonotonic reasoning, by* (6.6) *agent* $\mathcal{D}$ *believes formula* b *after the two updates and further believes* $\neg a$ *due to its unquestionable belief.*
With the second approach, agent $\mathcal{D}$ *reasons on* $\mathcal{A}$*'s report that the value of* b *has changed and* b *now holds that formula* a *does no longer hold using its unquestionable belief of* $\neg a \vee \neg b$. *By way of contrast, with the first approach, agent* $\mathcal{A}$ *has to report first that formula* a *does no longer hold before* $\mathcal{D}$ *accepts the report that formula* b *now holds. To do so, agent* $\mathcal{A}$ *must reason with* $\mathcal{D}$*'s unquestionable belief and* $\mathcal{D}$*'s assertions to find out the reason of the failure of the request* $\mathsf{up}(b)$ *- but this is rather cumbersome and even more agent* $\mathcal{A}$ *might not be able to do that reasoning.*

6.4.3 Update and Revision with Prioritized Merging

Still, we have to consider the interplay of update requests and revision requests. Again, we use prioritized merging proposed by Delgrande at al in [45]. Assertions that agent $\mathcal{D}$ gathers about a static environment (so that they cannot be ordered by recency) may be ordered by their credibility which usually depends on from which source of information an assertion comes. Following [45], assertions about a static environment are represented as a collection $\mathcal{C}as$ of finite sets of propositional formulas where each set is additionally attached with a *credibility level* $c \in \mathbb{N}_0$. Such collections are then elements of a list ordered by recency. In the remainder of this section, we let agent $\mathcal{D}$'s current assertions be represented by such type of lists:

$$as = \langle \mathcal{C}as_1, \ldots, \mathcal{C}as_n\rangle \qquad \text{with} \qquad \mathcal{C}as_i = \{\mathcal{M}[c] \mid \mathcal{M} \subset_{fin} \mathcal{L}_{PL}, c \in \mathbb{N}_0\}.$$

Agent $\mathcal{D}$'s belief operator merges the assertions according to credibility and recency and then uses nonmonotonic reasoning on the result as follows:

$$\begin{aligned}&\mathsf{Bel}^{NMR}_{\mathcal{D}}(\mathrel{|\!\sim}_{\mathcal{D}}, \langle \mathcal{C}as_1, \ldots, \mathcal{C}as_n\rangle) = \\ &\qquad \{B \in \mathcal{L}_{PL} \mid \circledast_{rec}(\mathrel{|\!\sim}_{\mathcal{D}}, \langle \circledast_{cred}(\mathrel{|\!\sim}_{\mathcal{D}}, \mathcal{C}as_1), \ldots, \circledast_{cred}(\mathrel{|\!\sim}_{\mathcal{D}}, \mathcal{C}as_n)\rangle) \mathrel{|\!\sim}_{\mathcal{D}} B\}.\end{aligned}$$

The operators $\circledast_{rec}$ and $\circledast_{cred}$ might be chosen as one of the prioritized merging operators discussed in [45] like the one presented in the previous subsection, but the details are not relevant for our purpose of sketching an extended interface.

The preprocessing with the prioritized merging operators serves the following purposes. First, the operator[19] $\circledast_{cred}$ resolves conflicts between unquestionable belief in $\mathrel{|\!\!\sim}_{\mathcal{D}}$ and the formulas contained in the collection $\mathcal{C}as_i$ and outputs a propositional formula for each $i = 1, \ldots, n$. Each output formula is the information that the agent considers about a particular state of the environment in the further process of forming its belief. Then, the operator $\circledast_{rec}$ merges these formulas according to their recency.
During an interaction, we assume that agent $\mathcal{A}$ sends with each piece of information a credibility level $c \in \mathbb{N}_0$. Alternatively, agent $\mathcal{D}$ could set the level itself according to the credibility it attributes to agent $\mathcal{A}$ which might be appropriate in a system with more than two agents. If agent $\mathcal{A}$ wants to indicate a change in the environment, it sends an update request $\mathtt{up}(A, c)$ with $A \in \mathcal{L}_{PL}$ and $c \in \mathbb{N}_0$. In response, agent $\mathcal{D}$ adds the received information as the most recent information into its assertions as follows:

$$\langle \mathrel{|\!\!\sim}_{\mathcal{D}}, as \rangle \bullet_{rec} (A, c) = \langle \mathrel{|\!\!\sim}_{\mathcal{D}}, as \centerdot \langle \{A\}[c] \rangle \rangle.$$

In contrast, if agent $\mathcal{A}$ does not want to indicate a change in the environment, but only wants to add information to its previous reports, it will send a revision request $\mathtt{rev}(A, c)$ with $A \in \mathcal{L}_{PL}$ and $c \in \mathbb{N}_0$ to agent $\mathcal{D}$. Then, agent $\mathcal{D}$ sorts formula A into its collection of most recent assertions according to the credibility level c of the information A as follows:

$$\langle \mathrel{|\!\!\sim}_{\mathcal{D}}, as \centerdot \langle \ldots \mathcal{M}[c] \ldots \rangle \rangle \circ_{cred} (A, c) = \langle \mathrel{|\!\!\sim}_{\mathcal{D}}, as \centerdot \langle \ldots (\mathcal{M} \cup \{A\})[c] \ldots \rangle \rangle.$$

We end this section with a final remark on its scope. With this section, we mainly intend to sketch a possible design of the defender $\mathcal{D}$ combing the full functionality considered in this thesis whereas the upcoming chapters deal with aspects of this functionality in separation. However, the sketch leaves open several questions for future investigations beyond this thesis. To name only a few: the choice of the prioritized merging operators[20] $\circledast_{rec}$ and $\circledast_{cred}$ and the properties of the update and revisions operators $\bullet_{rec}$ and $\circ_{cred}$ as studied for other operators in the literature, cf. Section 6.3.

[19]The subscript annotates that the priority is according to credibility.
[20]Here, linear merging was used due to its simplicity

6.4.4 Conclusion

As a conclusion, we see that the implementation of agent $\mathcal{D}$ with a complete propositional database might be seen as a special case of the implementation with nonmonotonic reasoning if the update operator $\bullet_{simp}$ was implemented (Proposition 6.1). But with that implementation of update, the agent would not reason with the information received through update requests as expected from an intelligent agent as we illustrated in Example 6.1.
Our motivation for investigating the database implementation despite its shortcomings for intelligent agents in an inaccessible, dynamic environment is the following. First, the database implementation is simpler than the nonmonotonic reasoning implementation, even more if the latter is extended as outlined in this section. Its relative simplicity makes it easier to study advanced aspects in the security engineering task for the database implementation which we will do in Chapter 7. Second, the study of the database implementation should be a preliminary step to the study of the nonmonotonic reasoning implementation. We may suppose so because the core of each of the operators $\bullet_{simp}$, $\bullet_{rec}$ and $\circ_{cred}$ is a flat merging operator $*$ which essentially checks for a conflict of two pieces of information with $\mathcal{D}$'s unquestionable belief and drops the piece of information with less priority in case of a conflict. Taking up the analogy of Proposition 6.1, we may consider such a check to be analogous to the check for integrity violation during database update.

Chapter 7

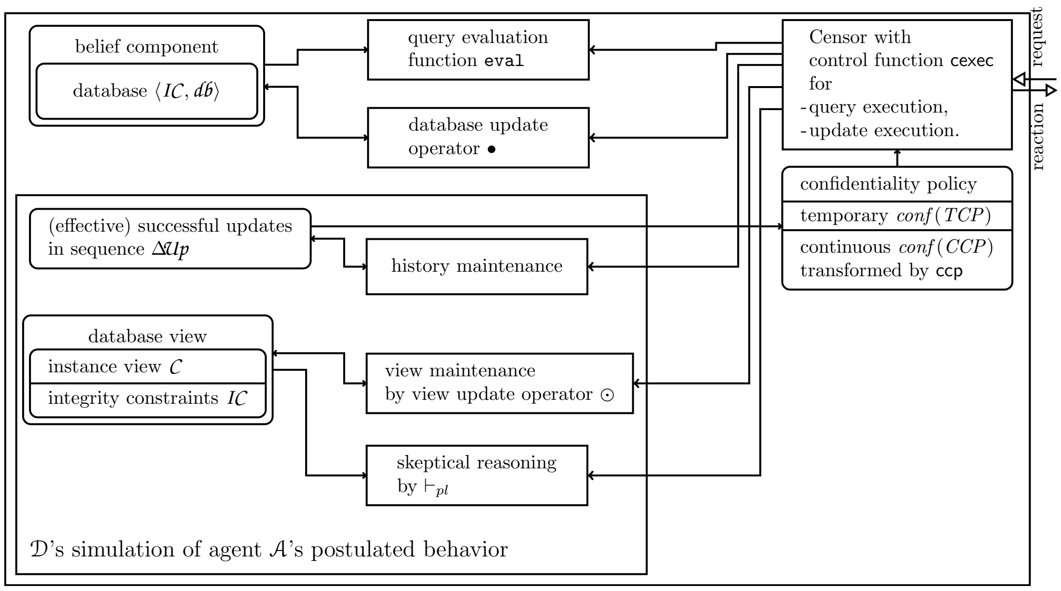
Updating Belief without Revealing Secrets

In this chapter, we study the defending agent $\mathcal{D}$'s controlled processing of update requests from the attacking agent $\mathcal{A}$ by the example of view update transactions on complete propositional databases, cf. Figure 6.1.

The contributions of this chapter are centered around three aspects. The first aspect is the conflict between integrity and confidentiality that has been studied in database security research: Maintenance of integrity constraints is a major task during database updates, but notifying integrity violation to a database user might breach the confidentiality of information in the database. The second aspect is the correctness of the simulation of the attacker $\mathcal{A}$'s postulated reasoning $\mathcal{K}_{\mathcal{A}}$. Correctness essentially means that all the information that $\mathcal{A}$ gained about $\mathcal{D}$'s epistemic state (here, the database) by observing the output of $\mathcal{D}$'s control function is represented in the attacker view used in the simulation, cf. Example 4.5. Related to the second aspect, the third aspect is the simulation of $\mathcal{A}$'s reasoning about $\mathcal{D}$'s previous belief. This simulation is necessary for the protection of continuous confidential belief.

The first contribution of this chapter is an implementation of the control function of the reacting agent $\mathcal{D}$'s censor as outlined in Figure 7.1. Another contribution is a formal proof of the effectiveness of these control functions in enforcing $\mathcal{D}$'s confidentiality interests expressed by its policy with the formal semantics of Definition 4.6. The control function's effectiveness is partly achieved by the correctness of the simulation of the attacker's reasoning. The third contribution is an analysis of $\mathcal{D}$'s interest of sharing information with agent $\mathcal{A}$ as cooperatively as possible. To this end, agent $\mathcal{D}$ follows a policy of *last-minute intervention* [16]. To judge the performance of control functions under this policy, we list further desirable properties of control functions. One property defines the requester $\mathcal{A}$'s information need under a policy of last-minute intervention. Then, we show that the presented control function for view update transactions achieves all these properties while no other function can increase availability under the last-minute intervention policy without lacking one of these properties.

The organization of this chapter is as follows. Above all, we make clear the scope of our contributions by surveying related work in Section 7.1.
Then, in the remaining sections, we elaborate on constructing a confidentiality preserving agent $\mathcal{D}$ as part of the security engineering task. The final construction is outlined in Figure 7.1. The implementation of $\mathcal{D}$'s belief component in Figure 7.1 is the basis of this construction (Activity 1 of security engineering) and has been introduced in Section 6.1.
Section 7.2 specifies $\mathcal{A}$'s awareness of the details of $\mathcal{D}$'s implementation (Activity 2 of security engineering).
Afterwards, Section 7.3 presents the simulation of the requesting agent $\mathcal{A}$'s postulated behavior as an attacker which is the crucial means of the control functions. In that section, $\mathcal{A}$'s skeptical entailment is simulated by means of propositional entailment. Choosing propositional entailment, $\mathcal{A}$'s reasoning about continuous confidential belief is simulated by an appropriate transformation of each continuous confidential belief to a propositional formula enforcing its protection. We show both the correctness of the transformation and the correctness of the simulation (which contributes to Activity 3 of security engineering).
Then, Section 7.4 presents the control functions for query and view update transaction execution and formally proves confidentiality preservation by means of these functions (completes Activity 3 of security engineering).
Finally, Section 7.5 discusses and formally analyses the aspect of information sharing under a policy of last-minute intervention.
We end this chapter with a discussion of the presented contributions and open problems in Section 7.6.



belief component
database $\langle IC, db \rangle$
query evaluation function eval
database update operator $\bullet$
Censor with control function cexec for -query execution, -update execution.
request
reaction
confidentiality policy
temporary $conf(TCP)$
continuous $conf(CCP)$ transformed by ccp
(effective) successful updates in sequence ΔUp
history maintenance
database view
instance view C
integrity constraints IC
view maintenance by view update operator $\odot$
skeptical reasoning by $\vdash_{pl}$
$\mathcal{D}$'s simulation of agent $\mathcal{A}$'s postulated behavior

Figure 7.1: Design of a confidentiality preserving agent $\mathcal{D}$ as defender against a requesting agent $\mathcal{A}$ as attacker

7.1 Related Work

Conflict Between Integrity and Confidentiality In the database security community, the so called conflict between integrity and confidentiality has been treated in several works, for example, in [67, 52, 21, 22]. This conflict arises from a database user's awareness of the integrity constraints in the database schema while the user is not authorized to acquire particular confidential information from the database instance. For example, assume the integrity constraint that "each employee is manager of at most one project". The database user requests to insert "Smith is manager of project A". From a successful insertion the user can conclude that Smith is manager of no other project than A whereas from a failed insertion (due to the integrity constraint) he can conclude that Smith is already manager of some other project. The user may combine the latter fact with the prior knowledge "there are only two projects A and B" and infer the confidential fact "Smith is manager of project B". In this situation, insertion is to fail due to integrity, but the database user must not be informed about this failure due to confidentiality so that an *immediate conflict* between the two requirements emerges. In order to resolve this conflict, other work insert misinformation to the user's database view, for example, by *lying* or *polyinstantiation*. Following the example, an appropriate lie would be a successful insertion although conforming to the constraint no insertion has been performed. We will survey a selection of these work in the following.

Within the line of research of Controlled Interaction Execution (CIE) surveyed in Section 6.3, control procedures for view update transactions, provider updates and view refreshments are presented (cf. [21] for the overall framework; cf. [22] for a detailed study of view update transactions including refreshments). These procedures resolve the conflict between integrity and confidentiality as follows. The procedures maintain a *materialized database view* (called logfile) for each database user with possibly distorted data in which the procedures track information released to the respective user. If a database user requests to modify its materialized database view, but cannot be notified about a resulting integrity violation, the view update transaction pretendedly succeeds, yet actually, without updating the database instance. Thus, the procedures implement the *lying* approach of CIE, cf. Section 6.3. This solution is close to polyinstantiation used in the area of multilevel secure (MLS) databases which we will survey below. Further, by allowing the control procedures to add spontaneous updates, which might affect the correctness of the data, the authors of [22] achieve continuous confidentiality, cf. Definition 4.6.

In an MLS database instance, data items and schema objects are assigned a security level as a *classification*. Likewise subjects (such as database users) receive a security level as a *clearance*. The set of security levels must be partially ordered. Information flow is only permitted from a lower level to a higher level in the partial order. The conflict illustrated in the above example may be caused by the fact "Smith is manager of project B" classified high, whereas the database user requiring the insertion has clearance low. To resolve it, the fact "Smith is manager of project A" is inserted but classified to the user's clearance low. As a consequence, the instance is *polyinstantiated* which means that Smith is a manager of two projects where each project is associated with a different security level [67, 52]. The inserted fact is implicitly considered as a *cover story* (which may be seen as a form of lying) [52, 43].

Other Works on Confidentiality and Update Xiao and Tao in [96] study the republication of generalized data from a relational table after insertions and deletions in that table while the privacy of individuals must be ensured. The publication of generalized data under privacy constraints has been widely investigated in research on database security, cf. [51] for a survey. In this research, an unpublished relational table contains data of individuals. The privacy interest is the each individual cannot be linked to the value of a sensitive attribute with a certain accuracy. A common assumption is that for every individual the attacker a priori knows the individual's values of particular attributes, the quasi identifiers[1]. For data publication, the values of quasi identifier attributes are generalized, for example, by replacing exact values with intervals as in [96]. In general, the attacker infers possible values of a sensitive attribute by linking a set of published generalized tuples (with the sensitive attribute) to each individual via the quasi identifiers. Xiao and Tao follow this general approach from research on privacy in data publication, but additionally consider that the attacker might base its inferences on a history of generalized tables each released after insertions and deletions in the original table. Moreover, the authors assume that the attacker knows whether the data of an individual is present as a tuple in the original table and, hence, whether that tuple has been inserted or deleted. Clearly, Xiao and Tao assume a very specific form of the attacker's knowledge which in particular does not include integrity constraints. However, their work indicates an interesting aspect for future work of this thesis. The formal privacy requirement in [96] is defined as an upper bound on the disclosure risk: In the set of values of a sensitive attribute that the attacker might link to an individual each value occurs less than $\frac{1}{m}$-times. The requirement is stricter than the confidentiality property in this thesis which only refers to indistinguishable situations. To establish the bound on the disclosure, Xiao and Tao introduce the requirement of *m-invariance* on the history of generalized data. Roughly, m-invariance ensures that the set of sensitive values linked to an individual is an invariant (under republication) and contains at least m distinct values.

[1] Quasi identifiers are understood to identify a relatively small group of individuals.

Cooperative Information Sharing In the database security community, several work care for sharing information to the farthest degree possible while ensuring the confidentiality of particular information in a database instance. To this end, several approaches optimize some availability measure, for example, minimize the number of distorted database entries [36] or select a lowest classification of data in MLS databases [44]. These approaches precompute a view to be published or a classification, respectively, whereas in this thesis the attacker's reasoning about confidential information is simulated dynamically. Other research preprocess data that is released for statistical queries or data mining so that after the release the loss of privacy is minimized whereas subordinately data utility is maximized, cf. [2] for a survey. Yet, these approaches have not accounted for updates of the database instance so far.

Other Aspects There are several aspects beyond the scope of this thesis. Other agents than the attacker might initiate an update, for example, in the MLS database setting, high level users [67] or, in the context of CIE, the data provider [21]. Then, the attacker's aged database view has to be refreshed, cf. [21, 22, 90]. Further, there are other types of unwanted information flow during the execution of an update transaction. This has been extensively studied in the MLS area, such as covert channels in concurrency control protocols etc [66].

7.2 Implementation of Defender $\mathcal{D}$ as Seen by Attacker $\mathcal{A}$

The model $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of the agent system in Definition 4.5 gives us two ways to detail $\mathcal{A}$'s awareness about $\mathcal{D}$'s implementation. First, we will specify the interface of the control function. The interface and the implementation of the function are open to $\mathcal{A}$ by Assumption 2. Implicitly, this interface defines invariants of the control function that $\mathcal{A}$ is aware of. Second, we will define the set $\mathcal{St}^{DB}_{Init}$ of the model $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$. This set represents facts about $\mathcal{D}$'s initial state that $\mathcal{A}$ is aware of in the implementation.
Beside types of requests that $\mathcal{D}$ accepts, $\mathcal{D}$'s abstract state in Definition 4.2 defines the interface of the control function of Definition 4.3. The next

definition summarizes the implementation of $\mathcal{D}$'s abstract state from Section 6.1.

Definition 7.1 (State of Defender $\mathcal{D}$ with Database).
Cf. Definition 4.2

The set $\mathcal{St}^{DB}$ of agent $\mathcal{D}$'s local states consists of tuples with the following components:

1. *Background knowledge:*
   a set $\mathcal{IC}$ of integrity constraints.

2. *Evidential knowledge:*
   a database instance $\mathfrak{db}$ such that $\langle \mathcal{IC}, \mathfrak{db} \rangle$ is a database of Def. 6.1.

3. *Belief operator:*
   the operator $\mathsf{Bel}^{DB}_{\mathcal{D}}$ defined in (6.1).

5. *History:*
   a sequence $\mathcal{H}^{DB}$ of pairs of request $\theta \in \mathcal{Req}^{DB}$ and respective reaction $react \in \mathcal{Rea}$, cf. Definition 4.1.

6. *Attacker view (view for short):*
   a database view $\langle \mathcal{IC}, \mathcal{C} \rangle$ of the database $\langle \mathcal{IC}, \mathfrak{db} \rangle$ according to Definition 6.2.

In the definition, the invariants of the control functions are that the components $\mathcal{IC}$ and $\mathfrak{db}$ form a database and that the attacker view $\langle \mathcal{IC}, \mathcal{C} \rangle$ is a database view. Further, according to the definition, $\mathcal{A}$ knows the integrity constraints $\mathcal{IC}$ because they are contained in the view. Thus, all data components but the database instance are open to $\mathcal{A}$. In the literature on database security there are alternative approaches which assume that some integrity constraints may be hidden from the attacker, for example, in [43]. Finally, we specify the attacker's a priori knowledge about possible initial states of $\mathcal{D}$ in the implementation by defining the set $\mathcal{St}^{DB}_{Init}$ of Assumption 2. These initial states may be seen as preconditions of the control function. The set $\mathcal{S}t^{DB}_{Init}$ is defined by a predicate $\texttt{pre}$ over $\mathcal{S}t^{DB}$

$$\begin{aligned}&\texttt{pre}(\mathcal{IC}, \boldsymbol{db}, conf, \mathcal{H}^{DB}, \langle \mathcal{IC}, \mathcal{C}\rangle) \text{ is true iff}\\ &\qquad\qquad \mathcal{H}^{DB} = \langle\rangle\\ &\quad\text{and}\quad \mathcal{IC} \cup \mathcal{C} \not\vdash_{pl} B \text{ for all } B \in conf(TCP) \cup conf(CCP) \text{ hold.}\end{aligned} \tag{7.1}$$

The definition of the set $\mathcal{S}t^{DB}_{Init}$ completes our postulates about $\mathcal{A}$'s awareness of $\mathcal{D}$'s implementation in this chapter.

Throughout this chapter, we assume that $\mathcal{D}$ informs $\mathcal{A}$ for each view update transaction request which of the requested insertions and deletions have been executed and which have not been executed. Single insertions or deletions might not be executed because the corresponding atom is already contained in the database instance or not contained, respectively.

Assumption 7 (Notification About Effective Updates)**.**

Let $\mathsf{cexec} : \mathcal{S}t^{DB} \times \mathcal{R}eq^{DB} \rightarrow \mathcal{S}t^{DB}$ *be a control function for the exemplary implementation of agent* $\mathcal{D}$ *of Section 6.1 and let* $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ *be the system of defender* $\mathcal{D}$ *and attacker* $\mathcal{A}$ *defined by this function. Further, let* $(r, m) \in \mathcal{PT}(\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}})$ *be a point such the* m*-th request in run* r *is of the form* $\mathsf{up}(\mathcal{L}) \in \mathcal{R}eq^{DB}$*. Then, by means of the history* $\mathcal{H}^{DB}(r, m)$*, agent* $\mathcal{A}$ *is able to determine the set* $\mathcal{U} \subseteq \mathcal{L}$ *defined by the two properties* $\boldsymbol{db}(r, m-1) \bullet \mathcal{U} = \boldsymbol{db}(r, m)$ *and* $\boldsymbol{db}(r, m-1) \not\models_{PL} L$ *for all* $L \in \mathcal{U}$*.*[2] *We call this set effective updates.*

The assumption largely simplifies the implementation of control functions in this chapter and the verification of the confidentiality property. The reason is that agent $\mathcal{D}$ must not take effort to hide the effectiveness of requested updates. It either reveals the set of all effective updates or refuses the view update transaction request altogether.

[2] The second property is also necessary since by Definition 6.4 for every literal L with $\boldsymbol{db} \models_{PL} L$ it holds $\boldsymbol{db} \bullet \{L\} = \boldsymbol{db}$.

7.3 Simulating the Attacker's Skeptical Reasoning About View Update Transactions

In Part I, the defending agent $\mathcal{D}$'s confidentiality policy is expressed by an equivalent possibility policy. The semantics of a possibility policy bases on a model of the attacking agent $\mathcal{A}$'s reasoning about $\mathcal{D}$'s belief by the operator $\mathcal{K}_\mathcal{A}$ over the abstract specification $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of the multiagent system. Essentially, the goal of simulating the attacker $\mathcal{A}$'s reasoning is deciding for each point $(r, m) \in \mathcal{PT}(\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}})$ whether agent $\mathcal{A}$ is able to skeptically conclude a confidential belief $S \in \mathit{conf}(\mathit{TCP}) \cup \mathit{conf}(\mathit{CCP})$, formally, whether

$$\mathcal{K}_\mathcal{A}(r, m) \cap I_S = \emptyset \tag{7.2}$$

holds where I_S is a set of points relevant for the protection of S. In the following, we will present an algorithmic solution for this decision by means of propositional entailment. The solution is straightforward for the protection of temporary confidential belief and follows the standard solution in the work on Controlled Interaction Execution, for example, in [19]. The relevant $\mathcal{D}$-property for the protection of a temporary confidential belief S is given by Definition 4.7 as the set

$$I_S^{TCP} = \{(r', m') \mid S \notin \mathsf{Bel}^{DB}_\mathcal{D}(\mathcal{IC}(r', m'), \mathcal{db}(r', m'))\}.$$

To simulate the attacker $\mathcal{A}$'s reasoning $\mathcal{K}_\mathcal{A}$, we will identify agent $\mathcal{A}$'s available information $r_\mathcal{A}(m)$ in run r at time m with the attacker view $\langle \mathcal{IC}, \mathcal{C} \rangle$ (hence, the history $\mathcal{H}^{DB}$ contained in $r_\mathcal{A}(m)$ will not be used in this simple case of simulating reasoning about temporary confidential belief). This way, we will further identify the points $\mathcal{K}_\mathcal{A}(r, m)$ that $\mathcal{A}$ considers possible with its available information $r_\mathcal{A}(m)$ with the database instances satisfying $\mathcal{IC} \cup \mathcal{C}$. Then, the simulation will check equation (7.2) for each temporary confidential belief $S \in \mathit{conf}(\mathit{TCP})$ by deciding whether every database instance satisfying $\mathcal{IC} \cup \mathcal{C}$ is not a model of $\neg S$ or, equivalently, deciding $\mathcal{IC} \cup \mathcal{C} \vdash_{pl} S$. In summary, the control functions will simulate $\mathcal{A}$'s reasoning with operator $\mathcal{K}_\mathcal{A}$ by means of the database view $\mathcal{V} = \langle \mathcal{IC}, \mathcal{C} \rangle$ and the operator

$$\mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C} \rangle) = \{F \in \mathcal{L}_{PL} \mid \mathcal{IC} \cup \mathcal{C} \vdash_{pl} F\}. \tag{7.3}$$

In Subsection 7.4.3, we will discuss which invariants of a control function suffice to ensure that the simulation of $\mathcal{K}_\mathcal{A}$ by means of the database view and operator $\mathsf{skeptical}$ is correct.
In the remainder of this section, we will present the more intricate algorithmic solution for verifying equation (7.2) for each continuous confidential belief $S \in \mathit{conf}(CCP)$. In this case, by Definition 4.7, the set of points relevant for its protection is given by the set

$$\begin{aligned}&\mathcal{PT}(R^{CCP}_{S,m}) = \\ &\quad \{(r',m') \mid m' \in \mathbb{N}_0 \text{ and for all } n \le m: \quad S \notin \mathsf{Bel}^{DB}_{\mathcal{D}}(\mathcal{IC}(r',n), \mathcal{db}(r',n))\}.\end{aligned} \tag{7.4}$$

In contrast to the protection of temporary confidential belief, the set $\mathcal{PT}(R^{CCP}_{S,m})$ involves belief that agent $\mathcal{D}$ had previously. In the following, we elaborate on a transformation of a continuous confidential belief S into a propositional formula that represents the set $\mathcal{PT}(R^{CCP}_{r,m})$ protecting S. To start with, we illustrate the general idea behind this transformation by the following example.

Example 7.1 (Reasoning About Continuous Confidential Belief).

Agent $\mathcal{D}$ declared its confidentiality interests by $\mathit{conf}(TCP) = \emptyset$ and $\mathit{conf}(CCP) = \{s_1 \wedge s_2\}$. After a view update transaction request $\mathtt{up}(\{\neg b, \neg s_2\})$ from agent $\mathcal{A}$, $\mathcal{D}$ successfully updates the database instance $\mathcal{db}_2 = \{b, s_1, s_2\}$ with integrity constraints $\mathcal{IC} = \{s_1 \vee s_2\}$. The ordinary transaction processing of Section 6.1.2 results in the database instance $\mathcal{db}_3 := \{s_1\}$ with instance view $\mathcal{C}_3 = \{\neg a, \neg b, \neg s_2, \ldots\}$ by Definition 6.6. Agent $\mathcal{A}$ is able to infer s_1 from $\neg s_2$ in the instance view and the integrity constraints $\mathcal{IC}$ by propositional entailment so that it knows that $\mathcal{db}_3$ satisfies $s_1 \wedge \neg s_2$. Now, the agent may use the result of Lemma 6.3 saying, in this situation, that $\mathcal{db}_3 \models_{PL} s_1 \wedge \neg s_2$ holds iff $\mathcal{db}_2 \models_{PL} s_1 \wedge s_2$ holds.[3] *This way, agent $\mathcal{A}$ is able to infer that $\mathcal{D}$ has believed the continuous confidential belief $s_1 \wedge s_2$ with the previous instance $\mathcal{db}_2$.*

The key to the just demonstrated inference of continuous confidential belief is Lemma 6.3. To apply this lemma, it suffices that agent $\mathcal{A}$ knows that $\mathcal{db}_2$ has

[3] To apply the lemma, agent $\mathcal{A}$ needs to know that the two equalities $\mathcal{db}_3 = \mathcal{db}_2 \bullet \{\neg b, \neg s_2\}$ and $s_1 \wedge \neg s_2 = \mathsf{neg}(s_1 \wedge s_2, \{\neg b, \neg s_2\})$ hold.

been updated to $\mathcal{db}_3$ by deleting b and s_2 or, even less precise, that the atoms b and s_2 have been altered from $\mathcal{db}_2$ to $\mathcal{db}_3$. This knowledge is postulated by Assumption 7. The important observation from this example is that $\mathcal{D}$'s current belief of $s_1 \wedge \neg s_2$ becomes relevant for the protection of the continuous confidential belief $s_1 \wedge s_2$.
In the following, we will generalize Lemma 6.3 to iterated view update transactions. To this end, as illustrated by the example, for each updated database instance $\mathcal{db}_i$ we need to determine the atoms that have been altered from $\mathcal{db}_i$ to the final instance $\mathcal{db}_k$ with $i \leq k$. These atoms define the set $\Delta\mathcal{Up}[i]$ of the following definition.

Definition 7.2 (Sequence $\Delta\mathcal{Up}$).

Let $\mathcal{Seq} = \langle \mathcal{U}_1, \dots, \mathcal{U}_{k-1} \rangle$ be a finite sequence of sets of literals with each set $\mathcal{U}_i$ being defined over distinct atoms. Then, we define the sequence $\Delta\mathcal{Up}[\mathcal{Seq}]$ as follows

$$\Delta\mathcal{Up}[\mathcal{Seq}][i] =$$
$$\begin{cases} (\Delta\mathcal{Up}[\mathcal{Seq}][i+1] \setminus \mathsf{At}(\mathcal{U}_i)) \cup (\mathsf{At}(\mathcal{U}_i) \setminus \Delta\mathcal{Up}[\mathcal{Seq}][i+1]) & 1 \leq i \leq k-1 \\ \emptyset & i = k. \end{cases}$$

Lemma 7.1 (Query Evaluation in Previous Instances).

Let $\mathcal{Seq} = \langle \mathcal{U}_1, \dots, \mathcal{U}_{k-1} \rangle$ be a sequence of sets of literals with each set $\mathcal{U}_i$ being defined over distinct atoms. Further, let $\langle \mathcal{db}_1, \dots, \mathcal{db}_k \rangle$ be a sequence of database instances such that for all $i = 1, \dots, k-1$ it holds

$$\mathcal{db}_i \not\models_{PL} L \text{ for all } L \in \mathcal{U}_i \qquad \text{and} \qquad \mathcal{db}_{i+1} = \mathcal{db}_i \bullet \mathcal{U}_i.$$

Then, for all formulas $A \in \mathcal{L}_{PL}$ it holds for all $i = 1, \dots, k$

$$\mathcal{db}_i \models_{PL} A \qquad \text{iff} \qquad \mathcal{db}_k \models_{PL} \mathsf{neg}(A, \Delta\mathcal{Up}[\mathcal{Seq}][i]).$$

In words, agent $\mathcal{D}$ has previously believed formula A with database instance $\mathcal{db}_i$ iff agent $\mathcal{D}$ currently believes $\mathsf{neg}(A, \Delta\mathcal{Up}[\mathcal{Seq}][i])$. Like Example 7.1, this result suggests that continuous confidential belief S might be protected by protecting some current belief. Even more, the result suggests *how* to define this current belief from S and a sequence $\mathcal{Seq}$ of effective updates appropriately.

We make this precise in the following. To protect a continuous confidential belief S, agent $\mathcal{A}$ should not be able to skeptically conclude (with operator $\mathcal{K}_{\mathcal{A}}$) that $\mathcal{D}$ has previously believed S at some point of time or currently believes S. Under Assumption 7, agent $\mathcal{A}$ may observe the sequence of effective updates and thus is enabled to reason about $\mathcal{D}$'s previous or current belief by Lemma 7.1. Using the lemma, a statement about previous or current belief of S is expressed by the propositional formula $\mathsf{ccp}(S, \Delta\mathcal{U}p[\mathcal{SU}p])$ saying "$\mathcal{D}$ has previously believed S at some point of time or currently believes S with respect to the sequence $\mathcal{SU}p$ of effective updates" and defined as follows.

Definition 7.3 (The Transformation ccp).

Let $\mathcal{S}eq$ be a finite sequence of k sets of literals and let S be a formula. Then, we define the following formula

$$\mathsf{ccp}(S, \mathcal{S}eq) := \mathsf{neg}(S, \mathcal{S}eq[1]) \vee \ldots \vee \mathsf{neg}(S, \mathcal{S}eq[k+1]).$$

If $\mathcal{S}eq = \langle\rangle$ then we set $\mathsf{ccp}(S, \mathcal{S}eq) = S$.

We end this section outlining the simulation of $\mathcal{A}$'s reasoning to decide Equation 7.2 based on Equation 7.4. Finally, we give a correctness result for the outlined simulation.

For the simulation, like in the situation with temporary confidential belief, agent $\mathcal{A}$'s reasoning with operator $\mathcal{K}_{\mathcal{A}}$ is simulated with the operator of (7.3) defined on the view. Further, the target of $\mathcal{A}$'s reasoning, some continuous confidential belief S, is transformed to the formula $\mathsf{ccp}(S, \Delta\mathcal{U}p[r, m])$ where the sequence $\Delta\mathcal{U}p[r, m]$ is determined from the effective updates tracked in the history of run r at time m by Assumption 7. This way, reasoning about the set of points of (7.4) with operator $\mathcal{K}_{\mathcal{A}}$ under Assumption 7 is simulated by propositional entailment of formula $\mathsf{ccp}(S, \Delta\mathcal{U}p[r, m])$ from $\mathcal{IC} \cup \mathcal{C}$. If our considerations focus on the control function cexec, we will further use the notation $\Delta\mathcal{U}p[s_{\mathcal{D}}]$ instead of $\Delta\mathcal{U}p[r, m]$ where $s_{\mathcal{D}} = r_{\mathcal{D}}(m)$ is agent $\mathcal{D}$'s state in run r at time m which tracks the effective update in that run until time m in the history component.

The following result says that agent $\mathcal{A}$ may find out that a run r has property $R^{CCP}_{S,m}$ of (7.4) by checking the database instance in that run at time m in the definition $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of the system. Therefore, reasoning about runs can be reduced to reasoning about database instances.

Proposition 7.1 (Correctness of Transformation ccp).

Let cexec *be a control function of Def. 4.3 with domain* $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}$*. Further, let* $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ *be the system of defender* $\mathcal{D}$ *and attacker* $\mathcal{A}$ *defined by* cexec*. Then, for all points* $(r, m) \in \mathcal{PT}(\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}})$ *and for all* $S \in \mathcal{L}_{PL}$

$$\mathcal{db}(r, m) \not\models_{PL} \mathsf{ccp}(S, \Delta\mathcal{Up}[r, m]) \qquad \textit{of Def. 7.3 holds}$$

$$\textit{iff}$$

$$r \in R^{CCP}_{S,m} \qquad \textit{of (4.2) holds.}$$

The proposition follows almost immediately from the definitions and Lemma 7.1.

7.4 Control Functions For Iterated Query and View Update Transaction Requests

This section presents the control function of $\mathcal{D}$'s censor for query requests and view update transaction requests. The control function is abstractly introduced by Definition 4.3 and its domain $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}$ in Section 6.1. For conciseness of the presentation, we present the implementation of the control function by two separate implementations for each type of request.

7.4.1 Query Requests

On a query request $\mathtt{que}(A) \in \mathcal{Req}^{DB}$ the control function for controlled query executions proceeds with checking the following three cases:

1. The answer to the query by ordinary query evaluation of Definition 6.3 is already included in the current instance view $\mathcal{C}$.

2. In some situations, being informed about $\mathcal{D}$'s belief of formula A or $\mathcal{D}$'s belief of formula $\neg A$ enables agent $\mathcal{A}$ to skeptically conclude that $\mathcal{D}$ currently believes $S \in \mathit{conf}(TCP)$ or that $\mathcal{D}$ has believed $S \in \mathit{conf}(CCP)$ or currently believes it. This check is designed to be independent of the ordinary query evaluation result (database instance independent) to preclude meta-inferences.

3. The query answer has to be included into the history of the interaction (and is, thus, forwarded to agent $\mathcal{A}$ by the censor).

Procedure 1 cexec for controlled query execution

```
Input: state $s \in \mathcal{St}^{DB}$ , request $\mathsf{que}(A)$
Output: state $s' \in \mathcal{St}^{DB}$
 Case 1 Agent $\mathcal{A}$ knows the query answer
 1: if $A \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C} \rangle)$ then
 2:     $note = A$
 3: else if $\neg A \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C} \rangle)$ then
 4:     $note = \neg A$
 Case 2 Answer reveals a potential secret
 5: else if
        [$\mathsf{ccp}(S, \Delta\mathcal{Up}[s]) \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C} \cup \{A\} \rangle)$
                 or
        $\mathsf{ccp}(S, \Delta\mathcal{Up}[s]) \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C} \cup \{\neg A\} \rangle)$]   for an $S \in conf(CCP)$
    or
        [$S \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C} \cup \{A\} \rangle)$
                 or
        $S \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C} \cup \{\neg A\} \rangle)$]   for an $S \in conf(TCP)$
    then
 6:     $note = refuse$
 Case 3 Query evaluation
 7: else if $A \in \mathsf{Bel}_{\mathfrak{D}}^{DB}(\langle \mathcal{IC}, db \rangle)$ then
 8:     $note = A$
 9:     $\mathcal{V} = \langle \mathcal{IC}, \mathcal{C} \cup \{A\} \rangle$
10: else
11:     [then $\neg A \in \mathsf{Bel}_{\mathfrak{D}}^{DB}(\langle \mathcal{IC}, db \rangle)$]
12:     $note = \neg A$
13:     $\mathcal{V} = \langle \mathcal{IC}, \mathcal{C} \cup \{\neg A\} \rangle$
```

```
14: end if
15: Append $(\mathtt{que}(A), note)$ to history
```

Proposition 7.2 (Output States of Controlled Query Execution)**.**

Procedure 1 outputs a state of Def. 7.1.

The proposition is immediate from the definition of the procedure.

7.4.2 View Update Transaction Requests

In Section 7.1, we surveyed approaches from the literature that deal with an *immediate conflict* between integrity and confidentiality, that is updating violates the integrity constraints, but notifying the requester about the violation discloses confidential information. To resolve this conflict, those approaches add lies (or cover stories) to the database view.
As a different approach, the control function for view update transaction presented here identifies situations in which the enforcement of integrity might lead to a confidentiality violation, that is, there is a *potential conflict*. In such situations, the control function aborts the transaction to escape a possible immediate conflict in the future. This approach comes at the cost of cooperative information sharing because the control function aborts in situations with no immediate, but a potential conflict between integrity and confidentiality.
The control function for controlled view update transaction executions follows the ordinary procedure of a view update transaction (especially, achieves atomicity and consistency), cf. Section 6.1.2, while it ensures that the attacker $\mathcal{A}$ does not skeptically conclude a temporary or continuous confidential belief. To this end, the function uses the simulation of $\mathcal{A}$'s postulated reasoning presented in Section 7.3. Essentially, a view update transaction involves several implicit queries to ensure acceptability [9] and consistency, cf. [22]. On a view update transaction request $\mathtt{up}(\mathcal{L}) \in \mathfrak{Req}^{DB}$ from agent $\mathcal{A}$ the control function proceeds by checking the following four cases:

1. For at least one input literal, due to a conflict with the confidentiality policy, agent $\mathcal{A}$ cannot be informed whether the requested deletion or insertion has to be executed (outstanding update request). In case of a

conflict, the transaction aborts to adhere to Assumption 7 and the atomicity requirement for view update transactions of Section 6.1.2.

2. A successful view update transaction necessarily would violate the confidentiality policy (because the updated database view reveals a temporary or continuous confidential belief). In this case, the transaction aborts.

3. Either a potential conflict between integrity and confidentiality arises or the integrity constraints cannot be preserved. In this case, the transaction aborts.

4. The database instance is to be updated.

Procedure 2 cexec for controlled view update transaction execution

```
Input: state $s \in \mathcal{S}t^{DB}$, request $\mathtt{up}(\mathcal{L})$
Output: state $s' \in \mathcal{S}t^{DB}$
    Case 1 Outstanding Updates
 1: for all literals $L$ in $\mathcal{L}$ do
 2:     Pose the query $\mathtt{que}(L)$ to Procedure 1
        (which modifies $\mathcal{D}$'s state $s$ accordingly
        so that $\mathcal{A}$ is informed about the query notification)
 3:     if the notification to the query is a refusal then
 4:         $note$ = transaction refused
 5:         Append $(\mathtt{up}(\mathcal{L}), note)$ to history
 6:         Exit procedure
 7:     else if the notification to the query is $\neg L$ then
 8:         $\mathcal{U} := \mathcal{U} \cup \{L\}$
 9:     end if
10: end for
    Case 2 Performing Transaction Reveals a Potential Secret
11: Set $\mathcal{S}eq$ to the sequence of all effective updates from the history in state $s$
    and append $\mathcal{U}$.
```

```
12: if
        $\mathsf{ccp}(S, \Delta\mathcal{U}p[\mathcal{S}eq]) \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}\rangle \odot \mathcal{U})$   for an $S \in conf(CCP)$
                        or
        $S \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}\rangle \odot \mathcal{U})$   for an $S \in conf(TCP)$
    then
13:   $note$ = view update transaction violates confidentiality
14:   Append $(\mathsf{up}(\mathcal{L}), (\mathcal{U}, note))$ to history
15:   Exit procedure
16: end if
    Case 3 Integrity check
17: if
    Agent $\mathcal{A}$ does not know about integrity preservation, that is
        $\mathsf{neg}(\mathcal{IC}, \mathcal{U}) \not\subseteq \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}\rangle)$
    then
18:   if
      Agent $\mathcal{A}$ knows about integrity violation, that is
        $\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U}) \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}\rangle)$
      then
19:     $note$ = view update transaction violates integrity constraints
20:     Append $(\mathsf{up}(\mathcal{L}), (\mathcal{U}, note))$ to history
21:     Exit procedure
22:   else if
      Notification of integrity violation harms confidentiality, that is
        $\mathsf{ccp}(S, \Delta\mathcal{U}p[\mathcal{S}eq]) \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C} \cup \{\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})\}\rangle)$
            for an $S \in conf(CCP)$
                        or
        $S \in \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C} \cup \{\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})\}\rangle)$
            for an $S \in conf(TCP)$
      then
```

```
23:       $note$ = integrity check conflicts confidentiality
24:       Append $(\mathsf{up}(\mathcal{L}), (\mathcal{U}, note))$ to history
25:       Exit procedure
26:    else if
       Integrity violation, that is
           $IC \not\subseteq \mathsf{Bel}^{DB}_{\mathcal{D}}(\langle IC, db \bullet \mathcal{U}\rangle)$
        then
27:       $note$ = view update transaction violates integrity constraints
28:       $\mathcal{V} = \langle IC, \mathcal{C} \cup \{\mathsf{neg}(\neg \bigwedge(IC), \mathcal{U})\}\rangle$
29:       Append $(\mathsf{up}(\mathcal{L}), (\mathcal{U}, note))$ to history
30:       Exit procedure
31:    end if
32: end if
    Case 4 View Update Transaction
33: $note$ = view update transaction successful
34: $\langle IC, db\rangle = \langle IC, db\rangle \bullet \mathcal{U}$
35: $\mathcal{V} = \mathcal{V} \odot \mathcal{U}$
36: Append $(\mathsf{up}(\mathcal{L}), (\mathcal{U}, note))$ to history
```

Proposition 7.3 (Output States of Controlled View Update Transaction Execution)**.**

Procedure 2 outputs a state of Def. 7.1.

Case 1 bases on Assumption 7 that the attacking agent has to be informed about the effective updates $\mathcal{U}$. The assumption may be argued, but we defer a discussion to Section 7.6.2.

In *Case 2*, there is no need to check for meta-inferences: Passing *Case 1*, the set $\mathcal{U}$ of outstanding updates can be made visible to agent $\mathcal{A}$ so that, in principle, the agent is able to execute *Case 2* itself and, thus, the execution of *Case 2* by the control function releases no additional information to $\mathcal{A}$. Therefore agent $\mathcal{A}$ cannot exploit the execution of *Case 2* to draw meta-inferences.

In *Case 3*, the integrity check can be understood as the implicit query about $\mathcal{D}$'s current belief of formula $\mathsf{neg}(\neg \bigwedge(IC), \mathcal{U})$ saying that updating the database instance by $\mathcal{U}$ violates the integrity constraints. This query can be evaluated with the current database $\mathcal{db}$ due to the equivalence of the evaluation $\mathcal{db} \models_{PL} \mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})$ with the evaluation $\mathcal{db} \bullet \mathcal{U} \models_{PL} \mathcal{IC}$ by Lemma 6.3. The control function first checks whether agent $\mathcal{A}$ already knows the result of the integrity check by its current database view in line 17 and line 18. This is important to avoid refusal due to inconsistency in the premises of propositional entailment checked within line 22. Further, the control function checks whether a notification of integrity violation breaches confidentiality in line 22. This check is made independently of integrity violation or preservation with respect to the current database instance and update transaction to prevent the meta-inference that a refusal in *Case 3* occurs if and only if integrity is violated.

Although the integrity check may be seen as an implicit query about the current belief of formula $\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})$, in contrast to the control function for controlled query execution, Procedure 2 does not check in *Case 3* whether the answer $\neg\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})$ to this query reveals some confidential belief. The answer says that the integrity constraints $\mathcal{IC}$ are preserved after updating the database instance by $\mathcal{U}$. As for continuous confidential belief S, the control function already prevents such a violation in line 12 (due to $\mathsf{ccp}(S, \Delta\mathcal{Up}[\mathcal{Seq}])$). As for temporary confidential belief S, the answer would only reveal that agent $\mathcal{D}$ believes S before updating the database instance. But after updating, agent $\mathcal{A}$ is no longer certain that $\mathcal{D}$ believes S by the condition checked in line 12.

Example 7.2 (Controlled View Update Transaction Execution).

We consider a situation similar to that of Ex. 7.1. Agent $\mathcal{D}$ has the database instance $\mathcal{db}_1 = \{a, s_1, s_2\}$ under the integrity constraints $\mathcal{IC} = \{a \Rightarrow s_2,\ s_1 \vee s_2,\ a \Rightarrow \neg b\}$ with policy $conf(TCP) = \emptyset$ and $conf(CCP) = \{s_1 \wedge s_2\}$. Agent $\mathcal{A}$ may freely access the view $\mathcal{V}_1 = \langle \mathcal{IC}, \{a\}\rangle$. In this situation, $\mathcal{A}$ sends the request $\mathsf{up}(\langle \neg a, b\rangle)$.

Next, we follow the computations of Proc. 2 from Case 2 on.[4] *In Case 2, the outstanding updates are $\mathcal{U} = \{\neg a, b\}$ and we have $\mathcal{Seq} = \langle\{a, b\}\rangle$. The updated view $\mathcal{V}_1 \odot \mathcal{U}$ contains all the information in the set* $\mathsf{neg}(\mathcal{C}_1 \cup \mathcal{IC}, \mathcal{U}) \cup \mathcal{U} \cup \mathcal{IC}$.

[4] In Case 1, the evaluation of $\neg a$ and b in $\mathcal{db}_1$ follows from the view $\mathcal{V}_1$ so that the view is not modified.

In (12)*, Proc. 2 checks this set for revelation of confidential belief:*

$$\begin{aligned}
&\mathsf{neg}(\mathcal{C}_1 \cup \mathcal{IC}, \mathcal{U}) \cup \mathcal{U} \cup \mathcal{IC} \\
&= \{\neg a, \neg a \Rightarrow s_2,\ s_1 \vee s_2,\ \neg a \Rightarrow b\} \cup \mathcal{U} \cup \mathcal{IC} \\
&\equiv \{\neg a, s_2, b\} \cup \mathcal{IC} \\
&\nvdash_{pl} s_1 \wedge s_2 = \mathsf{ccp}(s_1 \wedge s_2, \Delta\mathcal{U}p[\mathcal{S}eq]).
\end{aligned}$$

After the negative check, in (17) *the procedure infers that $\mathcal{A}$ knows about integrity preservation:*

$$\mathcal{C}_1 \cup \mathcal{IC} = \{a, s_1 \vee s_2, \ldots\} \vdash_{pl} \{\neg a \Rightarrow s_2,\ s_1 \vee s_2,\ \neg a \Rightarrow b\} = \mathsf{neg}(\mathcal{IC}, \mathcal{U}).$$

Finally, the database instance is updated to $\mathcal{db}_2 = \{b, s_1, s_2\}$ and the updated view $\mathcal{V}_2$ may be simplified to the view $\langle \mathcal{IC}, \{\neg a, b, s_2\}\rangle$ containing the same information under skeptical *of* (7.3)*.*
Afterwards, $\mathcal{A}$ requests $\mathsf{up}(\langle \neg b, \neg s_2\rangle)$*. Again Case 1 does not modify the view. Procedure 2 in Case 2 checks for revelation of confidential belief with* $\mathcal{U} = \{\neg b, \neg s_2\}$*:*

$$\begin{aligned}
&\mathsf{neg}(\mathcal{C}_2 \cup \mathcal{IC}, \mathcal{U}) \cup \mathcal{U} \cup \mathcal{IC} \\
&\equiv \{\neg a, \neg b, \neg s_2\} \cup \mathcal{IC} \\
&\equiv \{\neg a, \neg b, \neg s_2, s_1 \vee s_2\} \vdash_{pl} s_1.
\end{aligned}$$

The continuous confidential belief $s_1 \wedge s_2$ must be protected by preventing the inference of $\mathsf{ccp}(s_1 \wedge s_2, \langle\{a\}, \{b, s_2\}\rangle) \equiv (s_1 \wedge s_2) \vee (s_1 \wedge \neg s_2) \equiv s_1$*. Thus, the procedure detects an inference and refuses the request in* (12)*.*

7.4.3 Correctness of Simulation and Confidentiality Enforcement

Before we verify that the two control functions ensure the confidentiality property of Definition 4.6, we will present a result needed for that verification. The essential means of the control functions to enforce confidentiality is the simulation of agent $\mathcal{A}$'s postulated reasoning $\mathcal{K}_\mathcal{A}$ about $\mathcal{D}$'s belief, using operator skeptical with view $\mathcal{V}$. The strategy of the presented control functions is to enforce confidentiality by ensuring the invariants of the following lemma.

Lemma 7.2 (No Secrecy Revelation through $\mathcal{V}$).

Proc. 1 and Proc. 2 maintain the following two invariants:

$$\begin{aligned} \mathsf{ccp}(S, \Delta\mathcal{U}p[s]) \notin \mathsf{skeptical}(\mathcal{V}) \qquad & \textit{if } S \in \mathit{conf}(CCP), \textit{ and} \\ S \notin \mathsf{skeptical}(\mathcal{V}) \qquad & \textit{if } S \in \mathit{conf}(TCP). \end{aligned}$$

In the remainder, we will argue that these invariants suffice for enforcing confidentiality if the control functions correctly simulate operator $\mathcal{K}_{\mathcal{A}}$ as defined in the following.

Definition 7.4 (Correctness of Simulation).

Let cexec *be a control function of Def. 4.3 with domain* $\mathcal{S}t^{DB} \times \mathcal{R}eq^{DB}$ *adhering to Asmp. 7. Further, let* $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ *be the system of defender* $\mathcal{D}$ *and attacker* $\mathcal{A}$ *defined by* cexec. *Then,* cexec *correctly simulates operator* $\mathcal{K}_{\mathcal{A}}$ *by means of the view* $\mathcal{V}$ *and operator* $\mathsf{skeptical}$ *iff the following equivalence holds. For all* $(r, m) \in \mathcal{PT}(\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}})$ *it holds*

$$\begin{aligned} & db \models_{PL} \mathsf{skeptical}(\mathcal{V}(r, m)) \\ & \quad \textit{iff} \end{aligned}$$

there exists $(r', m) \in \mathcal{PT}(\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}})$ *such that*

$$db = db(r', m) \qquad \textit{and} \qquad (r', m) \in \mathcal{K}_{\mathcal{A}}(r, m).$$

In the following, we will present two requirements on control functions which guarantee the correctness of simulation.
First, recall that any information that the attacker $\mathcal{A}$ has gained by initiating a request and partially observing the control function's output is represented in the attacker's local state in the model $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$, cf. Definition 4.5. For a correct simulation, necessarily, that information gain must be represented in the attacker view in a "sound" way. In the following definition, we will make this requirement explicit.

Definition 7.5 (Soundness of Attacker View).

Let cexec *be a control function of Def. 4.3 with domain* $\mathcal{S}t^{DB} \times \mathcal{R}eq^{DB}$. *Then,* cexec *maintains a sound attacker view* $\mathcal{V}$ *iff*

for every request $\theta \in \mathcal{Req}^{DB}$
for every two input states $s_{\mathfrak{D}}$ *and* $s'_{\mathfrak{D}}$ *of the form*

$$s_{\mathfrak{D}} = (\mathcal{IC}, \mathcal{db}_{t-1}, conf, \mathcal{H}^{DB}_{t-1}, \mathcal{V}_{t-1}) \in \mathcal{St}^{DB}$$
$$s'_{\mathfrak{D}} = (\mathcal{IC}, \mathcal{db}'_{t-1}, conf, \mathcal{H}^{DB}_{t-1}, \mathcal{V}_{t-1}) \in \mathcal{St}^{DB},$$

that is the states $s_{\mathfrak{D}}$ *and* $s'_{\mathfrak{D}}$ *differ only in the database instance employed, the following holds:*
If $\mathcal{A}$ *is able to distinguish* $s'_{\mathfrak{D}}$ *from* $s_{\mathfrak{D}}$ *after execution on state* $s_{\mathfrak{D}}$ *and request* θ*, that is* $(\mathcal{H}^{DB}_t)' \neq \mathcal{H}^{DB}_t$ *or* $\mathcal{V}'_t \neq \mathcal{V}_t$*, then it follows that there exists a set* $\mathcal{U}$ *of literals over pairwise distinct atoms such that*

$$\mathcal{db}_t = \mathcal{db}_{t-1} \bullet \mathcal{U} \qquad \textit{and} \qquad \mathcal{db}'_{t-1} \bullet \mathcal{U} \not\models_{PL} \mathsf{skeptical}(\mathcal{V}_t).$$

We will explain the definition in the following. Consider the initial situation of Definition 7.5 in which $\mathfrak{D}$ is in state $s_{\mathfrak{D}}$. Then, the attacker $\mathcal{A}$'s information gain after request θ in the considered situation is represented in the history $\mathcal{H}^{DB}_t$ and view $\mathcal{V}_t$ in its local state at time t in system $\mathcal{R}^{\mathsf{cexec}}_{\mathfrak{D},\mathcal{A}}$. With that additional information, $\mathcal{A}$ is able to rule out (by operator $\mathcal{K}_{\mathcal{A}}$) that agent $\mathfrak{D}$ employs the database instance $\mathcal{db}'_{t-1}$ if no database update is reported in $\mathcal{H}^{DB}_t$; or $\mathcal{db}'_{t-1} \bullet \mathcal{U}$ if a database update by $\mathcal{U}$ is reported.[5] Then, the database instance ruled out by $\mathcal{A}$ must not be a model of the instance view that is output by the control function in the considered situation. Or, to put it into a different perspective, every information that agent $\mathcal{A}$ has gained about the database instance by observing the visible output of the control function is represented in the attacker view.
The second requirement follows the principle of Assumption 3: The security engineer does not not suppose the attacking agent $\mathcal{A}$ to forget information already disclosed to it. The next definition requires a control function to implement this principle in the maintenance of the attacker view $\mathcal{V}$.

[5] The database instance is the only component of $\mathfrak{D}$'s local state of which agent $\mathcal{A}$ has no complete knowledge by Definition 4.5 and Definition 7.1. Hence, if $\mathcal{A}$ gains information, it gains information about the database instance. Further, the attacker's reasoning about database updates bases on Assumption 7.

Definition 7.6 (No Loss of Information).

Let cexec *be a control function of Def. 4.3 with domain* $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}$*.*
Then, cexec *does not loose information from view* $\mathcal{V}$ *iff*
for every input state $s \in \mathcal{St}^{DB}$ *with database instance* db_{t-1} *and view* $\mathcal{V}_{t-1}$*,*
for every request $\theta \in \mathcal{Req}^{DB}$ *with output state* $\mathsf{cexec}(s, \theta)$*, including database instance* db_t *and view* $\mathcal{V}_t$*, there exists a set* $\mathcal{U}$ *of literals over pairwise distinct atoms such that*

$$db_t = db_{t-1} \bullet \mathcal{U} \qquad \textit{and} \qquad \mathsf{skeptical}(\mathcal{V}_{t-1} \odot \mathcal{U}) \subseteq \mathsf{skeptical}(\mathcal{V}_t).$$

Finally, we show that the two requirements suffice for the correctness of simulation.

Proposition 7.4 (Correctness of Simulation).

Let cexec *be a control function of Def. 4.3 with domain* $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}$ *under Assumption 7. Further, let* $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ *be the system of defender* $\mathcal{D}$ *and attacker* $\mathcal{A}$ *defined by* cexec*. If* cexec *satisfies* ***(Soundness of Attacker View)*** *of Def. 7.5 and* ***(No Loss of Information)*** *of Def. 7.6, then* cexec *correctly simulates operator* $\mathcal{K}_{\mathcal{A}}$ *in the sense of Def. 7.4.*

Lemma 7.3 (Soundness of Attacker View).

The control functions Proc. 1 and Proc. 2 maintain a sound attacker view.

Lemma 7.4 (No Loss of Information).

The control functions Proc. 1 and Proc. 2 do not loose information from the view in the sense of Def. 7.6.

The last lemma holds by design of the procedures.
The following lemma is an immediate consequence of Proposition 7.4 and Lemma 7.3 and Lemma 7.4. We state it for the reader's convenience.

Lemma 7.5 (Correctness of Simulation by Procedures).

The control functions Proc. 1 and Proc. 2 correctly simulate operator $\mathcal{K}_{\mathcal{A}}$ *in the sense of Def. 7.4.*

Theorem 7.1 (Confidentiality Preservation).

The control functions Proc. 1 and Proc. 2 preserve continuous/temporary confidentiality towards agent $\mathcal{A}$ in the sense of Def. 4.6.

7.5 Cooperative Information Sharing

Usually, beside confidentiality, agent $\mathcal{D}$ is interested in sharing information with agent $\mathcal{A}$ as cooperatively as possible. Similar, in database security, the database should provide each user with information the user is authorized to access which serves the users' interest in the *availability of information*. To emphasize the fact that $\mathcal{D}$ is both interested in the provision and acquisition of information we will speak of cooperative information sharing.
In this section, we analyze the performance of the control function for view update transaction execution in Procedure 2 under $\mathcal{D}$'s interest of cooperative information sharing. The control functions of Section 7.4 respect this interest by a policy of *last-minute intervention*. Roughly, this policy says that a control function only refuses a request from agent $\mathcal{A}$ if otherwise there is a potential of a confidentiality policy violation. Further, the policy of last-minute intervention lets the requesting agent $\mathcal{A}$ choose during an interaction which information should be shared as needed by $\mathcal{A}$. The next example illustrates the principle behind last-minute intervention to share information as needed by $\mathcal{A}$ by the processing of controlled query execution of Procedure 1:

Example 7.3 (Information Sharing Under a Policy of Last-Minute Intervention).

Let $\mathcal{C} = \{medB \rightarrow (cancer \vee flu)\}$ be an instance view on the database instance $\mathcal{db} = \{cancer, medB\}$. Agent $\mathcal{D}$'s confidentiality policy comprises the sets $conf(TCP) = \emptyset$ and $conf(CCP) = \{cancer\}$. Since the interaction between the agents has just started, it holds $\Delta\mathcal{Up} = \langle\rangle$ and, hence, $\mathsf{ccp}(cancer, \Delta\mathcal{Up}) \equiv cancer$ holds. The query sequence $Q_1 = \langle \mathsf{que}(flu), \mathsf{que}(medB)\rangle$ is answered with $\langle \neg flu, refuse\rangle$, whereas $Q_2 = \langle \mathsf{que}(medB), \mathsf{que}(flu)\rangle$ is answered with $\langle medB, refuse\rangle$. Thus, by the order of its queries agent $\mathcal{A}$ chooses whether it is informed about either medB *or $\neg flu$ if it could not be informed about the two pieces of information due to confidentiality.*

Imagine another control function cexec' that answers the query sequence Q_1 with $\langle \mathit{refuse}, \mathit{medB}\rangle$. Clearly, this control function follows another policy for cooperative information sharing and does not let agent $\mathcal{A}$ choose the information to be shared by the order of its requests. In this situation, the policy for cooperative information sharing followed by cexec' and the policy of last-minute intervention obviously do not agree in their assumptions about agent $\mathcal{A}$'s information needs. Certainly, control functions designed under different assumptions about agent $\mathcal{A}$'s information needs cannot be compared with respect to information sharing, but only those assumptions might be argued. We do so in Subsection 7.6.2.
Under a policy of last-minute intervention, agent $\mathcal{D}$ is interested to satisfy agent $\mathcal{A}$'s information needs according to the requests received from $\mathcal{A}$. The decision of a control function for a request depends on the information previously shared among the agents. For this reason, we will compare control functions on the same input for a single request, especially, on the same input view provided to $\mathcal{A}$. In particular, in the analysis of cooperative information sharing we do not consider sequences of requests.
Further, our analysis focuses on controlled view update transaction execution. Thus, within this section we restrict valid request messages to the set

$$\begin{aligned}\mathcal{R}eq^{DB}_{\mathsf{up}} = \{&\mathsf{up}(\{L_1,\ldots,L_n\}) \mid \\ &L_1,\ldots,L_n \in \mathcal{L}_{PL} \text{ literals over pairwise distinct atoms}\}.\end{aligned} \tag{7.5}$$

Further, for simplicity in the remainder of this section, we assume that, before a view update transaction request $\mathsf{up}(\mathcal{L})$, agent $\mathcal{A}$ has been informed for each literal $L \in \mathcal{L}$ whether agent $\mathcal{D}$ believes L or not – if permitted by $\mathcal{D}$'s confidentiality policy.[6] Otherwise, we will not consider the request $\mathsf{up}(\mathcal{L})$ in the analysis. The information should be contained in the instance view $\mathcal{C}$, formally,

$$(\mathcal{L} \setminus \mathcal{U}) \cup \neg(\mathcal{U}) \subseteq \mathcal{C} \quad \text{with } \mathcal{U} = \mathcal{L} \setminus \mathsf{Bel}^{DB}_{\mathcal{D}}(\langle \mathcal{IC}, \mathcal{db}\rangle). \tag{7.6}$$

[6] Note that every control function basing on Assumption 7 will eventually reveal this information to $\mathcal{A}$ if confidentiality will not be violated.

The main challenge now is to define cooperativeness of control functions under a policy of last-minute intervention or, in more general terms, the desired processing of requests by control functions under this policy. Under this policy, a request expresses the requesting agent $\mathcal{A}$'s need of sharing information. In particular, requesting a view update transaction by $\mathtt{up}(\mathcal{L})$ agent $\mathcal{A}$ is interested to share information as follows. First, it offers the information that the literals $\mathtt{up}(\mathcal{L})$ are true in the agents' environment. Second, it demands to be informed about whether the database update has succeeded or failed. Third, in case of failure it demands to be informed about the cause of the failure in order to take appropriate action. In particular, if a potential violation of confidentiality is the cause of the failure, $\mathcal{A}$ does not need to be informed that integrity would be violated if the update was executed. It is important to note that agent $\mathcal{A}$ by request $\mathtt{up}(\mathcal{L})$ does not need to share information apart from that (although as an attacker it might intend to obtain more information through the request). Any control function of Definition 4.3 shares information with agent $\mathcal{A}$ via the components $\mathcal{H}^{DB}$ and $\mathcal{V}$ in agent $\mathcal{D}$'s local state. In our analysis, we focus on the attacker view $\mathcal{V}$ since it has a formal semantics with respect to database instances. The requirement of **(Soundness of Attacker View)** of Definition 7.5 ensures that actually $\mathcal{A}$ has not gained information about the database instance beyond what is represented in the attacker view. In the following definition, we list requirements on control functions under a policy of last-minute intervention.

Definition 7.7 (Proper Control Functions for Controlled View Update Execution)**.**

Let cexec *be a control function of Def. 4.3 with domain $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}_{\mathtt{up}}$ fulfilling Asmp 7. Then, the control function is called* proper *if it is deterministic, follows the atomicity and consistency (here, integrity) preservation requirement (ACID) and has the following properties for each input request and input state*

$$s_{\mathcal{D}} = (\mathcal{IC}, \mathcal{db}_{t-1}, conf, \mathcal{H}^{DB}_{t-1}, \mathcal{V}_{t-1}) \in \mathcal{St}^{DB} \textit{ of Def. 7.1.}^{7}$$

1. ***No Loss of Information****: (Def. 7.6)*

2. ***Cooperativeness:***

   If the updated database view $\mathcal{V}_{t-1} \odot \mathcal{U}$ contains confidential information as defined in line 12 of Proc. 2, then

   $$\mathsf{skeptical}(\mathcal{V}_t) \subseteq \mathsf{skeptical}(\mathcal{V}_{t-1}),$$

   else if the update by $\mathcal{U}$ of (7.6) *is not effective, then*

   $$\mathsf{skeptical}(\mathcal{V}_t) \subseteq \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}_{t-1} \cup \{\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})\}\rangle),$$

   else

   $$\mathsf{skeptical}(\mathcal{V}_t) \subseteq \mathsf{skeptical}(\mathcal{V}_{t-1} \odot \mathcal{U}).$$

3. ***Soundness of Attacker View*** *(Def. 7.5)*

4. ***Confidentiality*** *(Def. 4.6).*

These properties meet the following objectives:
The *domain/co-domain of control function* ensures by Definition 7.1 that the function has to maintain the attacker view $\mathcal{V}_t$ as a database view. Therefore, every formula in the instance view $\mathcal{C}_t$ is true in the database instance and $\mathcal{A}$ is not misled by the information revealed to it. Further, Assumption 7 in particular ensures that $\mathcal{A}$ is truthfully notified of the success/failure of the update which is recorded in the history.
The property *cooperativeness* defines which information the control function should provide to $\mathcal{A}$ according to the agent's request $\mathsf{up}(\mathcal{L})$ and requires the control function to provide no other information. The definition accounts for $\mathcal{A}$'s information needs that have been intuitively described above as follows. Without a successful update agent $\mathcal{A}$ may not obtain any additional information if updating the database view violates confidentiality. Lemma 7.6 below shows that in such a case a successful update necessarily violates the confidentiality requirement of Definition 4.6. Otherwise, without a successful update $\mathcal{A}$ may only be informed about a failed integrity check while $\mathcal{C}_{t-1}$ is its

[7]Here, the history is only defined on the set $\mathcal{Req}^{DB}_{\mathrm{up}}$ of valid view update requests, not on $\mathcal{Req}^{DB}$ as in the definition.

instance view before the request. With a successful update, agent $\mathcal{A}$ may only be informed about the success of the operation and the set $\mathcal{U} \subseteq \mathcal{L}$ defined by the property $\mathcal{db}_{t-1} \bullet \mathcal{U} = \mathcal{db}_t$ by Assumption 7. Given all that information, agent $\mathcal{A}$ can determine the updated instance view $\mathcal{C}_{t-1} \odot \mathcal{U}$ itself.
The properties *no loss of information*, *soundness of attacker view* and *confidentiality* have been motivated in the context of the respective definitions.

Proposition 7.5 (Proper View Update Transactions).

The control function Proc. 2 is proper in the sense of Def. 7.7.

Atomicity, no loss of information and cooperativeness are ensured by design of Procedure 2. The other properties are shown in Proposition 7.3, Lemma 7.3 and Theorem 7.1.
The following property of *local optimality* must be viewed in the light of Definition 7.7 which requires a proper control function not to reveal more information than requested by $\mathcal{A}$ by the property of cooperativeness.

Definition 7.8 (Local Optimality).

A proper control function cexec *is said to be locally optimal*[8] *iff for every other proper control function* cexec′ *the following properties hold.*

- $\Delta^{\mathsf{cexec}'} \subset \Delta^{\mathsf{cexec}}$*:* cexec′ *performs strictly less updates than* cexec*,or*
- $\Delta^{\mathsf{cexec}'} = \Delta^{\mathsf{cexec}}$ *and* $\mathsf{skeptical}(\mathcal{V}^{\mathsf{cexec}'}) \subseteq \mathsf{skeptical}(\mathcal{V}^{\mathsf{cexec}})$*:* cexec′ *performs the same updates as* cexec*, but offers at most the information provided by* cexec*.*

Here, $\Delta^{\mathsf{cexec}} \subseteq \mathsf{At}(\mathcal{U})$ *denotes the atoms updated by procedure* cexec *and* $\mathcal{V}^{\mathsf{cexec}}$ *the view output by* cexec*.*

The following lemma helps to prove local optimality of the control function cexec implemented by Procedure 2. It essentially states that the conditions checked by cexec are necessary for guaranteeing confidentiality for any control function on $\mathcal{St}^{DB}$ of Definition 7.1 under Assumption 7.

[8]We speak of local optimality instead of optimality to emphasize that we consider only single requests instead of sequences due to a policy of last-minute intervention.

Lemma 7.6 (Revelation of Confidential Belief through the Database View)**.**

Let cexec *be a control function of Def. 4.3 with domain* $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}_{\mathsf{up}}$ *fulfilling Asmp. 7. Let the view* $\mathcal{V}_t$ *in the output state* s_t *contain confidential information, that is there exists* $S \in \mathcal{L}_{PL}$ *such that*

$$\begin{array}{rlcl} & (S \in \mathsf{skeptical}(\mathcal{V}_t) & \textit{and} & S \in \mathit{conf}(TCP)) \\ \textit{or} & (\mathsf{ccp}(S, \Delta\mathcal{Up}[s_t]) \in \mathsf{skeptical}(\mathcal{V}_t) & \textit{and} & S \in \mathit{conf}(CCP)). \end{array}$$

Then, cexec *violates the confidentiality requirements of Def. 4.6.*

Theorem 7.2 (Local Optimality of Procedure 2)**.**

Procedure 2 for controlled view update execution is locally optimal.

7.6 Conclusion

7.6.1 Summary

We presented two control functions for iterated query and view update transaction requests and proved their effectiveness in preserving confidentiality. The control functions achieve confidentiality by two activities which we discussed in this chapter. The first activity is the preservation of the invariant that the attacker view does not imply any propositional information that is relevant for the protection of confidential belief. To this end, we showed that continuous confidential belief $S \in \mathcal{L}_{PL}$ can be protected by a propositional formula that is the result of a transformation of formula S. The second activity is the maintenance of the attacker view. We introduced two requirements on this activity that in combination ensure that the control function correctly simulates $\mathcal{A}$'s reasoning. Finally, the invariant and the correctness of simulation imply confidentiality preservation which we proved in Theorem 7.1. Beyond confidentiality, we studied agent $\mathcal{D}$'s cooperativeness in sharing information. The dynamic enforcement of confidentiality by simulating the attacker $\mathcal{A}$ at runtime usually is complemented by a policy of last-minute intervention (within the research of Controlled Interaction Execution [18]). In this chapter, we made the objective of this policy precise mainly by two definitions. First, cooperativeness defines agent $\mathcal{A}$'s information needs expressed by the agent through a view update transaction request. Second, local optimality defines optimal control functions under a policy of last-minute intervention. Finally, we proved that the control function for view update transaction achieves local optimality.

7.6.2 Discussion

Notification About Effective Updates Rather than deciding to let the control function for view update transaction execution exit in Case 1, we could try to find an alternative way to let the function hide (parts of) the set $\mathcal{U}$ of effective updates (Assumption 7) from agent $\mathcal{A}$. Doing so, we might at best weaken Assumption 7 to the assumption that agent $\mathcal{A}$ is just reported the failure or success of the transaction or, in formal terms, that $\mathcal{A}$ is able to determine whether $\mathcal{db}(r, m) \models_{PL} \mathcal{L}$ holds. However, then, we would face the following problem. The control function uses the set $\mathcal{U}$ in the integrity check and for the view update $\mathcal{V} \odot \mathcal{U}$. Thus, for example, being informed about the result of the integrity check, agent $\mathcal{A}$ might *implicitly* acquire information about whether or not an input literal $L \in \mathcal{L}$ is contained in the set $\mathcal{U}$. Preventing such disclosures presumably would lead to significantly more complicated inference checks in Case 2 and Case 3 of the control function in Procedure 2.

Preprocessing Approaches A drawback of the dynamic approach of simulating the attacker is the high computational cost of the employed propositional entailment operator combined with the increasing size of the view as well as the increasing length of the history for computing the sequence of effective updates. Alternatively, the defender $\mathcal{D}$ may create a precomputed inference-proof database view for agent $\mathcal{A}$, but then is recommended to define an availability policy that estimates $\mathcal{A}$'s information needs [36, 35]. As another solution with preprocessing, but on the schema rather than on the data level, the authors of [40] split a relation into vertical views (fragments, i.e., several columns of the original relation) in order to break sensitive associations among the attributes, but inference based on integrity constraints is not considered in the seminal works on fragmentation. In [26], the reader may find a first investigation of the inference problem for fragmentation in the presence of integrity constraints. Yet, the control of the attacker's inference upon updating the preprocessed view or the data in a fragment is not treated by the respective work.

Availability Policies Throughout this chapter, we mentioned other approaches treating the aspect of cooperative information sharing. Several approaches do not give an estimate of the requesting agent's information needs, but follow the principle of minimal loss of information such as a minimal distortion of database entries in [36] or maximized data utility as surveyed in [51, 2]. These approaches have the disadvantage that a measure for the loss of information must be found that is appropriate for the later usage of the data such as a particular data-mining task. Other approaches define the requester's information needs by an explicit policy such as pieces of information that should not be distorted in [36, 35] or lower bounds on the classification of data in [44]. The disadvantage here is that the requester's information needs must be specified at the configuration time of the database. The policy of last-minute intervention does not have these disadvantages at configuration time. Rather it faces the problem of high computational costs at runtime due to the dynamic control of inferences.

7.6.3 Future Work

Local optimality should be analyzed for control functions for query execution. To this end, the information needs that agent $\mathcal{A}$'s expresses by request $\mathtt{que}(A)$ with formula $A \in \mathcal{L}_{PL}$ should be defined similar to **(Cooperativeness)** of Definition 7.7. This should simply be the requirement

$$\begin{aligned}&\mathsf{skeptical}(\mathcal{V}_t) \subseteq \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}_{t-1} \cup \{A\}\rangle) \qquad \text{or}\\ &\mathsf{skeptical}(\mathcal{V}_t) \subseteq \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}_{t-1} \cup \{\neg A\}\rangle).\end{aligned}$$

A control function might have the option to weaken a query answer A or $\neg A$, for example, weaken $a \wedge b$ with $a, b \in \mathcal{A}t$ to a. This is a more cooperative way to answer queries in the sense of local optimality of Definition 7.8 under the latter requirement.

In our analysis of local optimality, Definition 7.7 lists requirements of control functions for view update transaction execution and in particular stipulates the use of deterministic control functions. In future work, we might consider control functions that are non deterministic. We may argue that a potential conflict between confidentiality and integrity in a transaction must not force a protocol to refuse this transaction if the protocol can randomly choose to refuse this transaction. But then we need to augment the representation of agent $\mathcal{A}$'s information with probabilities.
Another open aspect is a comparison of local optimality of the presented control functions with control functions applying data distortion, that is lying such as [22] or cover stories such as [52]. At first sight, our approach seems to pay the rather high price of forcefully refusing a transaction with a potential conflict between integrity and confidentiality. Yet, lying or polyinstantiation provide misinformation, which obviously reduces the availability of correct or reliable information, so that a more thorough comparison is needed. For such a comparison, we would have to modify Definition 7.7 and Definition 7.8 in order to account for pretended success of an update or misinformation in the view.

Chapter 8

Revising Belief by Nonmonotonic Reasoning without Revealing Secrets

In this chapter, we study the defending agent $\mathcal{D}$'s controlled processing of iterated query and revision requests from the attacking agent $\mathcal{A}$, cf. Section 6.2. In contrast to Chapter 7, agent $\mathcal{D}$ is not able to gather complete information about its environment. Rather, agent $\mathcal{D}$ holds incomplete information, its set as of current assertions. From these assertions $\mathcal{D}$ derives its belief in the process of nonmonotonic reasoning with its consequence relation $\mathrel{|\!\sim}_{\mathcal{D}}$. To protect its confidential belief, agent $\mathcal{D}$ might hide parts of its assertions as and its consequence relation $\mathrel{|\!\sim}_{\mathcal{D}}$ from the attacking agent $\mathcal{A}$. The main goal of this chapter is the construction of a confidentiality preserving agent $\mathcal{D}$ as outlined in Figure 8.1. The following example illustrates the main challenges faced in the construction.

Example 8.1 ($\mathcal{A}$ Inquiring $\mathcal{D}$'s Belief Through Interactions).

Based on the running example of Sect. 6.2.2, we consider that manufacturer $\mathcal{D}$'s confidentiality interest is to hide from supplier $\mathcal{A}$ whether $\mathcal{D}$ currently believes either of the two following propositions: first, a positive result of the feasibility study and, second, the agreement of deployment with supplier #2. Altogether, $\mathcal{D}$'s policy is $conf(TCP) = \{r, d12, d22, d32, d42\}$[1]. $\mathcal{D}$ initially has

[1] Note that $\mathcal{D}$'s policy declaration should not include just the agreements actually settled

the assertions $\mathit{as} = \{d12, s12\}$ *according to its agreement with supplier #2, but has hidden from* $\mathcal{A}$ *the assertion* $d12$ *due to its confidentiality policy.*
Consider now two scenarios in which $\mathcal{A}$ *tries to find out whether* $\mathcal{D}$ *currently believes a formula in* $\mathit{conf}(\mathit{TCP})$. *In the first scenario,* $\mathcal{A}$ *is interested in the realizability of the manufacturer's proposal. In this scenario the history of the interaction is*

$$\mathcal{H}_1^{NMR} = \langle(\mathtt{que}(s21), \mathit{undef}),\ (\mathtt{que}(s21 \Rightarrow r), \mathit{true}),\ (\mathtt{rev}(s21), \mathit{success})\rangle.$$

In the second scenario, $\mathcal{A}$ *is interested in the manufacturer's agreement with supplier #2 and, thus, interacts with* $\mathcal{D}$ *as follows:*

$$\mathcal{H}_2^{NMR} =$$
$$\langle(\mathtt{rev}(\neg s11 \vee \neg s21), \mathit{success}),\ (\mathtt{que}(\neg s11 \Rightarrow d12), \mathit{true}),\ (\mathtt{rev}(\neg s21), \mathit{failure})\rangle.$$

Intuitively, in the first scenario after the successful revision, with the query answers $\mathcal{A}$ should be able to reason that $\mathcal{D}$ finally believes r. In the second scenario, the success and the failure notification helps $\mathcal{A}$ to conclude that $\mathcal{D}$ believes $\neg s11 \wedge s21$ as $\mathcal{D}$'s conclusion from $\mathcal{D}$'s hidden unquestionable belief and $\mathcal{D}$'s hidden assertions. Together with the query result, $\mathcal{A}$ might conclude that $\mathcal{D}$ believes $d12$. The main challenge is now to compute $\mathcal{A}$'s reasoning. To this end, we investigate the following questions

1. What has $\mathcal{A}$ learned about the hidden consequence relation $\mathrel{|\!\sim}_{\mathcal{D}}$, in particular, $\mathcal{D}$'s unquestionable belief, and the hidden parts of the assertions as from the history of an interaction?

2. For which classes of $\mathcal{D}$'s fixed nonmonotonic reasoning may we compute $\mathcal{A}$'s reasoning about $\mathcal{D}$'s belief?

3. How is $\mathcal{A}$ able to handle the fact that $\mathcal{D}$ might have hidden some assertions from $\mathcal{A}$?

In the following, we will survey our contributions to the construction of agent $\mathcal{D}$ in Figure 8.1. The first and main contribution is the computation of $\mathcal{A}$'s

but rather any possible agreement with supplier #2. Otherwise, the agreement would be revealed to $\mathcal{A}$ because $\mathcal{A}$ knows the declaration of the policy by Assumption 2. Here, we assume that $d12, d22, d32$ and $d42$ are all possible agreements with supplier #2.

postulated skeptical reasoning in the presence of hidden assertions. This contribution also presents solutions to the previous questions.

In this chapter, we do not consider continuous confidential belief (formally, we assume $\mathit{conf}(CCP) = \emptyset$). Thus, $\mathcal{D}$'s control procedures simulate agent $\mathcal{A}$'s postulated reasoning with $\mathcal{K}_{\mathcal{A}}$ about $\mathcal{D}$'s *current* belief to protect *temporary confidential belief.* The simulation is based on the attacker view $\mathcal{V}$. As part of this simulation, the control procedures incorporate what $\mathcal{A}$ has learned from outgoing notifications into the view by invoking respective view maintenance procedures (question 1).

We consider classes $\mathcal{C}$ for $\mathcal{D}$'s reasoning with an axiomatization satisfying dedicated “allowance properties” (question 2). Several important axiomatizations from the literature on nonmonotonic reasoning have these properties.

The simulation of $\mathcal{A}$'s postulated skeptical reasoning is by deduction with such an axiomatization $\mathbf{Ax}$ of the class of $\mathcal{D}$'s reasoning and a translation of the view into formulas of a conditional language. The ”allowance properties” of the axiomatization enable $\mathcal{A}$ to reason about $\mathcal{D}$'s belief in the situation of hidden situation as if $\mathcal{D}$ had not hidden any assertion at all (question 3).

The second contribution is the design of the control functions for iterated query and revision requests and a proof of their effectiveness. During simulation, the control procedures ensure that $B \notin \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V})$ for every $B \in \mathit{conf}(TCP)$ lest confidendiality is immediately breached. Like previously in Chapter 7, the main challenge to designing the control functions of the censor is the prevention of harmful meta-inferences from refusal notifications. Controlled revision execution and controlled view update transaction execution both base success/failure notifications on a check whether the information received through the request from $\mathcal{A}$ conflicts with $\mathcal{D}$'s background knowledge[2]. For this reason, the control function for revision execution can follow a similar strategy against meta-inferences like that of the control function for view update transaction execution.

Finally we remark that in this chapter we will not provide a formal analysis of the control functions' performance under information sharing. Instead we will

[2]In Section 6.4, we outlined the similarity in the processing of a database update and the processing of a revision in more detail.

discuss information sharing in Section 8.6 as an outlook to future work.
The organization of this chapter is as follows. Section 8.1 to Section 8.4 present the first contribution. First, Section 8.1 studies how $\mathcal{A}$ might approximate $\mathcal{D}$'s hidden consequence relation $\mathrel{|\!\sim}_{\mathcal{D}}$ and the partially hidden assertions as by observing the history of an interaction (question 1). Based on these approximations, the section elaborates on the simulation of agent $\mathcal{A}$'s postulated reasoning in the situation that $\mathcal{D}$ has hidden assertions from $\mathcal{A}$. Afterwards, Section 8.2 specifies $\mathcal{A}$'s awareness about $\mathcal{D}$'s implementation (Activity 2 of the security engineering task of implementation & verification). This way, we make precise how $\mathcal{A}$ might exploit the details of $\mathcal{D}$'s implementation in the postulated reasoning about $\mathcal{D}$'s belief in Chapter 4. Section 8.3 characterizes the class $\mathcal{C}$ of $\mathcal{D}$'s fixed reasoning by axiomatizations that capture its characteristic properties and satisfy dedicated "allowance properties" (question 2). These "allowance properties" are syntactic restrictions for such axiomatizations and give a basis for the results on the computation of skeptical entailment and the effectiveness of the censor in the subsequent sections.
In Section 8.4, we present results for the computation of the simulation of $\mathcal{A}$'s postulated reasoning as outlined in Figure 8.1 (question 3).
Finally, Section 8.5 presents the second contribution, the control functions of the censor and a formal proof of confidentiality preservation by means of these functions (Activity 3 of the security engineering task of implementation & verification). In Section 8.6, we will further discuss our contributions of this chapter and further open issues for future work.

8.1 Modeling $\mathcal{A}$'s Postulated Skeptical Reasoning For Simulation

8.1.1 Skeptical Entailment with Visible Assertions

As $\mathcal{D}$'s essential means to enforcing confidentiality, we now elaborate the attacking agent $\mathcal{A}$'s *postulated* reasoning of Chapter 4 for the simulation by $\mathcal{D}$. There, $\mathcal{A}$ is postulated to be a skeptical reasoner about $\mathcal{D}$'s belief opposing $\mathcal{D}$'s confidentiality aim expressed by $\mathcal{D}$'s confidentiality policy. In this chapter,

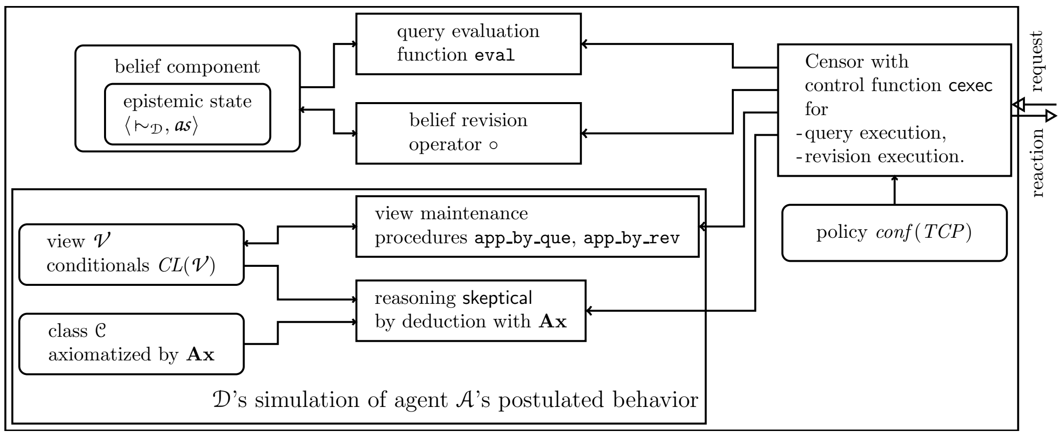

belief component
epistemic state
$\langle \mathrel{|\!\!\sim}_{\mathcal{D}}, \mathit{as} \rangle$
query evaluation
function eval
belief revision
operator $\circ$
Censor with
control function cexec
for
- query execution,
- revision execution.
request
reaction
policy $\mathit{conf}(TCP)$
view $\mathcal{V}$
conditionals $CL(\mathcal{V})$
class $\mathcal{C}$
axiomatized by $\mathbf{Ax}$
view maintenance
procedures app_by_que, app_by_rev
reasoning skeptical
by deduction with $\mathbf{Ax}$
$\mathcal{D}$'s simulation of agent $\mathcal{A}$'s postulated behavior

Figure 8.1: Design of a confidentiality preserving agent $\mathcal{D}$ as defender against a requesting agent $\mathcal{A}$ as attacker

the policy is limited to temporary confidential belief. Agent $\mathcal{A}$'s starting point for reasoning about $\mathcal{D}$'s belief are the requests sent by $\mathcal{A}$ and the reactions observed subsequently, cf. Example 8.1. $\mathcal{A}$'s observations during an interaction are recorded in the history of Definition 4.1.
To protect its confidential belief, agent $\mathcal{D}$ generally hides parts of its current assertions as and its fixed reasoning $\mathrel{|\!\!\sim}_{\mathcal{D}}$. Yet, for reasoning about $\mathcal{D}$'s belief, agent $\mathcal{A}$ (is assumed) to try to approximate these two components on the basis of the history of the agents' interaction:

- $\mathcal{A}$ approximates $\mathcal{D}$'s fixed reasoning $\mathrel{|\!\!\sim}_{\mathcal{D}}$
  (1) by a set $\mathcal{B}^+$ of pairs $(A, B) \in \mathcal{L}_{PL} \times \mathcal{L}_{PL}$ with the meaning that $\mathcal{A}$ *considers* that $A \mathrel{|\!\!\sim}_{\mathcal{D}} B$ holds, describing $\mathcal{D}$'s observed behavior to draw conclusions;[3]
  (2) by a set $\mathcal{B}^-$ of pairs $(A, B) \in \mathcal{L}_{PL} \times \mathcal{L}_{PL}$ with the meaning that $\mathcal{A}$ *considers* that $A \mathrel{|\!\!\not\sim}_{\mathcal{D}} B$ holds, describing $\mathcal{D}$'s observed behavior not to draw conclusions.[3]

- $\mathcal{A}$ approximates $\mathcal{D}$'s current assertions as by a set $\mathcal{C} \subset_{\mathit{fin}} \mathcal{L}_{PL}$ that describes which of the assertions propositionally entailed by as are visible to $\mathcal{A}$.[3]

[3]$\mathcal{B}$ refers to **b**ackground knowledge ($\mathrel{|\!\!\sim}_{\mathcal{D}}$) and $\mathcal{C}$ refers to **c**ontextual knowledge (as, also called evidential knowledge).

We abbreviate $\mathcal{B} = \mathcal{B}^+ \cup \mathcal{B}^-$. The next assumption simplifies our considerations in the remainder of the chapter.

Assumption 8 (Attacker's Ignorance of Fixed Reasoning).

Agent $\mathcal{A}$ is initially ignorant about $\mathcal{D}$'s fixed reasoning: $\mathcal{B}_0 = \emptyset$.[4]

But $\mathcal{C}_0$ is not assumed to be empty. For example, both the manufacturer $\mathcal{D}$ and $\mathcal{A}$ on behalf of supplier #1 might know that supplier #2 can supply part 1 (thus, $s12 \in \mathcal{C}_0$) because it is a publicly known fact.
For the time being, in this subsection we consider that agent $\mathcal{A}$ approximates $\mathcal{D}$'s components from its observations in the situation that the assertions as are visible to $\mathcal{A}$. In this situation, due to the open design (Assumption 2), during an uncontrolled interaction agent $\mathcal{A}$ is aware that $\mathcal{D}$ replies to a query $\texttt{que}(A)$ with $A \in \mathcal{L}_{PL}$ with answer *true* iff $\bigwedge(\mathit{as}) \mathrel{|\!\sim}_{\mathcal{D}} A$ holds by Definition 6.8 and (6.4). The same argument applies when $\mathcal{A}$ is notified by $\mathcal{D}$ that a revision $\texttt{rev}(A)$ with $A \in \mathcal{L}_{PL}$ has failed which is the case iff $\bigwedge(\mathit{as}) \wedge A \mathrel{|\!\sim}_{\mathcal{D}} \bot$ holds by Definition 6.9. Similarly, $\mathcal{A}$ may construct approximations from the visible assertions as and other reactions from $\mathcal{D}$.

Example 8.2 (Approximations).

Consider the interaction of the first scenario from Ex. 8.1 with history $\mathcal{H}_1^{NMR}$, but with $\mathcal{D}$'s initial assertions $\mathit{as} = \{s12\}$. Assume that as is visible to $\mathcal{A}$ so that $\mathcal{C}_0 = \{s12\}$. Then, $\mathcal{A}$ finds out the following from $\mathcal{H}_1^{NMR}$:

$$\begin{aligned}
\textit{Answer undef to } \texttt{que}(s21)\text{:} \quad & \mathcal{B}^- := \{(s12, s21),\ (s12, \neg s21)\}. \\
\textit{Answer true to } \texttt{que}(s21 \Rightarrow r)\text{:} \quad & \mathcal{B}^+ := \{(s12, s21 \Rightarrow r)\}. \\
\textit{Success notification after } \texttt{rev}(s21)\text{:} \quad & \mathcal{B}^- := \mathcal{B}^- \cup \{(s12 \wedge s21, \bot)\}, \\
& \mathcal{C} := \mathcal{C}_0 \cup \{s21\}.
\end{aligned}$$

During an interaction, under the current assumption of visible assertions, $\mathcal{C} = \mathit{as}$, agent $\mathcal{A}$ may refine its approximations on each observed reaction by the procedures in Table 8.1. In this situation, $\mathcal{B}^-$ approximates $\mathrel{|\!\sim}_{\mathcal{D}}$ from above ($\mathcal{B}^- \subseteq \mathrel{|\!\not\sim}_{\mathcal{D}}$) and $\mathcal{B}^+$ approximates $\mathrel{|\!\sim}_{\mathcal{D}}$ from below ($\mathcal{B}^+ \subseteq \mathrel{|\!\sim}_{\mathcal{D}}$). The

[4] As a consequence, agent $\mathcal{D}$'s unquestionable belief is not a priori known to $\mathcal{A}$ whereas in Chapter 7 we assumed that $\mathcal{A}$ knows the integrity constraints IC by Definition 4.2.

Procedure 3 `app_by_que`

```
Input: query $A$, query answer $ans$, approximations $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$
Output: refined approximations
1: if      $ans = true$    then return $\langle \mathcal{B}^+ \cup \{(\bigwedge(\mathcal{C}), A)\}, \mathcal{B}^-, \mathcal{C} \rangle$
2: else if $ans = false$   then return $\langle \mathcal{B}^+ \cup \{(\bigwedge(\mathcal{C}), \neg A)\}, \mathcal{B}^-, \mathcal{C} \rangle$
3: else if $ans = undef$   then return $\langle \mathcal{B}^+, \mathcal{B}^- \cup \{(\bigwedge(\mathcal{C}), A), (\bigwedge(\mathcal{C}), \neg A)\}, \mathcal{C} \rangle$
```

Procedure 4 `app_by_rev`

```
Input: revision $A$, notification $note$, approximations $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$
Output: refined approximations
1: if      $note = success$  then return $\langle \mathcal{B}^+, \mathcal{B}^- \cup \{(\bigwedge(\mathcal{C}) \wedge A, \bot)\}, \mathcal{C} \cup \{A\} \rangle$
2: else if $note = failure$  then return $\langle \mathcal{B}^+ \cup \{(\bigwedge(\mathcal{C}) \wedge A, \bot)\}, \mathcal{B}^-, \mathcal{C} \rangle$
```

Table 8.1: Procedures for refining approximations by observations

correct approximation in the situation of visible assertions is shown in Lemma 8.3 of the appendix.

With its approximations and the class $\mathfrak{C}$ characterizing $\mathfrak{D}$'s reasoning $\mid\!\sim_{\mathfrak{D}}$, agent $\mathcal{A}$ may skeptically conclude parts of $\mathfrak{D}$'s belief as follows. Above all, due to the open design by Assumption 2, $\mathcal{A}$ is aware that $\mathfrak{D}$'s fixed reasoning is a consequence relation within the class $\mathfrak{C}$. Moreover, $\mathcal{A}$ can rule out all consequence relations by which $\mathfrak{D}$ would reason differently from the already observed reasoning behavior (described in $\mathcal{B}^+$ and $\mathcal{B}^-$). Finally, $\mathcal{A}$ is aware that $\mathfrak{D}$ never considers its current assertions contradictory so that $\mathcal{A}$ rules out all consequence relations that infer $\bot$ from $\mathcal{C}$. Then, $\mathcal{A}$ determines which beliefs $\mathfrak{D}$ can derive in all the remaining consequence relations from (the visible part $\mathcal{C}$ of) its current assertions. The remaining consequence relations are referred to as possible consequence relations.

Definition 8.1 (Possible Consequence Relation)**.**

Let $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ be approximations and $\mathfrak{C}$ a class of consequence relations. Then, a consequence relation $\mid\!\sim' \subseteq \mathcal{L}_{PL} \times \mathcal{L}_{PL}$ is possible under $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ and $\mathfrak{C}$ iff the following holds:

($\mathcal{C}$ Defined) $\quad \mathrel{|\!\sim}' \in \mathcal{C}$.
($\mathcal{B}$ Compatible) $\quad \mathcal{B}^+ \subseteq \mathrel{|\!\sim}'$ *and* $\mathcal{B}^- \subseteq \mathrel{|\!\not\sim}'$.
(C Consistent) $\quad \bigwedge(C) \mathrel{|\!\not\sim}' \bot$.

Surely, $\mathcal{A}$ expects that there is at least one consequence relation within class $\mathcal{C}$ that, together with the visible assertions C, explains $\mathcal{D}$'s reactions described by $\mathcal{B}$, or in terms of the previous definition, that there exists a possible consequence relation under $\langle \mathcal{B}^+, \mathcal{B}^-, C \rangle$ and class $\mathcal{C}$.

Definition 8.2 (Skeptical Entailment)**.**

Let $\langle \mathcal{B}^+, \mathcal{B}^-, C \rangle$ be approximations and $\mathcal{C}$ a class of consequence relations. Then, skeptical entailment from $\langle \mathcal{B}^+, \mathcal{B}^-, C \rangle$ with respect to $\mathcal{C}$ is defined by

$$\mathsf{skeptical}_{\mathcal{C}}(\langle \mathcal{B}^+, \mathcal{B}^-, C \rangle) =$$
$$\{B \in \mathcal{L}_{PL} \mid \textit{for each } \mathrel{|\!\sim}' \textit{ possible under } \langle \mathcal{B}^+, \mathcal{B}^-, C \rangle \textit{ and } \mathcal{C}\textit{: } \bigwedge(C) \mathrel{|\!\sim}' B\}.$$

In this subsection, we elaborated $\mathcal{A}$'s reasoning as an attacker as postulated in Chapter 4 in the situation that $\mathcal{D}$ has not hidden any assertion from $\mathcal{A}$. But, in general, $\mathcal{D}$ might need to hide parts of its assertions like in Example 8.1 where the assertion $d12$ is confidential itself. In the next subsection, we will elaborate $\mathcal{A}$'s skeptical entailment in the situation of *hidden assertions*. A key result will be that $\mathcal{A}$ may skeptically reason in the situation of hidden assertions as if not any assertion was hidden at all (that is as if as was C).

8.1.2 Skeptical Entailment under Hidden Assertions

Usually, agent $\mathcal{D}$ may have acquired parts of its current assertions from other sources than agent $\mathcal{A}$, for example, other agents or sensors, before interacting with $\mathcal{A}$. Moreover, $\mathcal{D}$ may not want to reveal all of its current assertions in as to agent $\mathcal{A}$ to conceal some confidential beliefs (cf. Section 6.2.1: $\mathcal{D}$'s belief contains all propositional consequences of the current assertions). For example, in Example 8.1, the current assertion $d12$ of the deployment of part #1 from supplier #2 is part of the manufacturer's confidentiality policy. Thus, in the remainder of this chapter, we drop the assumption that as is visible to $\mathcal{A}$, and instead consider that $\mathcal{D}$ intentionally leaves $\mathcal{A}$ uncertain about some of its initial assertions during their interaction. This means that agent $\mathcal{A}$ considers possible that other sources contributed further hidden assertions $\mathcal{E} \subset_{fin} \mathcal{L}_{PL}$ to the visible part $\mathcal{C}$ of $\mathcal{D}$'s current assertions before the interaction:

Definition 8.3 (Possible Extension)**.**

Let $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ be approximations, $\mathfrak{C}$ a class of consequence relations and $\mathcal{E} \subset_{fin} \mathcal{L}_{PL}$ be a finite set of assertions.
Then, $\mathcal{E}$ is a possible extension (of $\mathcal{C}$) under *$\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ and class $\mathfrak{C}$ iff there exists a consequence relation $\mathrel{|\!\sim}'$ possible under $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ and class $\mathfrak{C}$ such that $\bigwedge(\mathcal{C} \cup \mathcal{E}) \mathrel{|\!\not\sim}' \bot$.*

In this situation, under its initial approximations $\langle \mathcal{B}_0^+, \mathcal{B}_0^-, \mathcal{C}_0 \rangle$ agent $\mathcal{A}$ may choose some possible extension $\mathcal{E}$, assume that actually $as = \mathcal{C}_0 \cup \mathcal{E}$ initially and, under this assumption, construct its approximations during an uncontrolled interaction with $\mathcal{D}$ as if that as was visible.

Example 8.3 (Possible Extension)**.**

Consider the initial approximations $\langle \emptyset, \emptyset, \{s12\} \rangle$ in the context of the first scenario of Ex. 8.1. In that scenario, agent $\mathcal{A}$ may simply assume the extension $\mathcal{E} = \{s22\}$. Then, with answer undef to `que`$(s21)$ *agent $\mathcal{A}$ sets $\mathcal{B}^- = \{(s12 \wedge s22, s21), (s12 \wedge s22, \neg s21)\}$.*

Next, we change perspective and consider $\mathcal{D}$'s simulation of $\mathcal{A}$'s approximations and of $\mathcal{A}$'s skeptical entailment. The end of that simulation is $\mathcal{D}$'s simulation of $\mathcal{A}$'s postulated reasoning $\mathcal{K}_{\mathcal{A}}$ of Chapter 4 for enforcing its confidentiality aims. Following that chapter, agent $\mathcal{D}$ is well aware of *all* of $\mathcal{A}$'s information about $\mathcal{D}$'s state on which $\mathcal{A}$ might base its reasoning $\mathcal{K}_{\mathcal{A}}$ about $\mathcal{D}$. Especially, $\mathcal{D}$ is aware of $\mathcal{A}$'s initial approximations $\langle \mathcal{B}_0^+, \mathcal{B}_0^-, \mathcal{C}_0 \rangle$ here with $\mathcal{B}_0^+ = \mathcal{B}_0^- = \emptyset$ by Assumption 8.
As indicated by Figure 8.1, agent $\mathcal{D}$ may simulate $\mathcal{A}$'s behavior as follows, employing the procedures in Table 8.1. The simulation uses the attacker view defined as follows.

Definition 8.4 (Attacker View)**.**

The attacker view *$\mathcal{V}$ (*view *for short) is a triple $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ with $\mathcal{B}^+$, $\mathcal{B}^- \subset_{fin} \mathcal{L}_{PL} \times \mathcal{L}_{PL}$ and $\mathcal{C} \subset_{fin} \mathcal{L}_{PL}$.*

The attacker view stores one of $\mathcal{A}$'s approximations considered by $\mathcal{D}$. Before the interaction, $\mathcal{D}$ sets its initial view $\mathcal{V}_0$ to $\mathcal{A}$'s initial approximations and then extends this view to the view $\mathcal{V} = \langle \emptyset, \emptyset, \mathcal{C}_0 \cup \mathcal{E} \rangle$ with a possible extension $\mathcal{E}$ of the initially visible assertions $\mathcal{C}_0$. During the interaction, $\mathcal{D}$'s simulation subsequently refines the approximations in view $\mathcal{V}$ by means of the two procedures and the interaction request together with the respective reaction. We denote the view computed in the simulation by $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})$ referring to the initial view $\mathcal{V}_0$, to the chosen extension $\mathcal{E}$ and to the history $\mathcal{H}^{NMR}$ of the interaction.
Postulating $\mathcal{A}$ to be a skeptical reasoner, $\mathcal{D}$ considers $\mathcal{A}$ to be interested in all of $\mathcal{D}$'s beliefs that $\mathcal{A}$ can skeptically conclude independently from the possible extension that $\mathcal{A}$ might choose initially. We define $\mathcal{A}$'s skeptical entailment operator in the way that it is simulated by $\mathcal{D}$ by computing views from the history and possible extensions.

Definition 8.5 (Skeptical Entailment with Hidden Assertions).

Let $\mathcal{V}_0$ be an initial view, $\mathfrak{C}$ a class of consequence relations and $\mathcal{H}^{NMR}$ the history of an interaction. Then, skeptical entailment from $\mathcal{H}^{NMR}$ under $\mathcal{V}_0$ with respect to $\mathfrak{C}$ is defined by

$$\begin{aligned}\mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR}) := \{B \in \mathcal{L}_{PL} \mid B \in \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR}))\\ \textit{for all possible extensions } \mathcal{E} \subset_{\mathit{fin}} \mathcal{L}_{PL} \textit{ under } \mathcal{V}_0 \textit{ and } \mathfrak{C}\}.\end{aligned}$$

Skeptical entailment is used in $\mathcal{D}$'s simulation of $\mathcal{A}$'s postulated reasoning of Chapter 4 as shown in Figure 8.1. Section 8.3 and Section 8.4 yield a procedure to decide skeptical entailment by deduction. This procedure is a major step in the construction of a confidentiality preserving agent $\mathcal{D}$.

8.2 Implementation of Defender $\mathcal{D}$ as Seen by Attacker $\mathcal{A}$

As a part of the security engineering task, we make precise which details of $\mathcal{D}$'s implementation $\mathcal{A}$ is aware of. Complementary, the model $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of Chapter 4 describes $\mathcal{A}$'s awareness about $\mathcal{D}$'s abstract functionality. In that model, the set $\mathcal{St}^{NMR}$ describes possible implementations of $\mathcal{D}$'s state from $\mathcal{A}$'s perspective.

Definition 8.6 (State of Defender $\mathcal{D}$ with Nonmonotonic Reasoning).
Cf. Definition 4.2

The set $\mathcal{St}^{NMR}$ of agent $\mathcal{D}$'s local states consists of tuples with the following components:

1. *Background knowledge:*
   agent $\mathcal{D}$'s fixed reasoning $\mid\!\sim_{\mathcal{D}}$ as an instance of the class $\mathcal{C}$ of consequence relations.

2. *Evidential knowledge:*
   agent $\mathcal{D}$'s set as of current assertions such that $\langle \mid\!\sim_{\mathcal{D}}, \mathit{as}\rangle$ forms an epistemic state of Def. 6.7.

3. *Belief operator:*
   the operator $\mathsf{Bel}_{\mathcal{D}}^{NMR}$ defined in (6.4) as the result of reasoning from as by $\mid\!\sim_{\mathcal{D}}$.

4. *Confidentiality policy:*
   the set $conf(TCP)$ of temporary confidential belief,
   whereas we set $conf(CCP) = \emptyset$.

5. *History:*
   a sequence $\mathcal{H}^{NMR}$ of pairs of request $\theta \in \mathcal{Req}^{NMR}$ and respective reaction $react \in \mathcal{Rea}$.

6. *Attacker view:*
   a triple $\langle \mathcal{B}^{+}, \mathcal{B}^{-}, \mathcal{C}\rangle$ of Def. 8.4 with $\langle \mathcal{B}^{+}, \mathcal{B}^{-}, \mathcal{C}\rangle = \mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})$ being one of the attacker $\mathcal{A}$'s approximations considered by $\mathcal{D}$.

The set $\mathcal{St}^{NMR}$ is a priori knowledge of the attacker $\mathcal{A}$ because the set is part of the interface of the control function which is open to $\mathcal{A}$ by Assumption 2. Knowing the set $\mathcal{St}^{NMR}$, agent $\mathcal{A}$ is aware that $\mathcal{D}$'s fixed reasoning $\mid\!\sim_{\mathcal{D}}$ is an instance of the class $\mathcal{C}$ of consequence relations. This awareness is a worst case for the defender $\mathcal{D}$ since it enables $\mathcal{A}$ to more easily follow $\mathcal{D}$'s process of forming belief with $\mathcal{D}$'s consequence relation. The worst case is reasonable

because, for example, in the application there might be a common agreement between (the designers of) $\mathcal{D}$ and $\mathcal{A}$ that the agents' fixed reasoning is implemented by ordinal conditional functions (or another plausibility structure) so that the class $\mathcal{C}$ is known to both agents.
Knowing the set $\mathcal{St}^{NMR}$, agent $\mathcal{A}$ is further aware that the components $\mid\!\sim_{\mathcal{D}}$ and $\boldsymbol{as}$ form an epistemic state of Definition 6.7. From another perspective, the set $\mathcal{St}^{NMR}$ may be further seen as defining invariants of the control function which $\mathcal{A}$ is a aware of.
Following Assumption 2, we may define $\mathcal{A}$'s awareness of $\mathcal{D}$'s initial state in the implementation, using the set $\mathcal{St}^{NMR}_{Init} \subseteq \mathcal{St}^{NMR}$. We define this set by a predicate `pre` over $\mathcal{St}^{NMR}$

$$\begin{aligned}
&\texttt{pre}(\mid\!\sim_{\mathcal{D}}, \boldsymbol{as}, \mathit{conf}(\mathit{TCP}), \mathcal{H}^{NMR}, \langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C}\rangle) \text{ is true iff}\\
&\qquad\qquad \mathcal{H}^{NMR} = \langle\rangle\\
&\text{and}\qquad \mathcal{B}^+ = \mathcal{B}^- = \emptyset\\
&\text{and}\qquad B \notin \mathsf{skeptical}_{\mathcal{C}}(\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C}\rangle) \text{ for all } B \in \mathit{conf}(\mathit{TCP})\\
&\text{and}\qquad \boldsymbol{as} \vdash_{pl} F \text{ for all } F \in \mathcal{C} \text{ hold.}
\end{aligned} \tag{8.1}$$

The predicate reflects Assumption 8 that agent $\mathcal{A}$ is initially ignorant about $\mathcal{D}$'s fixed reasoning.

8.3 Representing $\mathcal{A}$'s Awareness about $\mathcal{D}$'s Fixed Reasoning

Skeptical entailment (Definition 8.5 based on Definition 8.2) involves reasoning about instances of the class $\mathcal{C}$ of consequence relations. In this section, we require that the class $\mathcal{C}$ may be characterized by an appropriate axiomatization so that reasoning about its instances reduces to deduction with respect to its axiomatization. To this end, we introduce dedicated "allowance properties" of axiomatizations which we need in order to establish our results on deciding skeptical entailment. We illustrate that these properties are satisfied by axiomatizations from the literature on nonmonotonic reasoning for some common classes of consequence relations.
These "allowance properties" are syntactical restrictions motivated by the

following general idea. In the process of skeptical entailment of Definition 8.5, agent $\mathcal{A}$ might consider infinitely many different sets of assertions that $\mathcal{D}$ might have hidden. In order to simplify the simulation of that process, for two sets $\mathcal{E}_1$ and $\mathcal{E}_2$ of hidden assertions we want to transform the computed views $\mathcal{V}(., \mathcal{E}_1, .)$ and $\mathcal{V}(., \mathcal{E}_2, .)$ into one another by simple syntactic manipulations.

8.3.1 Allowance Properties of Classes of Consequence Relations

An appropriate language to describe properties of consequence relations is the language of conditional logic, cf. for example [50]. The relation $A \mathrel{|\!\sim} B$ on the semantic level is represented by the *conditional expression* $A \rightarrow B$ on the syntactic level. Here, $\rightarrow$ is the *conditional operator* whereas $\Rightarrow$ is material implication in propositional logic. In this thesis, we only use the restricted language of Def. 8.7 below which suffices to describe properties of a *fixed* consequence relation, cf. [50]. In the full conditional language, conditional expressions may be nested.

Definition 8.7 (Conditional Language $\mathcal{L}^*_{CL}$).

The conditional language $\mathcal{L}^*_{CL}$ is inductively defined as follows:

Basic Formulas	$A \rightarrow B \in \mathcal{L}^*_{CL}$ *for all* $A, B \in \mathcal{L}_{PL}$.
Compound Formulas	$\mathcal{L}^*_{CL}$ *is closed under the propositional connectives* $\wedge, \vee, \neg$ *and* $\Rightarrow$.

A satisfaction relation, denoted by the model-of operator $\models_{CL}$, between a consequence relation $\mathrel{|\!\sim}$ and a basic formula $A \rightarrow B \in \mathcal{L}^*_{CL}$ is defined as follows:

$$\mathrel{|\!\sim}\models_{CL} A \rightarrow B \text{ iff } (A, B) \in \mathrel{|\!\sim} . \tag{8.2}$$

The operator is inductively extended to the propositional connectives as usual. For example, $\mathrel{|\!\sim}\models_{CL} \neg(A \rightarrow B)$ means that $(A, B) \notin \mathrel{|\!\sim}$ holds or, using a different notation, that $A \mathrel{|\!\not\sim} B$ holds. The conjunction $(A \rightarrow B) \wedge (C \rightarrow D)$ is true for $\mathrel{|\!\sim}$ iff $(A, B) \in \mathrel{|\!\sim}$ and $(C, D) \in \mathrel{|\!\sim}$ hold.
The objects $\mathrel{|\!\sim} \subseteq \mathcal{L}_{PL} \times \mathcal{L}_{PL}$ defining the semantics are of very simple structure because, for the time being, we consider just consequence relations $\mathrel{|\!\sim}_{\mathcal{D}}$ but not the structures implementing them. Complementary, in

Section 8.4.2, following our running example, we will discuss the case that $\mathcal{D}$'s reasoning is implemented by an ordinal conditional function.
To describe classes of consequence relations, we consider schema formulas defining the interactions between premises and conclusions in consequence relations. In an axiomatization of a class of consequence relations, we allow only certain schema formulas as axiom schemes where the interaction between premises and conclusions is limited. Roughly, some marked schema variables may not change their sign when used both in premises and conclusions of conditional expressions. The limitation is sufficient for Theorem 8.1 of Section 8.4 and allows many of the well-known axiomatizations from the literature, for example in [74, 77, 14, 50, 81].
Let $\mathcal{X}_{PL}$ be a set of schema variables as placeholders for propositional formulas. The languages $\mathcal{L}_{PL}(\mathcal{X}_{PL})$ and $\mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL}))$ define schema formulas, replacing the alphabet $\mathcal{At}$ underlying $\mathcal{L}_{PL}$ by $\mathcal{X}_{PL}$ and replacing $\mathcal{L}_{PL}$ by $\mathcal{L}_{PL}(\mathcal{X}_{PL})$ in Definition 8.7, respectively.

Definition 8.8 (Allowed Schema Formulas in $\mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL}))$).

A schema formula $\mathbf{\Lambda} \in \mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL}))$ is allowed if there exists a marking of schema variables represented by disjoint sets $\mathcal{X}_{PL}^{Pos} \subseteq \mathcal{X}_{PL}$ (marked as positive) and $\mathcal{X}_{PL}^{Neg} \subseteq \mathcal{X}_{PL}$ (marked as negative) such that the following requirements are met. For every basic formula $\Phi \to \Gamma$ of $\mathbf{\Lambda}$ with $\Phi, \Gamma \in \mathcal{L}_{PL}(\mathcal{X}_{PL})$ we assume that Φ and Γ are in conjunctive normal form (CNF)[5] and, moreover, that $\mathbf{\Lambda}$ is in CNF[6].[7] For every basic formula $\Phi \to \Gamma$ of $\mathbf{\Lambda}$, the premise Φ and the conclusion Γ comply with the marking $\mathcal{X}_{PL}^{Pos}, \mathcal{X}_{PL}^{Neg}$ as follows:

1. *Φ: there exists at least one conjunct $(\neg)\lambda_1 \vee \ldots \vee (\neg)\lambda_l$, $\lambda_i \in \mathcal{X}_{PL}$, such that*
   for all schema variables λ_i with $i = 1, \ldots, l$,
   either λ_i occurs positive and $\lambda_i \in \mathcal{X}_{PL}^{Pos}$ (λ_i is marked as positive);

[5] That is, a conjunction of disjunctions where each disjunction uses literals over distinct atoms. Here, the atoms are schema variables of $\mathcal{X}_{PL}$ as placeholders for propositional formulas.

[6] Here, in the CNF the basic formulas of $\mathcal{L}^*_{CL}$ are treated as atoms.

[7] The restriction to CNF actually does not impose any restriction on axiom schemes, because the rule schemes (LLE) and (RW) in Def. 8.9 permit equivalence transformations within premises and conclusions of conditional expressions.

or λ_i occurs negative and $\lambda_i \in \mathcal{X}_{PL}^{Neg}$ (λ_i is marked as negative).

2. *Γ: for every schema variable $\lambda \in \mathcal{X}_{PL}$ in Γ:*
   If $\lambda \in \mathcal{X}_{PL}^{Pos}$, then each occurrence of λ is under an even number of negations in $\mathbf{\Lambda}$.[8]
   Else if $\lambda \in \mathcal{X}_{PL}^{Neg}$, then every occurrence of λ is under an odd number of negations in $\mathbf{\Lambda}$.
   Otherwise $\lambda \in \mathcal{X}_{PL} \setminus (\mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg})$ and the use of λ in Γ is unrestricted.

Here, $(\neg)$ denotes that the sign of the following expression is arbitrary.

Example 8.4 (Allowed Schema Formulas).

Rational monotony (RM) [77] $(\phi \rightarrow \psi) \wedge \neg(\phi \rightarrow \neg\xi) \Rightarrow (\phi \wedge \xi \rightarrow \psi)$ with $\mathcal{X}_{PL} = \{\phi, \psi, \xi\}$ is an allowed schema formula. Def. 8.8 requires that for each conditional expression we mark all schema variables in at least one conjunct of its premise, here we do so by $\mathcal{X}_{PL}^{Pos} = \{\phi\}$ and $\mathcal{X}_{PL}^{Neg} = \emptyset$. Because the marked schema variable ϕ does not appear in any conclusion, (RM) complies with our marking. A negative example is rational contraposition (RC) [14] $(\alpha \rightarrow \beta) \wedge \neg(\neg\beta \rightarrow \alpha) \Rightarrow (\neg\beta \rightarrow \neg\alpha)$ with CNF $\neg(\alpha \rightarrow \beta) \vee (\neg\beta \rightarrow \alpha) \vee (\neg\beta \rightarrow \neg\alpha)$. Def. 8.8 forces the marking $\mathcal{X}_{PL}^{Pos} = \{\alpha\}$ and $\mathcal{X}_{PL}^{Neg} = \{\beta\}$ but, as for the third disjunct, α occurs under an odd number of negations in (RC).

Finally, we illustrate how agent $\mathcal{A}$ may reason about possible consequence relations in the situation of hidden assertions as being enabled by the marking of an allowed schema formula of Definition 8.8.

Example 8.5 (Reasoning with Hidden Assertions).

Consider that agent $\mathcal{A}$ is aware that $\mathcal{D}$'s fixed reasoning $\mid\!\sim_{\mathcal{D}}^{\kappa_1}$ of Sect. 6.2.2 is implemented by an OCF. Thus, $\mathcal{A}$ knows that $\mid\!\sim_{\mathcal{D}}^{\kappa_1}$ shows the property of rational monotony (RM) of Ex. 8.4, cf. [50]. With that property and the visible initial assertion $s12$ agent $\mathcal{A}$ may reason that from $s12 \rightarrow (s21 \Rightarrow r)$ and $\neg(s12 \rightarrow \neg s21)$ follows $s12 \wedge s21 \rightarrow r$ (where $s12$ substitutes the marked conjunct ϕ in the schema formula of (RM) of Ex. 8.4). With this reasoning, $\mathcal{A}$

[8] Note that both Γ and $\mathbf{\Lambda}$ are in CNF so that negations are not implicit in material implications.

assumes that $\mathcal{D}$ has not hidden any assertion. Differently, agent $\mathcal{A}$ may assume that $\mathcal{D}$ has hidden the initial assertion $s22$ like in Ex. 8.3. Under this assumption, $\mathcal{A}$ may want to do the same reasoning, but from the premise $s12 \wedge s22$. To this end, $\mathcal{A}$ replaces the marked conjunct ϕ with $s12 \wedge s22$. Since the schema formula (RM) is allowed, the presumable hidden assertion $s22$ does not interact with the conclusions in the conditional expressions in a way that precludes $\mathcal{A}$ from doing the same reasoning as with visible assertions.

8.3.2 Axiomatizing Classes of Consequence Relations

As Figure 8.1 shows, agent $\mathcal{D}$'s simulation of agent $\mathcal{A}$'s postulated reasoning uses an axiomatization of the class $\mathcal{C}$ of consequence relations. This axiomatization should be a set of axiom and rule schemes that follow some restrictions needed in the proof of Theorem 8.1 on skeptical entailment. To make clear the different aspects that $\mathcal{A}$ considers about $\mathcal{D}$'s fixed reasoning, that is, propositional consequence, unquestionable belief, nonmonotonic consequence and classes of consequence relations, we present a set of axiom and rule schemes in four corresponding layers. In the first three layers, schema variables are placeholders for propositional formulas whereas in the fourth layer they are placeholders for conditional formulas.
Let $\mathcal{X}_{CL}$ be a set of schema variables as placeholders for conditional formulas (disjoint with $\mathcal{X}_{PL}$). The language $\mathcal{L}_{PL}(\mathcal{X}_{CL})$ defines schema formulas, replacing the alphabet $\mathcal{At}$ by $\mathcal{X}_{CL}$.

Definition 8.9 (Allowed Set **Ax** of Axiom and Rule Schemes)**.**

An allowed set **Ax** *of axiom and rule schemes is finite and consists of the following schemes.*
First layer, with $\alpha, \beta, \lambda \in \mathcal{X}_{PL}$:

(REF) (reflexivity)	$\lambda \to \lambda$
(LLE) (left logical equivalence)	If $\vdash_{pl} \alpha \Leftrightarrow \beta$ then $(\beta \to \lambda) \Leftrightarrow (\alpha \to \lambda)$
(RW) (right weakening)	If $\vdash_{pl} \beta \Rightarrow \lambda$ then $(\alpha \to \beta) \Rightarrow (\alpha \to \lambda)$

Second layer, with $\alpha, \beta \in \mathcal{X}_{PL}$ and $\mathbf{\Lambda} \in \mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL}))$:

(IMP) (implausibility)	$(\alpha \rightarrow \perp) \Rightarrow (\alpha \wedge \beta \rightarrow \perp)$
(CON) (consistency)	$\neg(\top \rightarrow \perp)$

Axiom schemes $\mathbf{\Lambda}$, where $\mathbf{\Lambda}$ is allowed (Def. 8.8),
for properties of unquestionable belief defined by the consequence relation.
Third layer, with $\mathbf{\Lambda} \in \mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL}))$:
axiom schemes $\mathbf{\Lambda}$, where $\mathbf{\Lambda}$ is allowed (Def. 8.8), for other properties
such as nonmonotonic properties.
Fourth layer, with $\Lambda, \Lambda_i \in \mathcal{L}_{PL}(\mathcal{X}_{CL})$:
axiom Λ and rule schemes of the form If $\Lambda_1, \ldots, \Lambda_n$ then Λ_{n+1}
of any sound and complete propositional calculus.

In the first layer, axiom and rule schemes relate consequence relations and propositional entailment $\vdash_{pl}$, cf. [77, 50].

(a) The rule scheme right weakening (RW) and the axiom scheme reflexivity (REF) ensure that a consequence relation extends propositional consequence $\vdash_{pl}$ from the assertions as. They guarantee property (6.6) of belief in Section 6.2.1.

(b) The rule scheme left logical equivalence (LLE) ensures that consequences do not depend on the syntactical representation of premises. It guarantees property (6.7) of belief.

In the second layer, axiom schemes[9] define properties of unquestionable belief:

(a) The axiom scheme implausibility (IMP) says that if A is contradictory under $\mid\!\sim$ then it is also contradictory in the presence of additional information B. Especially, it implies together with (LLE) that unquestionable belief is monotonic, that is, never given up. More precisely, if $\mathcal{D}$ unquestionably believes $F \in \mathcal{L}_{PL}$ in the presence of the current assertions as, formally, $\bigwedge(\mathit{as}) \wedge \neg F \mid\!\sim_{\mathcal{D}} \perp$, then it does so in the presence of additional information A by (IMP) and (LLE), formally $\bigwedge(\mathit{as} \cup \{A\}) \wedge \neg F \mid\!\sim_{\mathcal{D}} \perp$.

[9] Cf. [50] for the relation between these axioms and epistemic logic S5 of knowledge. In particular, (CON) corresponds to the axiom consistency and (IMP) follows from a scheme in $\mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL}))$ corresponding to the axiom scheme of distributivity.

(b) The axiom consistency (CON) ensures in the presence of (IMP) and (RW) that the relation $\mathcal{L}_{PL} \times \mathcal{L}_{PL}$ saying that "every formula may be inferred from any assertion" is not in the axiomatized class of consequence relations because it cannot be used in an epistemic state (Definition 6.7).

(c) Further axiom schemes may define further properties of unquestionable belief. Their basic formulas should have the form of $\Phi \to \bot$ with $\Phi \in \mathcal{L}_{PL}(\mathcal{X}_{PL})$ so that these axiom schemes define what assertions agent $\mathcal{D}$ considers contradictory. These schemes are restricted to allowed schema formulas (Definition 8.8).

In the third layer, axiom schemes define the interaction between premises and conclusions in a consequence relation. They are restricted to allowed schema formulas (Definition 8.8). These axiom schemes reflect the characteristics of $\mathcal{D}$'s fixed reasoning that agent $\mathcal{A}$ is aware of. In contrast to the second layer, they may define nonmonotonic properties like those of Example 8.4.
In the fourth and last layer, axioms and rules schemes of any sound and complete propositional calculus define propositional reasoning about classes of consequence relations.
In the following, we define axiomatizations of a class $\mathcal{C}$ of consequence relations in the usual way. An *axiomatization* of $\mathcal{C}$ is a set $\mathbf{Ax}$ of axiom and rules schemes that should capture the characteristics of $\mathcal{C}$ in the following sense. There may be properties $F \in \mathcal{L}^*_{CL}$ that every consequence relation of the class $\mathcal{C}$ has, formally, $\mathrel{|\!\sim} \models_{CL} F$ for all $\mathrel{|\!\sim} \in \mathcal{C}$. These properties characterize the class $\mathcal{C}$ and, thus, these properties and only these should follow from deduction with an axiomatization $\mathbf{Ax}$ of $\mathcal{C}$. More formally, first, deduction is defined as applying finitely many substitution instances of axiom and of rule schemes in the usual way:

Definition 8.10 (Deduction $\Vdash_{\mathbf{Ax}}$).

Given an allowed set $\mathbf{Ax}$ *of axiom and rule schemes, a set* $M \subseteq \mathcal{L}^*_{CL}$ *and a formula* $F \in \mathcal{L}^*_{CL}$*, we write* $M \Vdash_{\mathbf{Ax}} F$ *iff there exists a sequence* $F_1, \dots, F_n$ *of formulas* $F_i \in \mathcal{L}^*_{CL}$ *such that* $F_n = F$ *and*
for all $i = 1, \dots, n$

- *$F_i \in M$,*
- *or applying rule scheme (LLE) with a substitution $\mathsf{sub} : \mathcal{X}_{PL} \to \mathcal{L}_{PL}$ extended to $\mathsf{sub} : \mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL})) \to \mathcal{L}^*_{CL}$ as usual such that $F_i = \mathsf{sub}((\alpha \to \lambda) \Leftrightarrow (\beta \to \lambda))$ and $\vdash_{pl} \mathsf{sub}(\alpha) \Leftrightarrow \mathsf{sub}(\beta)$,*
- *or applying rule scheme (RW) with a substitution $\mathsf{sub} : \mathcal{X}_{PL} \to \mathcal{L}_{PL}$ extended to $\mathsf{sub} : \mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL})) \to \mathcal{L}^*_{CL}$ as usual such that $F_i = \mathsf{sub}((\alpha \to \beta) \Rightarrow (\alpha \to \lambda))$ and $\vdash_{pl} \mathsf{sub}(\beta) \Rightarrow \mathsf{sub}(\lambda)$,*
- *or there exists an axiom scheme $\mathbf{\Lambda}$ where $\mathbf{\Lambda} \in \mathbf{Ax} \cap \mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL}))$ is allowed (Def. 8.8) and a substitution $\mathsf{sub} : \mathcal{X}_{PL} \to \mathcal{L}_{PL}$ extended to $\mathsf{sub} : \mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL})) \to \mathcal{L}^*_{CL}$ as usual such that $F_i = \mathsf{sub}(\mathbf{\Lambda})$,*
- *or there exists an axiom scheme $\Lambda \in \mathbf{Ax} \cap \mathcal{L}_{PL}(\mathcal{X}_{CL})$ and a substitution $\mathsf{sub} : \mathcal{X}_{CL} \to \mathcal{L}^*_{CL}$ extended to $\mathsf{sub} : \mathcal{L}_{PL}(\mathcal{X}_{CL}) \to \mathcal{L}^*_{CL}$ as usual such that $F_i = \mathsf{sub}(\Lambda)$,*
- *or there exists a rule scheme* If $\Lambda_1, \ldots, \Lambda_m$ then $\Lambda_{m+1} \in \mathbf{Ax}$ *with $\Lambda_j \in \mathcal{L}_{PL}(\mathcal{X}_{CL})$ and a substitution $\mathsf{sub} : \mathcal{X}_{CL} \to \mathcal{L}^*_{CL}$ extended to $\mathsf{sub} : \mathcal{L}_{PL}(\mathcal{X}_{CL}) \to \mathcal{L}^*_{CL}$ as usual such that $F_i = \mathsf{sub}(\Lambda_{m+1})$ and for all $j = 1, \ldots, m$ there exists $k < i$ with $F_k = \mathsf{sub}(\Lambda_j)$.*

The sequence $F_1, \ldots, F_n$ is called a valid proof of F from M with respect to $\mathbf{Ax}$.

Next, a class $\mathcal{C}$ of consequence relations is identified with the common properties of its instances, that is, the set of all conditional formulas that are $\mathcal{C}$-entailed from $\emptyset$. Given a class $\mathcal{C}$ of consequence relations, a set $M \subseteq \mathcal{L}^*_{CL}$ and a formula $F \in \mathcal{L}^*_{CL}$, we write $M \vdash_{\mathcal{C}} F$, reading M $\mathcal{C}$-entails F, iff for all $\mathrel{|\!\sim} \in \mathcal{C}$ it holds that, if $\mathrel{|\!\sim} \models_{CL} P$ holds for all $P \in M$, then $\mathrel{|\!\sim} \models_{CL} F$ holds. Finally, a class $\mathcal{C}$ of consequence relations is characterized via the common properties of its instances by a corresponding set of axiom and rule schemes, called an axiomatization of $\mathcal{C}$.

Definition 8.11 (Soundness and Completeness, Axiomatizations).
Cf. Chapter 7.2.3 of [60]

Let $\mathcal{C}$ be a class of of consequence relations and **Ax** *an allowed set of axiom and rule schemes. Then,* **Ax** *is sound for $\mathcal{C}$ iff for all $F \in \mathcal{L}^*_{CL}$ from $\emptyset \Vdash_{\mathbf{Ax}} F$ it follows that $\emptyset \vdash_{\mathcal{C}} F$ holds. Further,* **Ax** *is complete for $\mathcal{C}$ iff for all $F \in \mathcal{L}^*_{CL}$ from $\emptyset \vdash_{\mathcal{C}} F$ it follows that $\emptyset \Vdash_{\mathbf{Ax}} F$ holds. An axiomatization of $\mathcal{C}$ is a sound and complete set of axiom and rule schemes for $\mathcal{C}$.*

In this article, we consider only allowed sets of axiom and rules schemes for axiomatizations in the design of agent $\mathcal{D}$ (Figure 8.1):

Assumption 9 (Characterization of Class $\mathcal{C}$).

The class $\mathcal{C}$ can be axiomatized by an allowed set **Ax** *of axiom and rule schemes.*

8.4 Computing $\mathcal{A}$'s Postulated Skeptical Reasoning For Simulation

8.4.1 Reduction to Deduction with Allowed Axiomatizations

In this subsection, we give means to decide skeptical entailment of confidential belief as needed to simulate the postulated behavior of the attacking agent $\mathcal{A}$, cf. Figure 8.1. In brief, the procedure for deciding $B \in \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$ works as follows. Initially, provide an allowed axiomatization **Ax** of class $\mathcal{C}$. During the interaction compute $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})$ by the procedures in Table 8.1, as described in Section 8.1.2. Then, decide $B \in \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}))$ with a solver for deduction with **Ax**. First, we will outline the computation of $\mathsf{skeptical}_{\mathcal{C}}(\mathcal{V})$ with a view $\mathcal{V}$ by deduction and straightforwardly prove its correctness. Then, as the main contribution of this section, we provide the correctness result of the procedure for deciding $B \in \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$ sketched in the beginning of this section. The proof of this result is much more involved and bases on the allowance properties for the axiomatization **Ax** of class $\mathcal{C}$ from Section 8.3.

To start with, we characterize the operator $\mathsf{skeptical}_{\mathcal{C}}(\mathcal{V})$ by deduction with respect to the axiomatization $\mathbf{Ax}$. Since this operator formalizes the reasoning about possible consequence relations under view $\mathcal{V}$ and the class $\mathcal{C}$, we translate the view $\mathcal{V} = \langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C}\rangle$ to the following set $CL(\mathcal{V}) \subset_{fin} \mathcal{L}^*_{CL}$ of conditional expressions describing the related properties of a possible consequence relation (Definition 8.1):

$$\begin{aligned} &CL(\mathcal{V}) := \\ &\quad \{A \to B \mid (A,B) \in \mathcal{B}^+\} \cup \{\neg(A \to B) \mid (A,B) \in \mathcal{B}^-\} \quad &(\mathcal{B}\text{ Compatible})\\ &\quad\quad \cup \{\neg(\textstyle\bigwedge(\mathcal{C}) \to \bot)\} \quad &(\mathcal{C}\text{ Consistent}) \end{aligned} \tag{8.3}$$

Now, the following proposition gives a means to compute the operator $\mathsf{skeptical}_{\mathcal{C}}(\mathcal{V})$ by deduction with respect to the axiomatization $\mathbf{Ax}$ of class $\mathcal{C}$.

Proposition 8.1 (Characterization of Possible Consequence Relations)**.**

Let $\mathcal{V}$ be a view and $\mathcal{C}$ a class of consequence relations axiomatized by an allowed set $\mathbf{Ax}$ *of axiom and rules schemes (Asmp. 9). Then, there exists a possible consequence relation under $\mathcal{V}$ and $\mathcal{C}$ iff $CL(\mathcal{V}) \nVdash_{\mathbf{Ax}} (\top \to \bot)$.*

The only-if part needs the presence of the axiom (CON) $\neg(\top \to \bot)$ in $\mathbf{Ax}$ given by Definition 8.9.

Corollary 8.1 (Computation of $\mathsf{skeptical}_{\mathcal{C}}(\mathcal{V})$)**.**

Let $\mathcal{V} = \langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C}\rangle$ be a view and $\mathcal{C}$ a class of consequence relations axiomatized by $\mathbf{Ax}$ *as in Asmp. 9. Then, for every $B \in \mathcal{L}_{PL}$ it holds $B \in \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V})$ iff $CL(\mathcal{V}_B) \Vdash_{\mathbf{Ax}} (\top \to \bot)$ where $\mathcal{V}_B := \langle \mathcal{B}^+, \mathcal{B}^- \cup \{(\bigwedge(\mathcal{C}), B)\}, \mathcal{C}\rangle$.*

The corollary follows right from Proposition 8.1 and Definition 8.2.
Next, we will prove in Theorem 8.1 that for deciding $B \in \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$ of Def. 8.5 we may assume that $\mathcal{D}$ has not hidden any assertion, that is, $\mathcal{E} = \emptyset$. Or in other words, deciding $B \in \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$ is equivalent to deciding

$B \in \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}))$, the skeptical entailment of B with visible assertions of Definition 8.2.

For a better understanding of that claim, we recall that skeptical entailment $B \in \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$ may be described as the failure of the following process of finding an alternative situation in which $\mathcal{D}$ does not believe $B \in \mathcal{L}_{PL}$ (Definition 8.5 based on Definition 8.2).

In the first step of that process, assume initially that $\mathcal{D}$ has hidden the assertions $\mathcal{E}$ (Definition 8.3). Then, throughout the interaction, approximate $\mathcal{D}$'s fixed reasoning $\mid\!\sim_{\mathcal{D}}$ by the observations recorded in history $\mathcal{H}^{NMR}$ in the components $\mathcal{B}^+(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})$ and $\mathcal{B}^-(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})$ of the view $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})$ where $\mathcal{V}_0$ denotes the initial view.

In the second step, try to find a possible consequence relation $\mid\!\sim'$ under the view $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})$ computed in the first step and the class $\mathcal{C}$ known a priori such that B is not a defeasible conclusion from the tracked visible and the assumed hidden assertions given $\mid\!\sim'$. Or equivalently, try to verify that $B \notin \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR}))$ (Definition 8.2).

In this context, Theorem 8.1 says that the first step of the process may be skipped by setting $\mathcal{E} = \emptyset$.

Theorem 8.1 (Skeptical Entailment under Hidden Assertions)**.**

Let $\mathcal{V}_0 = \langle \emptyset, \emptyset, \mathcal{C}_0 \rangle$ be the initial view as in Asmp. 8, $\mathcal{C}$ a class of consequence relations axiomatized by **Ax** *as in Asmp. 9 and $\mathcal{H}^{NMR}$ a history. Then,*

$$\mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) = \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}_0, \mathcal{H}^{NMR}).$$

Sketch of proof. We briefly outline the proof of the appendix. By Definition 8.5 and Corollary 8.1, we may decide $B \in \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$ by verifying that for all possible extensions $\mathcal{E}$ under $\mathcal{V}_0$ and $\mathcal{C}$, yielding a respective view $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})$, we find a valid proof of $CL(\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})_B) \Vdash_{\mathbf{Ax}} (\top \to \bot)$ where $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})_B$ is defined as in Corollary 8.1.

Now, the key to the proof of Theorem 8.1 is that we may translate a valid proof

$$F_1, \dots, F_n \qquad \text{of} \qquad CL(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})_B) \Vdash_{\mathbf{Ax}} (\top \to \bot)$$

to a valid proof

$$G_1, \dots, G_m,\ m \geq n \qquad \text{of} \qquad CL(\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})_B) \Vdash_{\mathbf{Ax}} (\top \to \bot)$$

for any set $\mathcal{E} \subset_{fin} \mathcal{L}_{PL}$.
To justify that our translation is indeed a valid proof, we essentially need that $\mathbf{Ax}$ is an allowed set of axiom and rule schemes (Definition 8.9). The justification roughly proceeds as follows. Above all, we translate the valid proof $F_1, \ldots, F_n$ by a *transformation function* t and show by induction that $\mathsf{t}(F_n) = G_m$ indeed may be deduced from $CL(\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})_B)$. The transformation function adds the hidden assertions $\mathcal{E}$ to premises in conditional expressions. In the special case that F_n is a substitution instance of an allowed formula $\mathbf{\Lambda} \in \mathbf{Ax}$ (Def. 8.8) we find another appropriate substitution sub' such that from $\mathsf{sub}'(\mathbf{\Lambda})$ and possibly further steps we may deduce $\mathsf{t}(F_n)$. The substitution sub' adds the hidden assertions when substituting the marked conjunct of Definition 8.8, cf. Example 8.5. Then, in further steps, with the rules (LLE) and (RW) of Definition 8.9 we may bring the substitution instance $\mathsf{sub}'(\mathbf{\Lambda})$ into the form of the transformed formula $\mathsf{t}(F_n)$. In those steps, the application of (RW) bases on the compliance of conclusions in conditional expressions with the marking of Def. 8.8. Finally, with the translation we obtain $CL(\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})_B) \Vdash_{\mathbf{Ax}} (\bigwedge(\mathcal{E}) \rightarrow \bot)$, that is, the hidden assertions $\mathcal{E}$ are considered contradictory. But with the axiom scheme (IMP) of Definition 8.9 the latter implies that the assertions $\mathcal{C}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR}) \supseteq \mathcal{E}$ are also considered contradictory which leads to a propositional contradiction with the hypothesis $CL(\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})_B)$ by (8.3) and, hence, to the deduction of the formula $(\top \rightarrow \bot)$.

□

Note that Definition 8.9 of allowed sets of axiom and rules schemes and the underlying Definition 8.8 are tailored to the proof of this theorem.
Theorem 8.1 immediately gives the following result for computing the operator $\mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$.

Corollary 8.2 (Computation of $\mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$)**.**

Let $\mathfrak{C}$ be axiomatized by an allowed set $\mathbf{Ax}$ of axioms and rule schemes (Asmp. 9). Then, for all $B \in \mathcal{L}_{PL}$ it holds that $B \in \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$ iff $CL(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \Vdash_{\mathbf{Ax}} \bigwedge(\mathcal{C}) \rightarrow B$.

Example 8.6 (Skeptical Entailment from a Successful Revision).

Consider agent $\mathcal{D}$'s components κ_1 from Sect. 6.2.2 and its assertions $\mathit{as} = \{d12, s12\}$ with agent $\mathcal{A}$'s initial approximations $\langle\emptyset, \emptyset, \{s12\}\rangle$. We follow $\mathcal{A}$'s skeptical reasoning about whether $\mathcal{D}$ believes r in the first scenario of Ex. 8.1. In that example, to find out about a positive result r of the manufacturer $\mathcal{D}$'s feasibility study, agent $\mathcal{A}$ interacts with $\mathcal{D}$ as follows

$$\mathcal{H}_1^{NMR} = \langle(\mathtt{que}(s21), \mathit{undef}),\ (\mathtt{que}(s21 \Rightarrow r), \mathit{true}),\ (\mathtt{rev}(s21), \mathit{success})\rangle.$$

Assuming $\mathcal{E} = \emptyset$ (there are no hidden assertions), $\mathcal{A}$'s final approximations from $\mathcal{H}_1^{NMR}$ are tracked in the view $\mathcal{V}_3$ in which elements are labeled $\boxed{l}$ for later reference, cf. Ex. 8.2:

$$\mathcal{V}_3 = \langle\{(s12,\ s21 \Rightarrow r)\boxed{l_1}\},\ \{(s12,\ s21),\\ (s12,\ \neg s21)\boxed{l_2},\ (s12 \wedge s21, \bot)\},\ \{s12, s21\}\rangle.$$

Skeptical entailment $r \in \mathsf{skeptical}_{\mathcal{C}^\kappa}(\mathcal{V}_0, \mathcal{H}^{NMR})$ is decided by computing $CL(\mathcal{V}_3) \Vdash_{\mathbf{Ax}} s12 \wedge s21 \rightarrow r$ where we choose $\mathbf{Ax}$ as the axiomatization of the class $\mathcal{C}^\kappa$ of consequence relations definable by OCFs from [50]. The computation is illustrated by the following proof tree where each label stands for the respective formula in $CL(\mathcal{V}_3)$:

$$\dfrac{\vdash_{pl} ((s21 \Rightarrow r) \wedge s21) \Rightarrow r \qquad \textit{(AND)}^{10}\dfrac{\textit{(RM) cf. Ex. 8.4}\ \dfrac{\boxed{l_1} \qquad \boxed{l_2}}{s12 \wedge s21 \rightarrow s21 \Rightarrow r} \qquad \begin{matrix}\textit{(REF)+(RW)}\\ s12 \wedge s21 \rightarrow s21\end{matrix}}{s12 \wedge s21 \rightarrow (s21 \Rightarrow r) \wedge s21}}{s12 \wedge s21 \rightarrow r}\ \textit{(RW)}$$

In the proof tree, applying modus ponens to an axiom $A \Rightarrow B$ and a formula A is abbreviated by "(axiom name)$\frac{A}{B}$". Regarding the rule scheme (RW) (Def. 8.9), we write $\frac{\vdash_{pl} B \Rightarrow C \quad A \rightarrow B}{A \rightarrow C}$.

Example 8.7 (Skeptical Entailment from a Failed Revision).

Consider the second scenario of Ex. 8.1 where $\mathcal{A}$ reasons about a possible agreement $d12$ of $\mathcal{D}$ with supplier #2. To this end, agent $\mathcal{A}$ interacts with $\mathcal{D}$

as follows $\mathcal{H}_2^{NMR} =$

$\langle(\mathtt{rev}(\neg s11 \vee \neg s21), \mathit{success}),\ (\mathtt{que}(\neg s11 \Rightarrow d12), \mathit{true}),\ (\mathtt{rev}(\neg s21), \mathit{failure})\rangle.$

[10] The axiom scheme (AND) is $(\phi \rightarrow \alpha) \wedge (\phi \rightarrow \beta) \Rightarrow (\phi \rightarrow \alpha \wedge \beta)$, cf. [50].

Again, $\mathcal{A}$ knows that $\mathcal{D}$'s fixed reasoning is implemented with an OCF so that skeptical entailment can be decided with the axiomatization from [50]. With history $\mathcal{H}_2^{NMR}$ and assuming $\mathcal{E} = \emptyset$ the final view is

$$\begin{aligned}\mathcal{V}_3 = \langle\{&(s12 \wedge (\neg s11 \vee \neg s21),\, \neg s11 \Rightarrow d12)\,\boxed{l_1},\\ &(s12 \wedge (\neg s11 \vee \neg s21) \wedge \neg s21,\, \bot)\,\boxed{l_2}\},\, \ldots,\, \{s12, \neg s11 \vee \neg s21\}\rangle.\end{aligned}$$

Then, $\mathcal{A}$ can skeptically conclude that $\mathcal{D}$ believes $s21$ as follows where we abbreviate $s12 \wedge (\neg s11 \vee \neg s21)$ by C:

$$\dfrac{\vdash_{pl} (C \wedge \neg 21) \vee (C \wedge s21) \Leftrightarrow C \qquad (OR)^{11}\dfrac{(RW)\dfrac{\boxed{l_2} \qquad \vdash_{pl} \bot \Rightarrow s21}{C \wedge \neg s21 \to s21} \qquad \begin{array}{c}\textit{(REF)+(RW)}\\ C \wedge s21 \to s21\end{array}}{(C \wedge \neg 21) \vee (C \wedge s21) \to s21}}{C \to s21}\,\textit{(LLE)}$$

Now, $\mathcal{A}$ may do propositional reasoning from the visible assertions C and the revealed belief of $s21$ and $\neg s11 \Rightarrow d12$, applying (REF), (AND) and (RW) like in Ex. 8.6. With this way of reasoning, from $C \to s21$ agent $\mathcal{A}$ can further deduce $C \to \neg s11$. Finally, in the same way, with the last result and $\boxed{l_1}$ $\mathcal{A}$ can obtain $C \to d12$.

8.4.2 Discussion

Here, we outline the computation of skeptical entailment and its complexity with the results from Section 8.4.1 in the context of our previous work [30]. There, we assumed that the defending agent $\mathcal{D}$'s fixed reasoning is implemented by an ordinal conditional function and that the attacking agent $\mathcal{A}$ is aware of this fact. The latter assumption follows the open design assumption of Assumption 2 and says that agent $\mathcal{A}$ knows that $\mathcal{D}$'s fixed reasoning is an instance of the class

$$\mathcal{C}^{\kappa} := \{\, \mid\!\sim \in \mathcal{L}_{PL} \times \mathcal{L}_{PL} \mid \text{exists OCF } \kappa \text{ over } \Omega \text{ such that } \mid\!\sim = \mid\!\sim^{\kappa}\}.$$

To actually decide $\mathcal{A}$'s skeptical entailment under this class with the computational results from Section 8.4.1, by Assumption 9 we first must provide an allowed set of axiom and rule schemes that axiomatizes class $\mathcal{C}^{\kappa}$.

[11] The axiom scheme (OR) is $(\phi \to \alpha) \wedge (\psi \to \alpha) \Rightarrow (\phi \vee \psi \to \alpha)$, cf. [50].

System $C + \{C5, C6\}$ from [50] presents such an axiomatization.[12] The system C generalizes the well-known system P [74], which axiomatizes the class of consequence relations stemming from preference orderings, to the language $\mathcal{L}^*_{CL}$. Beside axiom and rule schemes of a propositional calculus for the outer propositional connectives of the language $\mathcal{L}^*_{CL}$, system C comprises the rule schemes (LLE) and (RW) and the axiom scheme (REF) (all in Definition 8.9), the further axiom schemes (AND) and (OR) (in Example 8.6 and Example 8.7 respectively) and finally the axiom scheme (CM) of *cautious monotonicity*[13].
By Corollary 8.2 skeptical entailment under class $\mathfrak{C}^\kappa$ can be decided by deduction with respect to $C + \{C5, C6\}$. The deduction may implemented, for example, by the solver KLMLean [55] which decides non-deduction in nondeterministic polynomial time [54] (in the size of the set (8.3) translated from the view). In comparison, deciding $\bigwedge(\mathit{as}) \mathrel{|\!\!\sim}^{\kappa}_{\mathcal{D}} B$ needs a logarithmic number of SAT solver calls if we use a compact representation of the OCF κ like in [47].[14]
Following the above, we see that Proposition 3 on skeptical entailment with respect to the single class $\mathfrak{C}^\kappa$ from our previous work [30] is covered by the more general Theorem 8.1 on skeptical entailment with respect to axiomatized classes. Likewise, we may use system $C + \{C5, C6\}$ if agent $\mathcal{D}$'s fixed reasoning was implemented by a possibility measure and agent $\mathcal{A}$ was aware of that fact [50].

8.5 Control Functions for Iterated Query and Revision Requests

This section presents the control function of $\mathcal{D}$'s censor for query requests and view update transaction requests. The control function is abstractly introduced by Definition 4.3 and its domain $\mathcal{St}^{NMR} \times \mathcal{Req}^{NMR}$ in Section 6.2. For

[12] The names $C5$ and $C6$, respectively, are other names for axiom scheme (RM) from Ex. 8.4 and axiom (CON) from Def. 8.9, respectively. We can easily see with the results from [50] that system $C + \{C5, C6\}$ is an allowed axiomatization of class $\mathfrak{C}^\kappa$.

[13] The axiom scheme (CM) is defined as $(\phi \to \psi) \wedge (\phi \to \zeta) \Rightarrow (\phi \wedge \psi \to \zeta)$.

[14] More precisely, we use z-entailment of the conditional $\bigwedge(\mathit{as}) \to B$ from a set of conditionals representing the OCF κ through rational closure.

conciseness of the presentation, we present the implementation of the control function by two separate implementations for each type of request. Further, we illustrate the execution of each implementation with our running example.

8.5.1 Query Requests

For *controlled query-answering*, we equip agent $\mathcal{D}$ with Procedure 5 as illustrated by Figure 8.1. By Theorem 8.1, it suffices that this procedure only maintains the view $\mathcal{V} = \mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})$. More specifically, the procedure for controlled query evaluation of the censor `cexec` proceeds as follows.

1. First (line 1-2), the censor determines whether a return value
   $ans \in \{true, false,$
   $undef\}$ is a *possible answer* to the query request $\texttt{que}(A)$, that is, an answer that $\mathcal{A}$ does not rule out given its approximations in $\mathcal{V}$ and being aware that $\mid\!\sim_{\mathcal{D}} \in \mathcal{C}$:
   The censor temporarily refines $\mathcal{V}$ by the answer *ans* (`app_by_que`, Section 8.1.1). Then, the censor checks, whether the refinement still can approximate $\mathcal{D}$'s fixed reasoning, by evaluating the following predicate on the refined view:

$$\texttt{poss}(\mathcal{V}, \mathcal{C}) : \text{ there exists } \mid\!\sim' \text{ possible under } \mathcal{V} \text{ and } \mathcal{C}. \qquad (8.4)$$

2. Next (line 1-2), the censor checks whether returning a possible answer *ans* enables agent $\mathcal{A}$ to skeptically conclude a confidential belief.

3. In case of such a conclusion (line 3), the censor refuses the query.

4. Otherwise (line 6-7), the censor maintains the view, evaluates the query and sends the correct answer to $\mathcal{A}$.

Procedure 5 is adopted from [19] but here the answer reveals belief or non-belief, and the derivation or non-derivation of belief from agent $\mathcal{D}$'s current assertions is described by subsets of $\mathcal{L}_{PL} \times \mathcal{L}_{PL}$ (Section 8.1.1). Meta-inference is prevented as shown by Theorem 8.2 below. The reason is that a query is refused depending on $\mathcal{V}$ and *conf*(*TCP*) only (line 1-5), but not on the epistemic state (used in line 6-7). Note that Procedure 5 does not refine the view after a refusal since by the above considerations about meta-inference a refusal does not provide information about $\mathcal{D}$'s epistemic state. Therefore, $\mathcal{A}$'s approximations and skeptical entailment based thereupon, simulated by the censor during a controlled interaction, do not differ from those during uncontrolled interactions in Section 8.1.

Example 8.8 (Controlled Query-Answering).

Consider the policy conf(TCP) and the first scenario of Ex. 8.1 until the answer undef to $\mathcal{A}$'s query $\texttt{que}(s21)$. *In this situation, the view is*

$$\mathcal{V}_1 = \langle \emptyset, \{(s12,\, s21),\, (s12,\, \neg s21)\},\ \{s12\}\rangle.$$

This situation satisfies the preconditions (8.1) *of the censor. Now, agent $\mathcal{A}$ initiates the query request* $\texttt{que}(s21 \Rightarrow r)$.
Next, the censor checks whether the answer true is possible which means whether there exists possible consequence relation $\mid\!\sim'$ under the refined view

$$\mathcal{V}' = \langle \{(s12,\, s21 \Rightarrow r)\},\ \{(s12,\, s21),\, (s12,\, \neg s21)\},\ \{s12\}\rangle$$

and $\mathcal{C}^{\kappa}$. By Prop. 8.1, this is equivalent to checking $CL(\mathcal{V}') \nvdash_{\mathbf{Ax}} \top \to \bot$. Here, the value true is a possible answer and also the actual answer which further does not reveal whether $\mathcal{D}$ believes r, because $r \notin \mathsf{Bel}(\mid\!\sim_{\mathcal{D}}^{\kappa_1}, \{d12, s12\})$, cf. Sect. 6.2.2. After a check of values false and undef with the same result, the answer true is returned to $\mathcal{A}$ and the view is updated accordingly.

8.5.2 Revision Requests

In a *controlled revision processing*, if it is successful, agent $\mathcal{D}$ adds information $A \in \mathcal{L}_{PL}$ received from agent $\mathcal{A}$ by the request $\texttt{rev}(A)$ to its current assertions. Here, a notification of success/failure implicitly informs agent $\mathcal{A}$ whether $\mathcal{D}$ considers the information A together with its assertions as contradictory or not (Section 6.2.3). Sometimes the notification must be suppressed if it enables $\mathcal{A}$ to skeptically conclude some confidential belief. But checking whether such a suppression is needed must be independent from $\mathcal{D}$ considering $\mathit{as} \cup \{A\}$ contradictory or not. Otherwise, suppressing a notification may leak information about the epistemic state. Such a leakage would be caused by agent $\mathcal{A}$'s meta-inference (about the suppression).

Procedure 5 cexec for controlled query execution

Input: axiomatization $\mathbf{Ax}$, epistemic state $\langle \mathrel{|\!\!\sim}_{\mathcal{D}}, \mathit{as}\rangle$,policy $\mathit{conf}(\mathit{TCP})$, view $\mathcal{V}$, request $\texttt{que}(A)$

Output: notification to $\mathcal{A}$

```
1: for $ans \in \{true, false, undef\}$ do
2:    if $\texttt{poss}(\texttt{app\_by\_que}(A, ans, \mathcal{V}), \mathcal{C})$ and there exists
          $B \in \mathsf{skeptical}_{\mathcal{C}}(\texttt{app\_by\_que}(A, ans, \mathcal{V}))$ with $B \in conf(TCP)$
      then
3:       Notify $\mathcal{A}$: $refuse$. Exit procedure
4:    end if
5: end for
6: $\mathcal{V} := \texttt{app\_by\_que}(A, \texttt{eval}(A)(\mathrel{|\!\!\sim}, as), \mathcal{V})$
7: Notify $\mathcal{A}$: $\texttt{eval}(A)(\mathrel{|\!\!\sim}, as)$
```

The procedure for controlled revision execution of the censor cexec implemented by Procedure 6 proceeds in three steps:

1. First (line 1-3), the censor checks whether through a success notification agent $\mathcal{A}$ is able to skeptically conclude a confidential belief $B \in \mathit{conf}(\mathit{TCP})$ and whether success of the belief revision is possible from agent $\mathcal{A}$'s point of view[15]. In this case, the revision is refused.

2. Otherwise (line 4-6), the censor checks a failure notification analogously.

3. Otherwise (line 7-11), the censor processes the revision, maintains the view and sends a success/failure notification to $\mathcal{A}$.

Example 8.9 (Controlled Revision Execution).

We continue Ex. 8.8 to illustrate the censor for controlled revision processing. Processing agent $\mathcal{A}$'s request $\texttt{rev}(s21)$*, the censor exits in line 2 because otherwise $\mathcal{A}$ would skeptically entail that $\mathcal{D}$ believes $r \in \mathit{conf}(\mathit{TCP})$, cf. Ex. 8.6. The notification is specific for line 2 and therefore reveals to agent $\mathcal{A}$ that the censor exits in this line and, thus, without a revision. Apart from this,*

[15] Otherwise, agent $\mathcal{A}$ already knows that the revision will fail and should be notified accordingly.

Procedure 6 cexec for controlled revision execution

```
Input: axiomatization $\mathbf{Ax}$, epistemic state $\langle \mathrel{|\!\sim}_{\mathcal{D}}, \mathit{as}\rangle$, policy $\mathit{conf}(\mathit{TCP})$,
    view $\mathcal{V}$, request $\texttt{rev}(A)$
Output: notification to $\mathcal{A}$
 1: if $\texttt{poss}(\texttt{app\_by\_rev}(A, \mathit{success}, \mathcal{V}), \mathcal{C})$ and there exists
        $B \in \mathsf{skeptical}_{\mathcal{C}}(\texttt{app\_by\_rev}(A, \mathit{success}, \mathcal{V}))$ with $B \in \mathit{conf}(\mathit{TCP})$
    then
 2:   Notify $\mathcal{A}$: "Revision would breach confidentiality".
      Exit procedure
 3: end if
 4: if $\texttt{poss}(\texttt{app\_by\_rev}(A, \mathit{failure}, \mathcal{V}), \mathcal{C})$ and there exists
        $B \in \mathsf{skeptical}_{\mathcal{C}}(\texttt{app\_by\_rev}(A, \mathit{failure}, \mathcal{V}))$ with $B \in \mathit{conf}(\mathit{TCP})$
    then
 5:   Notify $\mathcal{A}$: "Failure notification would breach confidentiality".
      Exit procedure
 6: end if
    Process revision
 7: if $\texttt{notify}(A)(\mathrel{|\!\sim}, \mathit{as}) = \mathit{success}$ then
 8:   $\mathit{as} := \mathit{as} \cup \{A\}$
 9: end if
10: $\mathcal{V} := \texttt{app\_by\_rev}(A, \texttt{notify}(A)(\mathrel{|\!\sim}, \mathit{as}), \mathcal{V})$
11: Notify $\mathcal{A}$: $\texttt{notify}(A)(\mathrel{|\!\sim}, \mathit{as})$.
```

the notification has no further informational value since, by Asmp. 2, agent $\mathcal{A}$ can freely access all arguments used in the computations up to line 2.

Theorem 8.2 (Confidentiality Preservation)**.**

Agent $\mathcal{D}$ preserves temporary confidentiality towards $\mathcal{A}$ according to Def. 4.6 by means of the two procedures of the censor cexec *under Asmp. 8 and Asmp. 9.*

Sketch of proof. We briefly outline the proof in the appendix. Starting with any initial situation of agent $\mathcal{D}$, after any finite sequence *Int* of requests by agent $\mathcal{A}$, we construct an alternative initial epistemic state $\langle \mathrel{|\!\sim}', \mathit{as}'\rangle$. In the following we outline the construction and its properties:

(1) Invariant. We prove that the censor cexec preserves the following invariant:

$$B \notin \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}^k) \text{ for each } B \in \mathit{conf}(TCP), \tag{8.5}$$

where $\mathcal{V}^k$ denotes the view computed by the censor in the k-th request. The main argument is that the precondition (8.1) and both procedures explicitly ensure the invariant, but the procedures only guarantee the invariant if the predicate $\mathtt{poss}(\mathcal{V}^k, \mathcal{C})$ holds. We will exemplify this argument for the procedure of controlled query execution with a request $\mathtt{que}(A)$. Previously, the view might have been modified only by either a correct answer in case of a previous query request (see line 6) or a correct notification in case of a previous revision request (see line 11). Therefore, we intuitively see that $\mathcal{D}$'s fixed reasoning $\mathrel{|\!\sim}_{\mathcal{D}}$ $\mathcal{D}$ is possible under the view $\mathcal{V}(., \mathit{as}_0, .)^k$ and class $\mathcal{C}$ where $\mathcal{V}(., \mathit{as}_0, .)^k$ corresponds to agent $\mathcal{A}$'s approximations in the k-th request when $\mathcal{D}$'s initial assertions as_0 were visible (Section 8.1). This intuitive argument is formalized in Lemma 8.3 in the appendix which requires class $\mathcal{C}$ to be axiomatized by a set $\mathbf{Ax}$ containing rule scheme (LLE), cf. Assumption 9 and Definition 8.9. Consequently, together with Proposition 8.3 of the appendix, we see that there must exist a possible consequence relation under $\mathcal{V}^k$ and $\mathcal{C}$ so that the predicate $\mathtt{poss}(\mathcal{V}^k, \mathcal{C})$ holds. We remark that Assumption 9 and, thus, Definition 8.9, are essential for the proof of Proposition 8.3. Finally, line 2 of Procedure 5 enforces the invariant (8.5) with $\mathcal{V}^k = \mathtt{app\_by\_que}(A, \mathtt{eval}(A)(\mathrel{|\!\sim}_{\mathcal{D}} \mathcal{D}, \mathit{as}), \mathcal{V}^{k-1})$.

(2) Safe Final Situation. The invariant (8.5) guarantees the existence of a possible consequence relation $\mathrel{|\!\sim}_{\mathcal{D}}{}'$ under the final view $\mathcal{V}_{\mathit{Int}}$ after the interaction and class $\mathcal{C}$ such that $\mathcal{D}$ does not defeasibly conclude B from the visible assertions $\mathcal{C}_{\mathit{Int}}$ after the interaction: $\bigwedge(\mathcal{C}_{\mathit{Int}}) \mathrel{|\!\not\sim}' B$. Thus, in the final alternative situation, $\mathcal{D}$ does not believe B: $B \notin \mathsf{Bel}\mathrel{|\!\sim}_{\mathcal{D}}{}', \mathcal{C}_{\mathit{Int}}$ (Requirement 2(b)[16] of Definition 4.6). The chosen alternative *initial* epistemic state is $\langle \mathrel{|\!\sim}_{\mathcal{D}}{}', \mathcal{C}_0\rangle$. The state $s' = (\mathrel{|\!\sim}_{\mathcal{D}}{}', \mathcal{C}_0, \mathit{conf}(TCP), \langle\rangle, \mathcal{V}_0)$ with initial view $\mathcal{V}_0 = \langle\emptyset, \emptyset, \mathcal{C}_0\rangle$ is a possible initial state of $\mathcal{D}$ from $\mathcal{A}$'s perspective, that is $s' \in \mathcal{St}^{NMR}_{Init}$ which is defined by the predicate $\mathtt{pre}$ in (8.1).

[16]Recall that in this chapter we set $\mathit{conf}(CCP) = \emptyset$.

(3) Indistinguishable by Observations We can prove that agent $\mathcal{A}$ cannot distinguish the alternative and $\mathcal{D}$'s actual initial state after the sequence Int of requests by the observed reactions from agent $\mathcal{D}$ (Requirement 1 of Definition 4.6). The main argument is that $\mathcal{D}$'s reactions with the alternative epistemic state are entirely determined by the view $\mathcal{V}_{\mathit{Int}}$ that the censor has computed according to $\mathcal{D}$'s reactions.

□

8.6 Conclusion

8.6.1 Summary

In this chapter, we constructed a confidentiality preserving agent $\mathcal{D}$ and proved that $\mathcal{D}$'s censor of Section 8.5 effectively enforces $\mathcal{D}$'s confidentiality interests towards agent $\mathcal{A}$. The crucial task of the censor is to simulate $\mathcal{A}$'s postulated skeptical entailment of $\mathcal{D}$'s belief while $\mathcal{D}$ has hidden some part of its current assertions from $\mathcal{A}$. To this end, we assumed that $\mathcal{D}$'s fixed reasoning is an instance of a class of consequence relations that has an allowed axiomatization. The definition of allowed sets of axiom and rule schemes for axiomatizing $\mathcal{D}$'s reasoning class is sufficient to establish our results for the simulation of $\mathcal{A}$, yet general enough to use classes of reasoning well-known from the literature. In Section 8.4, we showed that such an allowed axiomatization provides a means for computing skeptical entailment via deduction. In Section 8.4.2, we discussed how the results presented here extend our preliminary work in [30] where $\mathcal{D}$'s consequence relation is defined by an ordinal conditional function.

8.6.2 Discussion and Future Work

Information Sharing

Beside confidentiality, agent $\mathcal{D}$ is also interested in sharing information with $\mathcal{A}$ to achieve its own goals or joint goals with $\mathcal{A}$. To this aim, $\mathcal{D}$'s censor should not be too restrictive. Generally, $\mathcal{D}$'s censor follows a policy of last-minute intervention like in Chapter 7: Requests should only be refused if the correct notification might harm confidentiality. In future work, we might study and improve upon the optimality of the censor's procedures under this policy as we partly did in Chapter 7.

First, we may study the cooperativeness of the presented censor under information sharing by reviewing the simulation of $\mathcal{A}$ under that aspect since the simulation is a crucial task of the censor. Considering this, agent $\mathcal{D}$ might unnecessarily withhold information if its simulation of $\mathcal{A}$ mistakenly detected a skeptical entailment of confidential belief. A detection is made by $\mathcal{D}$ deducing $CL(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \Vdash_{\mathbf{Ax}} \bigwedge(\mathcal{C}) \to B$ for some confidential belief $B \in \mathit{conf}(\mathit{TCP})$ (Section 8.4). It is an erroneous detection if $\mathcal{A}$ was not able to skeptically conclude B at all, that is, $B \notin \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$ held. Corollary 8.2 verifies that such an erroneous detection may not happen. The result of the corollary relies on the choice of $\mathbf{Ax}$ as an axiomatization of the class $\mathcal{C}$. In particular, the *completeness property* ensures that the above deduction implies the skeptical entailment of B from the view $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})$. Complementary, Theorem 8.1 completes the result of the corollary. It says that skeptical entailment of B from the view $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})$ indeed implies the skeptical entailment $B \in \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$ that models $\mathcal{A}$'s postulated skeptical reasoning.

In the following, we review two results from the literature which indicate how to improve $\mathcal{D}$'s censor under the information sharing aspect.

First, Biskup and Weibert in [34] study confidentiality preserving query-answering for incomplete propositional databases but there the censor is not as restrictive as Procedure 5. In our context, $\mathcal{D}$'s current assertions as may be viewed as an incomplete propositional database and queries to that database are evaluated by propositional entailment $\vdash_{pl}$ (in terms of our definitions, agent $\mathcal{D}$ has belief $\mathsf{Bel}_{\mathcal{D}}^{NMR}(\vdash_{pl}, \mathit{as})$). The idea of a less restrictive censor in [34] is as follows. If for example only returning *true* to query $\mathsf{que}(B)$ was harmful, it would suffice to refuse the answers *true* and, say, *false*. The answer *undef* could be returned safely in contrast to our approach. In this example, a refusal reveals to agent $\mathcal{A}$ that agent $\mathcal{D}$ believes B or believes $\neg B$. This fact can be expressed in propositional modal logic in the special case of [34]. In general ($\mid\!\sim_{\mathcal{D}} \neq \vdash_{pl}$), the fact that $\mathcal{D}$ defeasibly concludes B from as or defeasibly concludes $\neg B$ from as cannot be expressed using the relations $\mathcal{B}^+$ and $\mathcal{B}^-$: $(\bigwedge(\mathcal{C}), B)$ is neither definitely in $\mathcal{B}^+$ nor in $\mathcal{B}^-$. Instead, this fact

may be represented by the formula $(\bigwedge(\mathcal{C}) \to B) \vee (\bigwedge(\mathcal{C}) \to \neg B) \in \mathcal{L}^*_{CL}$ so that Procedure 5 may be enhanced following [34].
Second, we may augment $\mathcal{D}$'s belief $\mathsf{Bel}^{NMR}_{\mathcal{D}}(\mathrel{|\!\sim}_{\mathcal{D}} \mathcal{D}, \mathit{as})$ with weights where a higher weight expresses more confidence in a belief. Weighted belief can be used to specify a finer grained confidentiality policy as we mentioned in Section 6.3. Wiese in [93] uses such policies for query-answering to a possibilistic knowledge base. Confidentiality is preserved by weakening answers which "is a more cooperative way of communication than denying access altogether".

Merging Evidence

Our approach is closely related to the work of Booth and Nittka [37], cf. the paragraph of Section 6.3 on knowledge and reasoning about other agents. However, in their proposal, agent $\mathcal{A}$ gives a '*rational explanation* for its observations of $\mathcal{D}$'s reactions that is different from skeptical entailment. A more significant difference to our contributions is that the revision process in [37] uses linear merging (Section 6.4) of all assertions that $\mathcal{D}$ gathered so far giving precedence to new information if inconsistency must be repaired. For example, agent $\mathcal{D}$ as a manufacturer (Section 6.2.2) may merge the list of the assertions $\langle s11, s21, \neg s11 \vee \neg s21\rangle$ ordered by recency. The merging yields $s21 \wedge (\neg s11 \vee \neg s21) =: A$ with the linear merging operator in [37]. Now in [37], $\mathcal{D}$'s belief are all propositional consequences of the formula A, that is, $\mathsf{Bel}^{NMR}_{\mathcal{D}}(\vdash_{pl}, A)$ in terms of our definitions. Altogether, because of the differences in processing a revision request and in the observer's reasoning, we had to adapt their approach to our scenario and to an observer $\mathcal{A}$ as a skeptical reasoner.
In future work, we may extend agent $\mathcal{D}$'s functionality with the operator of linear merging as we sketched in Section 6.4. For this work, Booth and Nittka's results on the observer's reasoning about the process of linear merging might be useful, especially, for $\mathcal{D}$'s simulation of $\mathcal{A}$'s reasoning about that process.

Conclusion

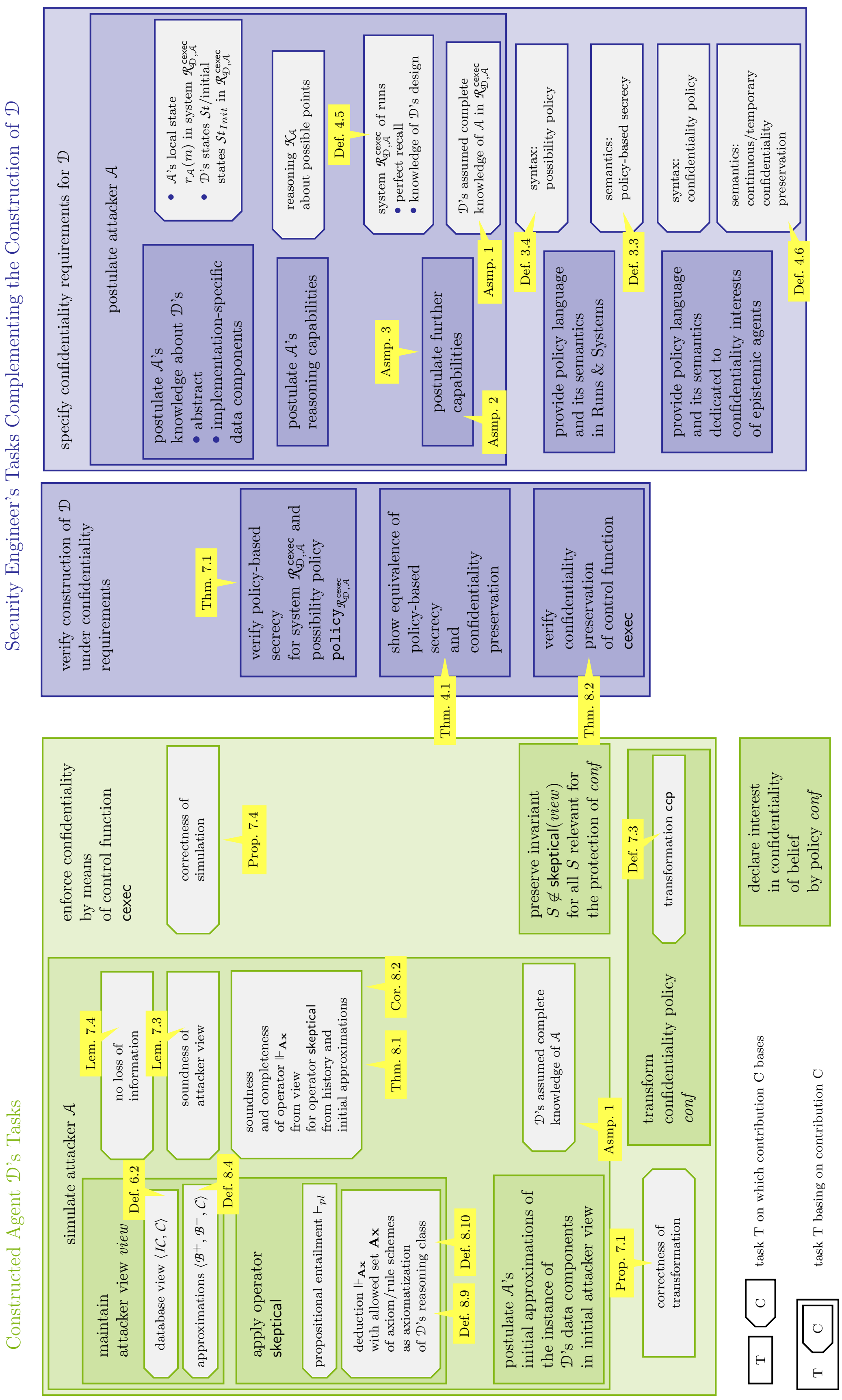
Summary

The leading research question investigated in this thesis was how to enable an epistemic agent $\mathcal{D}$ to enforce its confidentiality interests regarding its belief when it shares information with another epistemic agent $\mathcal{A}$. In our investigation, we focused on agent $\mathcal{D}$'s controlled processing of information received from $\mathcal{A}$ by dedicated *belief change operators*, here, update and revision, beside $\mathcal{D}$'s controlled processing of queries. Figure A.1 shows the main contributions, setting them into the context of the security engineer's tasks and agent $\mathcal{D}$'s tasks. In the following, we will highlight several contributions shown in the figure and make clear their relationships. Then, we will discuss why and in how far our overall contributions are of relevance to the perspective of confidentiality preserving intelligent agents in multiagent environments as outlined in Section 1.1.

Before the construction of a confidentiality preserving agent $\mathcal{D}$, the security engineer provides the agent with means to declare its interests in a specification language with a formal semantics. To this end, in Chapter 3 of Part I, we first introduced possibility policies and their semantics of policy-based secrecy based on our proposed model $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of the agent system in the Runs & Systems framework, following the work [58] by Halpern and O'Neill on secrecy in multiagent systems. Then, we defined the syntax of confidentiality policies which enables agent $\mathcal{D}$ to declare its interests in the confidentiality of current or previous belief. The semantics of confidentiality policies follows the usual confidentiality requirements in the line of research of Controlled Interaction Execution [16, 18]. Finally, we showed how to translate confidentiality policies to possibility policies while preserving the semantics. This translation shows the equivalence of confidentiality preservation and policy-based secrecy in the considered system of agents. Moreover, with this translation, we related the specification of confidentiality to research in information flow control via policy-based secrecy.

A mandatory part of the specification of confidentiality requirements was postulating the capabilities of the attacker $\mathcal{A}$. The security engineer's postulates (Chapter 4) are formalized in the model $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of the agent system in Runs & Systems, comprising postulates about $\mathcal{A}$'s knowledge of $\mathcal{D}$'s data

Figure A.1: Summary of approach and contributions in this thesis


Security Engineer's Tasks Complementing the Construction of $\mathcal{D}$
specify confidentiality requirements for $\mathcal{D}$
postulate attacker $\mathcal{A}$
postulate $\mathcal{A}$'s knowledge about $\mathcal{D}$'s • abstract • implementation-specific data components
• $\mathcal{A}$'s local state $r_{\mathcal{A}}(m)$ in system $\mathcal{R}^{\text{cexec}}_{\mathcal{D},\mathcal{A}}$ • $\mathcal{D}$'s states $\mathcal{S}t$/initial states $\mathcal{S}t_{Init}$ in $\mathcal{R}^{\text{cexec}}_{\mathcal{D},\mathcal{A}}$
postulate $\mathcal{A}$'s reasoning capabilities
reasoning $\mathcal{K}_{\mathcal{A}}$ about possible points
Def. 4.5
system $\mathcal{R}^{\text{cexec}}_{\mathcal{D},\mathcal{A}}$ of runs • perfect recall • knowledge of $\mathcal{D}$'s design
Asmp. 3
postulate further capabilities
Asmp. 2
$\mathcal{D}$'s assumed complete knowledge of $\mathcal{A}$ in $\mathcal{R}^{\text{cexec}}_{\mathcal{D},\mathcal{A}}$
Asmp. 1
provide policy language and its semantics in Runs & Systems
syntax: possibility policy
Def. 3.4
semantics: policy-based secrecy
Def. 3.3
provide policy language and its semantics dedicated to confidentiality interests of epistemic agents
syntax: confidentiality policy
semantics: continuous/temporary confidentiality preservation
Def. 4.6
verify construction of $\mathcal{D}$ under confidentiality requirements
Thm. 7.1
verify policy-based secrecy for system $\mathcal{R}^{\text{cexec}}_{\mathcal{D},\mathcal{A}}$ and possibility policy $\texttt{policy}_{\mathcal{R}^{\text{cexec}}_{\mathcal{D},\mathcal{A}}}$
show equivalence of policy-based secrecy and confidentiality preservation
Thm. 4.1
verify confidentiality preservation of control function cexec
Thm. 8.2
Constructed Agent $\mathcal{D}$'s Tasks
enforce confidentiality by means of control function cexec
correctness of simulation
Prop. 7.4
preserve invariant $S \not\in$ skeptical$(view)$ for all S relevant for the protection of $conf$
transform confidentiality policy $conf$
transformation ccp
Def. 7.3
declare interest in confidentiality of belief by policy $conf$
simulate attacker $\mathcal{A}$
Lem. 7.4
no loss of information
Lem. 7.3
soundness of attacker view
soundness and completeness of operator $\Vdash_{\mathbf{Ax}}$ from view for operator skeptical from history and initial approximations
Cor. 8.2
Thm. 8.1
$\mathcal{D}$'s assumed complete knowledge of $\mathcal{A}$
Asmp. 1
maintain attacker view $view$
Def. 6.2
database view $\langle \mathcal{IC}, \mathcal{C} \rangle$
approximations $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$
Def. 8.4
apply operator skeptical
propositional entailment $\vdash_{pl}$
deduction $\Vdash_{\mathbf{Ax}}$ with allowed set $\mathbf{Ax}$ of axiom/rule schemes as axiomatization of $\mathcal{D}$'s reasoning class
Def. 8.10
Def. 8.9
postulate $\mathcal{A}$'s initial approximations of the instance of $\mathcal{D}$'s data components in initial attacker view
Prop. 7.1
correctness of transformation
T C task T on which contribution C bases
T C task T basing on contribution C

components, $\mathcal{A}$'s reasoning about $\mathcal{D}$'s state and about further capabilities of $\mathcal{A}$. Chapter 7 and Chapter 8 detailed the postulates of $\mathcal{A}$'s knowledge about the implementation of $\mathcal{D}$'s data components.

In Part II, we presented the construction of agent $\mathcal{D}$ and its tasks for enforcing confidentiality in two exemplary implementations of $\mathcal{D}$'s belief with different belief change operators: a complete propositional database with view update transactions on the one hand (Chapter 7) and nonmonotonic reasoning on current assertions with belief revision on the other hand (Chapter 7). The defender $\mathcal{D}$'s most involved task for enforcing confidentiality is to simulate the postulated attacker $\mathcal{A}$. The postulated attacker $\mathcal{A}$ is a skeptical reasoner with operator $\mathcal{K}_{\mathcal{A}}$ about points of the system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$. Consequently, the simulation of $\mathcal{A}$ needs a representation *view* of $\mathcal{A}$'s information $r_{\mathcal{A}}(m)$ about the system in the logic used for simulation; an associated operator $\mathsf{skeptical}$ for reasoning in the logic; and finally an account for the further postulates about $\mathcal{A}$ as formalized in system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$.
The simulation has to fulfill additional requirements so that by means of simulation $\mathcal{D}$ is enabled to effectively enforce confidentiality. In the following, we will recapitulate the requirements made in this thesis and their importance for effectively enforcing the agent's confidentiality aims. In general, the requirements should ensure the following principle: Every information that $\mathcal{A}$ has gained about $\mathcal{D}$'s epistemic state by its observations and reasoning about system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ must be represented in the attacker view *view* when applying operator $\mathsf{skeptical}$. The idea is that, following this principle in the simulation, $\mathcal{D}$ is enabled to enforce confidentiality by preserving an invariant on the attacker view by means of simulation. Possibly, $\mathcal{D}$ needs to transform parts of its confidentiality policy to do so. In Chapter 7 and Chapter 8, we elaborated on this principle in different ways. In Chapter 7, we defined the correctness of the simulation of $\mathcal{A}$ which makes precise this principle and is achieved by $\mathcal{D}$ ensuring no loss information and the soundness of the attacker view during attacker view maintenance. In Chapter 8, the simulation of $\mathcal{A}$ involved the simulation of $\mathcal{A}$'s skeptical reasoning about $\mathcal{D}$'s nonmonotonic reasoning in the situation that $\mathcal{D}$ has hidden parts of its assertions and its consequence relation from $\mathcal{A}$. As a solution, we first introduced the operator $\mathsf{skeptical}$ used for simulation as a model of $\mathcal{A}$'s reasoning about $\mathcal{D}$'s uncontrolled processing of

requests in the system. However, we did not explicitly relate the simulation to $\mathcal{A}$'s postulated reasoning $\mathcal{K}_{\mathcal{A}}$. Then, we proposed to compute the reasoning with operator $\mathsf{skeptical}$ from the initial approximations and the current history by deduction with an allowed axiomatization $\mathbf{Ax}$ of $\mathcal{D}$'s reasoning class. The deduction starts with conditional formulas defined from the attacker view that assumes no hidden assertions. Finally, we showed that the deduction is equivalent to reasoning by the operator $\mathsf{skeptical}$ and thus follows the above principle – except that the postulated reasoning $\mathcal{K}_{\mathcal{A}}$ involves reasoning about the execution of the control procedures whereas $\mathsf{skeptical}$ does not.
As a further contribution, as part of the security engineer's task, we proved the effectiveness of $\mathcal{D}$'s control procedures to enforcing confidentiality during its interaction with $\mathcal{A}$. On the one hand, in Chapter 7, in the proof we verified that policy-based secrecy holds in the constructed agent system for the translated confidentiality policy and then we used the equivalence of the verification to the proof of the confidentiality requirement. This way we could base the proof on our study of the correctness of the simulation in the same chapter. Then, in Chapter 8, the control procedures were explicitly shown to fulfill the confidentiality requirement. The proof bases on the results for the simulation of $\mathcal{A}$ by deduction with an allowed axiomatization.
As a closing remark, we want to point out that the effectiveness of the control procedures must be viewed in the light of the security engineer's postulate of $\mathcal{D}$'s complete knowledge of $\mathcal{A}$, in particular, of $\mathcal{A}$'s initial approximations. The justification of this postulate needs an appropriate initial attacker view postulated by $\mathcal{D}$ as $\mathcal{A}$'s initial approximations.

Future Work

Finally, we will evaluate the contributions of this thesis under the perspective of designing intelligent agents with confidentiality interests in multiagent environments as anticipated in Section 1.1. In this perspective, we consider the design of an intelligent agent according to the BDI-agent model (**B**eliefs, **D**esires, **I**ntentions) as outlined in Figure A.2. Guided by the figure, we will point to open problems and the impact of our contributions to these problems along three aspects: the internals of the BDI-agent, its interface to the environment and the environment itself. A forth aspect is the applicability of our approach in realistic agent scenarios.

Internals Our contributions focus on belief change and belief representation, abstracting away other internal components of a BDI-agent related to the activities of deliberation and planning which enable the agent to decide on its actions in a goal-directed way (see the discussion in Section 5.2). The figure outlines the information flow between the internal components of the BDI-agent $\mathcal{D}$. Information flow control during the processing of belief change is a major contribution of this thesis as summarized in the previous section. The major principle to handle information flow control in face of the complexity of the agent's functionality was the clear separation of tasks, mainly, belief change by agent $\mathcal{D}$, simulation by $\mathcal{D}$ of the attacker on basis of the attacker view, release of information to the attacker view by $\mathcal{D}$ in compliance with its confidentiality policy and the prevention of any information flow otherwise by design of $\mathcal{D}$'s control procedures by the security engineer.

In future work, we need to manage information flow control in the even more complex internals of a BDI-agent. To this end, we may first apply our solution for the belief change component and the belief representation of Chapter 8 which allows to choose the class of $\mathcal{D}$'s reasoning flexibly given that the class has an allowed axiomatization. Then, we may try to control the information flow in the other components by the above separation of tasks and, in particular, to use the same approach for other belief change operators (see the overview in Section 6.3 to Section 6.4 and a discussion in Section 8.6.2). Notably, in research on information flow control we so far found no results that are apt to significantly contribute to achieve information flow control for a BDI-agent (see the discussion in Section 5.2).

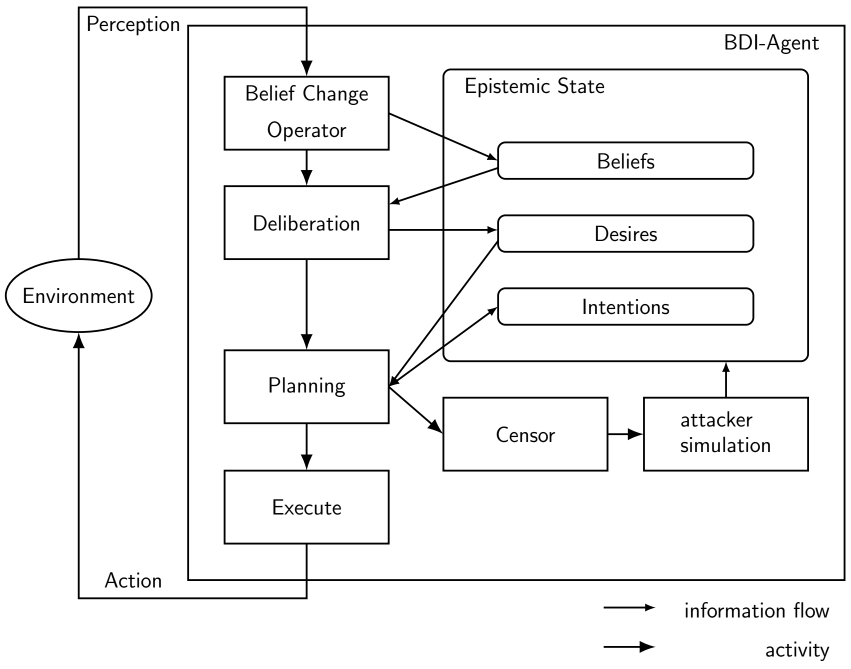
Beside confidentiality, the agent aims at cooperative information sharing with other agents. This aim might be expressed by desires of the BDI-agent and might further require the agent's estimate of the information need of other agents, for example, to achieve joint goals (see the discussion in Section 7.6.2 on availability policies). Our contributions include a first analysis of one of the proposed control functions under the aspect of cooperative information sharing in Section 7.5. We further discussed the improvement of the control functions of Chapter 8 under this aspect in Section 8.6.2.

Interface In this thesis, we chose types of interaction for information sharing in general, namely, query, update and revision requests. These types and the processing of the requests might be fundamental to other types of interaction such as negotiation as anticipated in Section 6.3. Thus, our contributions present a first essential step towards an extended interaction interface starting with the extension sketched in Section 6.4. In particular, in that sketch we considered the processing of update and revision requests by prioritized merging, the control of which may be based on the results of Chapter 8. Further, Section 6.3 gave an overview on other communication languages and the impact of choosing a language on the agent's aim of confidentiality preservation.

Environment Agent $\mathcal{D}$'s environment in this thesis includes only one other agent $\mathcal{A}$ to interact with and the world in general about which $\mathcal{D}$ has belief. On the one hand, we considered agent $\mathcal{D}$ to reason about an inaccessible and dynamic world (according to the terminology in [94]). Thus, $\mathcal{D}$'s information about the world might be incomplete and not up-to-date (or not reliable because of the lack of credibility of the sources of information, cf. Section 6.4) and the world might change beyond $\mathcal{D}$'s control. On the other hand, we considered that the state of $\mathcal{A}$ is accessible to $\mathcal{D}$ by Assumption 1 and that $\mathcal{A}$'s perception of $\mathcal{D}$'s belief is totally under $\mathcal{D}$'s control. In Section 5.2, we briefly discussed a different situation in systems with many agents. Especially, agent $\mathcal{D}$ might be uncertain about $\mathcal{A}$'s state and thus about $\mathcal{A}$'s conclusions about $\mathcal{D}$'s belief in that state. Moreover, due to lack of control over the dissemination of information in the multiagent system agent $\mathcal{D}$ might be confronted with a violation of its confidentiality interests. Under these additional challenges, agent $\mathcal{D}$ might not be able to enforce its confidentiality policy by maintaining the invariant $S \notin \mathsf{skeptical}(\mathit{view})$ for pieces S of information relevant for protecting confidential belief. Instead, the agent must be able to reason about possible violations of its confidentiality policy under uncertainty about the states of the other agents. Being autonomous, the agent will utilize this reasoning ability to plan its actions in compliance with its confidentiality policy while during planning the agent reasonably deals with conflicting desires such as cooperative information sharing.

Applicability In [31], we started to model an application scenario of negotiating agents for supply contracts which we took up as a running example in this thesis. In our ongoing work, we are extending this scenario to negotiating BDI-agents in e-marketplaces.
Moreover, we tested the proposed control procedures in prototypical implementations. First, agent $\mathcal{D}$'s functionality as presented in Chapter 7 has been implemented as part of a prototype for Controlled Interaction Execution [18]. The prototype is a front-end to the Oracle database management system for which additionally the interface of query and view update transaction requests of Chapter 7 has been implemented together with the control functions for processing these requests. Several example scenarios have been successfully tested in that prototype.
Second, agent $\mathcal{D}$'s functionality of Chapter 8 has been integrated into the Angerona multiagent platform [75] as a plug-in. The Angerona platform implements a BDI-agent model as outlined in Figure A.2. Implementing the plug-in, the agent's design of Chapter 8 had to be adapted to the Angerona agent model. The plug-in uses the solver KLMLean [55] for the simulation of $\mathcal{A}$'s reasoning while $\mathcal{D}$'s nonmonotonic reasoning is implemented by an ordinal conditional function using the Tweety library. Several experiments were successfully run with the supply chain scenario of Section 6.2.2.

Figure A.2: Outline of a BDI-agent following the design of Angerona agents [75]



Proofs

Part I

Proposition 3.1

Let f be a $\mathcal{D}$-information function and $\texttt{policy}_f$ *the resulting policy given by Equation* (3.1)*. Then, agent $\mathcal{D}$ maintains total f-secrecy with respect to agent $\mathcal{A}$ in $\mathcal{R}$ iff $\mathcal{D}$ maintains* $\texttt{policy}_f$*-based secrecy with respect to $\mathcal{A}$ in $\mathcal{R}$.*

Proof Consider an abstract information value $v \in V$. We rewrite the pre-image of f in the value $v \in V$ as follows:

$$\begin{aligned} f^{-1}(v) &= \{(r', m') \in \mathcal{PT}(\mathcal{R}) \mid f(r', m') = v\} \\ &= \bigcup_{(r',m'):f(r',m')=v} \mathcal{K}_{\mathcal{D}}(r', m') \\ &= I_v && \text{by (3.1) on p. 41.} \end{aligned}$$

The second equality holds because by Definition 2.2 f only depends on $\mathcal{D}$'s state. Formally, from $(r, m) \in \mathcal{K}_{\mathcal{D}}(r', m')$ follows the equality $r_{\mathcal{D}}(m) = r'_{\mathcal{D}}(m')$ which implies $f(r, m) = f(r', m')$ by Definition 2.2.

Altogether, we immediately see that for all points $(r, m) \in \mathcal{PT}(\mathcal{R})$ and values $v \in V$ holds

$$\mathcal{K}_{\mathcal{A}}(r, m) \cap f^{-1}(v) \neq \emptyset \qquad \text{iff} \qquad \mathcal{K}_{\mathcal{A}}(r, m) \cap I_v \neq \emptyset.$$

This shows the equivalence of the requirements in Definition 2.3 and Definition 3.3.

$\square$(proof Prop. 3.1)

Proposition 3.3

Let $\mathcal{R}$ be a system and $\texttt{policy}\mathcal{R}$ *an $\mathcal{R}$-possibility policy defined over $\mathcal{R}$. Based on this policy, we define a possibility policy* $\texttt{policy}$ *of Def. 3.2 by*

$$\texttt{policy}(r,m) = \{\mathcal{PT}(R) \mid R \in \texttt{policy}\mathcal{R}(r,m)\}.$$

Then, agent $\mathcal{D}$ maintains $\texttt{policy}$*-based secrecy towards agent $\mathcal{A}$ iff agent $\mathcal{D}$ maintains* $\texttt{policy}\mathcal{R}$*-based run-based secrecy towards agent $\mathcal{A}$.*

Only-If Part:
Policy-based Secrecy Implies Policy-based Run-based Secrecy

Consider a point $(r,m) \in \mathcal{PT}(\mathcal{R})$ and a property of runs $R \in \texttt{policy}\mathcal{R}(r,m)$. Since agent $\mathcal{D}$ enforces policy-based secrecy towards $\mathcal{A}$ with respect to the possibility policy $\texttt{policy}$ as defined from $\texttt{policy}\mathcal{R}$ in the claim of the proposition, there exists a point

$$(r',m') \in \mathcal{K}_{\mathcal{A}}(r,m) \cap \mathcal{PT}(R).$$

In the following, we argue that r' is also an alternative run that proves the enforcement of policy-based run-based secrecy, that is $r' \in \mathcal{R}(\mathcal{K}_{\mathcal{A}}(r,m)) \cap R$. First, by the definition of operator $\mathcal{R}$ in (2.4) on p. 29 it holds $r' \in \mathcal{R}(\mathcal{K}_{\mathcal{A}}(r,m))$ since $(r',m') \in \mathcal{K}_{\mathcal{A}}(r,m)$ holds.
Second, $r' \in R$ since $(r',m') \in \mathcal{PT}(R) = \{(r'',m'') \mid r'' \in R, m'' \in \mathbb{N}_0\}$.

If Part:
Policy-based Run-based Secrecy implies Policy-based Secrecy

Again, consider a point $(r,m) \in \mathcal{PT}(\mathcal{R})$ and a property $I \in \texttt{policy}(r,m)$ with $I = \mathcal{PT}(R)$ for some $R \in \texttt{policy}\mathcal{R}(r,m)$ by definition of $\texttt{policy}$. As agent $\mathcal{D}$ enforces policy-based run-based secrecy towards $\mathcal{A}$, there exists a run

$$r' \in \mathcal{R}(\mathcal{K}_{\mathcal{A}}(r,m)) \cap R.$$

Next, we claim and prove that there exists a time $m' \in \mathbb{N}_0$ such that $(r',m') \in \mathcal{K}_{\mathcal{A}}(r,m) \cap I$.

First, according to (2.4) on p. 29 the fact $r' \in \mathcal{R}(\mathcal{K}_{\mathcal{A}}(r,m))$ means that there exists a time $m' \in \mathbb{N}_0$ such that $(r',m') \in \mathcal{K}_{\mathcal{A}}(r,m)$.
Second, $r' \in R$ implies that
$(r',m') \in \mathcal{PT}(R) = \{(r'',m'') \mid r'' \in R, m'' \in \mathbb{N}_0\} = I$.

□(proof Prop. 3.3)

Theorem 4.1

Let $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ be a system of defender $\mathcal{D}$ and attacker $\mathcal{A}$. Then, agent $\mathcal{D}$ maintains $\mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}$-based secrecy with respect to agent $\mathcal{A}$ according to Def. 3.3 iff cexec *preserves continuous/temporary confidentiality according to Def. 4.6.*

Only-If Part:
Policy-based Secrecy Implies Confidentiality Preservation

Following Definition 4.6 we assume any initial abstract state $s_{\mathcal{D}} \in \mathcal{St}_{Init}$ and a sequence $\mathit{Int} \in \mathcal{Req}^n$ of requests with length n as input to the control function cexec of agent $\mathcal{D}$.
For later reference we write $s_{\mathcal{D}} = (bknowl_0, eknowl_0, conf_0, hist_0, view_0)$.
For each potential secret $S \in conf_0(CCP) \cup conf_0(TCP)$ we now specify alternative initial background knowledge $bknowl'_0$ and evidential knowledge $eknowl'_0$ that verify confidentiality preservation of S according to Definition 4.6.
Consider run $r^{\mathit{Int},s_{\mathcal{D}}}$. We recall that $bknowl(r^{\mathit{Int},s_{\mathcal{D}}}, m)$ for example denotes $\mathcal{D}$'s background knowledge in run $r^{\mathit{Int},s_{\mathcal{D}}}$ at time m and $\mathcal{D}$'s other components in the run are denoted likewise. Especially, $bknowl(r^{\mathit{Int},s_{\mathcal{D}}}, 0)$ denotes $bknowl_0$.
Since in the system agent $\mathcal{D}$ maintains policy-based secrecy (Definition 3.3) with possibility policy $\mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}$ (Definition 4.7) we obtain that

$$\begin{array}{rl}
\text{if} & S \in conf_0(CCP) \cup conf_0(TCP) \\
\text{then by Asmp. 5} & S \in conf(CCP)(r^{\mathit{Int},s_{\mathcal{D}}}, n) \cup conf(TCP)(r^{\mathit{Int},s_{\mathcal{D}}}, n) \\
\text{and there exists} & I_S \in \mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}(r^{\mathit{Int},s_{\mathcal{D}}}, n) \\
\text{such that} & \\
\text{there exists} & (r',m') \in \mathcal{K}_{\mathcal{A}}(r^{\mathit{Int},s_{\mathcal{D}}}, n) \cap I_S.
\end{array} \tag{6}$$

At this place, it is important to note that the confidentiality policy *conf* which defines $\texttt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}$ is constant during the run by Assumption 5.
We define the initial alternative belief components bknowl'_0 and eknowl'_0 as the respective components in agent $\mathcal{D}$'s local state $r'_{\mathcal{D}}(0)$.
Now we study the execution of cexec on state $r'_{\mathcal{D}}(0)$ with interaction $\boldsymbol{\mathit{Int}}$ and show both the indistinguishability property and the confidentiality property from Definition 4.6.

Indistinguishability

In the possible point $(r', m') \in \mathcal{K}_{\mathcal{A}}(r^{\mathit{Int},s_{\mathcal{D}}}, n)$ of (6) agent $\mathcal{A}$ is in the same local state as in the considered point $(r^{\mathit{Int},s_{\mathcal{D}}}, n)$ by Equation 2.3 on page 24. Therefore, since agent $\mathcal{A}$ has perfect recall (Assumption 3), the sequences of agent $\mathcal{A}$'s local state in run r' up to time m' and in run $r^{\mathit{Int},s_{\mathcal{D}}}$ up to time n agree when ignoring subsequent duplicate states. Immediately, perfect recall together with (6) ensures

$$\mathit{conf}(r',0) = \mathit{conf}(r^{\mathit{Int},s_{\mathcal{D}}},0) \qquad \mathit{hist}(r',0) = \mathit{hist}(r^{\mathit{Int},s_{\mathcal{D}}},0)$$
$$\mathit{view}(r',0) = \mathit{view}(r^{\mathit{Int},s_{\mathcal{D}}},0).$$

The complete line of argumentation is similar. Since the control function tracks each request in the history (Definition 4.3), there are no duplicate states of agent $\mathcal{A}$ up to time m' or up to time n respectively (the system is synchronous [58]) and, thus, by perfect recall it holds for all $0 \leq k \leq n$

$$m' = n \qquad \mathit{conf}(r',k) = \mathit{conf}(r^{\mathit{Int},s_{\mathcal{D}}},k) \qquad \mathit{hist}(r',k) = \mathit{hist}(r^{\mathit{Int},s_{\mathcal{D}}},k) \tag{7}$$
$$\mathit{view}(r',k) = \mathit{view}(r^{\mathit{Int},s_{\mathcal{D}}},k).$$

Hence, both runs must be defined over the same interaction sequence up to time n, since requests are recorded in the history, that is

$$\theta'_k = \theta_k \text{ for all } 1 \leq k \leq n. \tag{8}$$

Finally, we use that the system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ simulates the control function

(Definition 4.5), that is for all $1 \leq k \leq n$ we obtain

$$\begin{aligned}
&(bknowl'_k, eknowl'_k, conf'_k, hist'_k, view'_k) = r'_{\mathcal{D}}(k) && (9)\\
&= \mathsf{cexec}(r'_{\mathcal{D}}(k-1), \theta'_k) && \text{by Def. 4.5}\\
&= \mathsf{cexec}(r'_{\mathcal{D}}(k-1), \theta_k) && \text{by (8)}\\
&= \mathsf{cexec}((bknowl'_{k-1}, eknowl'_{k-1}, conf_{k-1}, hist_{k-1}, view_{k-1}), \theta_k) && \text{by (7)}
\end{aligned}$$

and

$$\begin{aligned}
&(bknowl_k, eknowl_k, conf_k, hist_k, view_k) = r_{\mathcal{D}}^{Int,s_{\mathcal{D}}}(k)\\
&= \mathsf{cexec}(r_{\mathcal{D}}^{Int,s_{\mathcal{D}}}(k-1), \theta_k) && \text{by Def. 4.5}\\
&= \mathsf{cexec}((bknowl_{k-1}, eknowl_{k-1}, conf_{k-1}, hist_{k-1}, view_{k-1}), \theta_k).
\end{aligned}$$

Together with (7), these equalities show the indistinguishability property of Definition 4.6.

Confidentiality

$\boxed{\textbf{Case } 1}$ $S \in conf_0(TCP)$

By (6) it holds $(r', m') \in I_S$ where $m' = n$ by (7). In this case, the set I_S defined by the possibility policy in Definition 4.7 ensures by (4.1) on p. 69 that $S \notin \mathsf{Bel}_{\mathcal{D}}(bknowl(r', n), eknowl(r', n))$ holds where $bknowl(r', n) = bknowl'_n$ and $eknowl(r', n) = eknowl'_n$ are in the output state of the control function at time n as in (9).

$\boxed{\textbf{Case } 2}$ $S \in conf_0(CCP)$

In this case, the fact that $(r', n) \in I_S$ holds implies that $r' \in R_{S,n}^{CCP}$ of Equation 4.2 on p. 70 holds. Following that equation and (9), we obtain $S \notin \mathsf{Bel}_{\mathcal{D}}(bknowl'_k, eknowl'_k)$ for all $0 \leq k \leq n$.

If Part: Confidentiality Preservation Implies Policy-based Secrecy

Let $(r, m) \in \mathcal{PT}(\mathcal{R}_{\mathcal{D},\mathcal{A}}^{\mathsf{cexec}})$. Consider a property $I \in \texttt{policy}_{\mathcal{R}_{\mathcal{D},\mathcal{A}}^{\mathsf{cexec}}}(r, m)$. To show policy-based secrecy of Definition 3.3, we have to find a point in

$$\mathcal{K}_{\mathcal{A}}(r, m) \cap I. \tag{10}$$

As previously explained, we can retrieve the sequence of requests from a run because it is recorded in the agents' history components. Let $\mathit{Int} \in \mathcal{Req}^m$ be the sequence of requests in run r up to time m. To find the required point, we consider the execution of the control function cexec on agent $\mathcal{D}$'s initial state $r_{\mathcal{D}}(0)$ with the sequence Int.
We write $r_{\mathcal{D}}(0) = s_{\mathcal{D}} = (\mathit{bknowl}_0, \mathit{eknowl}_0, \mathit{conf}_0, \mathit{hist}_0, \mathit{view}_0)$ for later reference.
By Definition 4.7 the property $I \in \mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}(r,m)$ is either of the form $\mathcal{PT}(R^{CCP}_{S,m})$ in (4.2) on p. 70 with $S \in \mathit{conf}(\mathit{CCP})(r,m)$ or of the form I^{TCP}_S in (4.1) on p. 69 with $S \in \mathit{conf}(\mathit{TCP})(r,m)$. Since the confidentiality policy is constant over the run r by Assumption 5, the confidential belief S defining I is from the initial confidentiality policy conf_0 in run r.
By presupposition, cexec preserves confidentiality (Definition 4.6) for each confidential belief in conf_0, in particular, for the confidential belief S defining property I. Confidentiality preservation for S is guaranteed by alternative initial background knowledge bknowl'_0 and evidential knowledge eknowl'_0 with properties we recall and use in the sequel.
We write $s'_{\mathcal{D}} = (\mathit{bknowl}'_0, \mathit{eknowl}'_0, \mathit{conf}_0, \mathit{hist}_0, \mathit{view}_0)$ for later reference.
First, the execution of cexec on the initial state $s'_{\mathcal{D}}$ with interaction Int is indistinguishable from that on state $s_{\mathcal{D}}$ with interaction Int. In particular, this means that at time m, considered in (10), it holds

$$\mathit{conf}'_m = \mathit{conf}_m \qquad \mathit{hist}'_m = \mathit{hist}_m \qquad \mathit{view}'_m = \mathit{view}_m, \tag{11}$$

meaning that the outputs of cexec after the m-th request are equal by Definition 4.6. Since by Definition 4.5 the runs of $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ are defined by cexec, we obtain

$$\begin{aligned} r_{\mathcal{A}}(m) &= r^{\mathit{Int},s_{\mathcal{D}}}_{\mathcal{A}}(m) && \text{by choice of } \mathit{Int} \text{ and } s_{\mathcal{D}} \\ &= (\mathit{conf}_m, \mathit{hist}_m, \mathit{view}_m) && \text{by Def. 4.5} \\ &= (\mathit{conf}'_m, \mathit{hist}'_m, \mathit{view}'_m) && \text{by (11)} \\ &= r^{\mathit{Int},s'_{\mathcal{D}}}_{\mathcal{A}}(m) && \text{by Def. 4.5.} \end{aligned}$$

and, thus, $(r^{\mathit{Int},s'_{\mathcal{D}}}, m) \in \mathcal{K}_{\mathcal{A}}(r,m)$ by Equation 2.3 on page 24. This shows that the point $(r^{\mathit{Int},s'_{\mathcal{D}}}, m)$ is a possible point from agent $\mathcal{A}$'s perspective at (r,m) as required in (10).

Finally, we consider the protection of the secret S.

$\boxed{\textbf{Case } 1}$ $S \in \mathit{conf}_0(\mathit{TCP})$

By Definition 4.6 we know that in the alternative situation after the m-th requests it holds $S \notin \mathsf{Bel}_{\mathfrak{D}}(\mathit{bknowl}'_m, \mathit{eknowl}'_m)$. This means that $(r^{\mathit{Int}, s'_{\mathfrak{D}}}, m) \in I$ by (4.1) on p. 69.

$\boxed{\textbf{Case } 2}$ $S \in \mathit{conf}_0(\mathit{CCP})$

In this case, by Definition 4.6 we obtain that in the alternative situation until the m-th request it holds $S \notin \mathsf{Bel}(\mathit{bknowl}'_k, \mathit{eknowl}'_k)$ for all $0 \leq k \leq m$. It follows that $r^{\mathit{Int}, s'_{\mathfrak{D}}} \in R^{CCP}_{S,m}$ holds by Equation 4.2 on page 70. This means that $(r^{\mathit{Int}, s'_{\mathfrak{D}}}, m) \in I = \mathcal{PT}(R^{CCP}_{S,m})$.

$\square$(proof Thm. 4.1)

Part II

Lemma 6.1

Let $\mathcal{U}$ be a set of literals over pairwise distinct atoms. Further let $\mathcal{db}$ be a database instance such that $\mathcal{db} \not\models_{PL} L$ for all $L \in \mathcal{U}$. Then, it holds that $\mathcal{db} = (\mathcal{db} \bullet \mathcal{U}) \bullet \neg(\mathcal{U})$.

Inclusion $\subseteq$

Let $a \in \mathcal{db}$. By presupposition there are only the following two cases.

$\boxed{\textbf{Case } 1}$ $\neg a \in \mathcal{U}$

Then $a \in \neg(\mathcal{U})$ by (6.3) and it follows by Definition 6.4 of the database update $\bullet$ that

$$(\mathcal{db} \bullet \mathcal{U}) \bullet \neg(\mathcal{U}) \supseteq \neg(\mathcal{U}) \cap \mathcal{At} \ni a.$$

$\boxed{\textbf{Case } 2}$ $a \notin \neg(\mathcal{U})$ and $a \notin \mathcal{U}$

In this case, by (6.3) it holds $\neg a \notin \neg(\mathcal{U})$ and $\neg a \notin \mathcal{U}$. Thus, $a \in \{b \in \mathcal{db} \mid \neg b \notin \mathcal{U}\} = \mathcal{db} \setminus \{b \mid \neg b \in \mathcal{U}\} \subseteq \mathcal{db} \bullet \mathcal{U}$. Thus, it holds $a \in \{b \in \mathcal{db} \bullet \mathcal{U} \mid \neg b \notin \neg(\mathcal{U})\} \subseteq (\mathcal{db} \bullet \mathcal{U}) \bullet \neg(\mathcal{U})$.

Inclusion $\supseteq$

Let $a \notin \mathcal{db}$. By presupposition there are only the following two cases.

$\boxed{\textbf{Case } 1}$ $a \in \mathcal{U}$

In this case, by (6.3) it follows that $\neg a \in \neg(\mathcal{U})$ holds, but $a \notin \neg(\mathcal{U})$. Thus, we obtain that

$$a \notin (\mathcal{At} \cap \neg(\mathcal{U})) \cup \{b \in \mathcal{db} \bullet \mathcal{U} \mid \neg b \notin \neg(\mathcal{U})\} = (\mathcal{db} \bullet \mathcal{U}) \bullet \neg(\mathcal{U}).$$

$\boxed{\textbf{Case } 2}$ $a \notin \neg(\mathcal{U})$ and $a \notin \mathcal{U}$

Then, Definition 6.4 of the database update $\bullet$ implies $a \notin \mathcal{db} \bullet \mathcal{U}$ and thus $a \notin (\mathcal{db} \bullet \mathcal{U}) \bullet \neg(\mathcal{U})$.

□(proof Lem. 6.1)

Proposition 6.1

Let $\langle \mathcal{IC}, \mathcal{db}\rangle$ be a database over a finite alphabet $\mathcal{At}$ and $\bullet_{db}$ the database update[1] of Def. 6.4. Further, we define a transformation t of a database into an epistemic state of Def. 6.7 as follows:

$$\begin{array}{lccl} & \mathsf{t}(\langle \mathcal{IC}, \mathcal{db}\rangle) & = & \langle \mathsf{t}(\mathcal{IC}), \mathsf{t}(\mathcal{db})\rangle \\ \textit{with} & (A,B) \in \mathsf{t}(\mathcal{IC}) & \textit{iff} & \{A\} \cup \mathcal{IC} \vdash_{pl} B \\ \textit{and} & \mathsf{t}(\mathcal{db}) & = & \mathcal{db} \cup \{\neg a \mid a \in \mathcal{At} \setminus \mathcal{db}\}. \end{array}$$

Let L be a literal over $\mathcal{At}$. Then, it holds

1. $\mathsf{t}(\langle \mathcal{IC}, \mathcal{db}\rangle)$ *is an epistemic state according to Def. 6.7,*
2. $\mathsf{Bel}^{DB}_{\mathfrak{D}}(\langle \mathcal{IC}, \mathcal{db}\rangle) = \mathsf{Bel}^{NMR}_{\mathfrak{D}}(\mathsf{t}(\langle \mathcal{IC}, \mathcal{db}\rangle))$,
3. $\mathsf{t}(\langle \mathcal{IC}, \mathcal{db}\rangle \bullet_{db} \{L\}) = \mathsf{t}(\langle \mathcal{IC}, \mathcal{db}\rangle) \bullet_{simp} L$.

Before the proof of the proposition, we introduce and prove four claims:

Claim 1 $I^{\mathcal{db}} \models_{PL} \mathsf{t}(\mathcal{db})$

The claim follows right from the definitions of the transformation t in Proposition 6.1 and the propositional interpretation $I^{\mathcal{db}}$. We recall that the interpretation is inductively defined by $\mathcal{db}$ with the base case of atoms $a \in \mathcal{At}$ being $I^{\mathcal{db}} \models_{PL} a$ iff $a \in \mathcal{db}$.[2]

[1]In the scope of this proposition and its proof, we use the notation $\bullet_{db}$ for the operator of Def. 6.4 for clarity of notation.

[2] The interpretation $I^{\mathcal{db}}$ has been defined right before Definition 6.1.

Claim 2 If $I \models_{PL} \mathsf{t}(\mathcal{db})$ then $I = I^{\mathcal{db}}$

The claim also follows right from the definition of the transformation t and the interpretation $I^{\mathcal{db}}$, recalling the finiteness of the alphabet $\mathcal{At}$ in the scope of Proposition 6.1.

Claim 3 $\mathsf{t}(\mathcal{db} \bullet_{db} \{L\}) = \mathsf{t}(\mathcal{db}) \setminus \neg(\{L\}) \cup \{L\}$

First, we observe that the two sets of literals may only differ in the literal L and its complement literal[3]. The observation is immediate from the definition of the transformation function t in Proposition 6.1 and the database update $\bullet_{db}$ of Definition 6.4.

With the latter observation, we must only compare the two sets with respect to the literal L and its complement literal. The literal L is contained in $\mathsf{t}(\mathcal{db}) \setminus \neg(\{L\}) \cup \{L\}$, but not its complement literal which is removed by $\neg(\{L\})$. We show the same property for the set $\mathsf{t}(\mathcal{db} \bullet_{db} \{L\})$, formally

$$a \in \mathsf{t}(\mathcal{db} \bullet_{db} \{a\}) \quad \text{and} \quad \neg a \notin \mathsf{t}(\mathcal{db} \bullet_{db} \{a\}) \quad \text{if } L = a \in \mathcal{At}, \tag{12}$$

$$\neg a \in \mathsf{t}(\mathcal{db} \bullet_{db} \{\neg a\}) \quad \text{and} \quad a \notin \mathsf{t}(\mathcal{db} \bullet_{db} \{\neg a\}) \quad \text{if } L = \neg a, a \in \mathcal{At}. \tag{13}$$

$\boxed{\textbf{Case } 1}$ $L = a \in \mathcal{At}$.
Then, by Definition 6.4 it follows that $a \in \mathcal{db} \bullet_{db} \{a\}$ holds and thus $a \in \mathsf{t}(\mathcal{db} \bullet_{db} \{a\})$ and $\neg a \notin \mathsf{t}(\mathcal{db} \bullet_{db} \{a\})$ follow. This shows (12).

$\boxed{\textbf{Case } 2}$ $L = \neg a, a \in \mathcal{At}$.
Then, by Definition 6.4 it follows that $a \notin \mathcal{db} \bullet_{db} \{\neg a\}$ holds and thus $\neg a \in \mathsf{t}(\mathcal{db} \bullet_{db} \{\neg a\})$ and $a \notin \mathsf{t}(\mathcal{db} \bullet_{db} \{\neg a\})$ follow. This shows (13).

Claim 4 $\mathcal{db} \bullet_{db} \{L\} \models_{PL} \mathcal{IC}$ iff $(\bigwedge(\mathsf{t}(\mathcal{db}) \setminus \neg(\{L\}) \cup \{L\}), \bot) \notin \mathsf{t}(\mathcal{IC})$

For the only-if part we argue as follows. Recall that the notation $\mathcal{db} \models_{PL} F \in \mathcal{L}_{PL}$ abbreviates $I^{\mathcal{db}} \models_{PL} F$ with the interpretation $I^{\mathcal{db}}$ defined

[3] The complement literal of the literal $\neg a, a \in \mathcal{At}$ is a and vice versa.

by $\mathcal{db}$ (cf. Claim 1).

$$\begin{aligned} &\text{By presupposition} && I^{\mathcal{db}\bullet_{db}\{L\}} \models_{PL} \mathcal{IC} && (14)\\ &\text{and by Claim 1} && I^{\mathcal{db}\bullet_{db}\{L\}} \models_{PL} \mathsf{t}(\mathcal{db} \bullet_{db} \{L\}) && \\ &\text{hence, by Claim 3} && I^{\mathcal{db}\bullet_{db}\{L\}} \models_{PL} \mathsf{t}(\mathcal{db}) \setminus \neg(\{L\}) \cup \{L\}. && (15)\end{aligned}$$

Then, by (14) and (15) it follows $\{\bigwedge(\mathsf{t}(\mathcal{db}) \setminus \neg(\{L\}) \cup \{L\})\} \cup \mathcal{IC} \not\vdash_{pl} \bot$ which shows the only-if part by the definition of $\mathsf{t}(\mathcal{IC})$ in Proposition 6.1.

For the if part, the presupposition yields a propositional interpretation I over $\mathcal{At}$ such that

$$\begin{aligned} & && I \models_{PL} \{\bigwedge(\mathsf{t}(\mathcal{db}) \setminus \neg(\{L\}) \cup \{L\})\} \cup \mathcal{IC} && (16)\\ &\text{hence} && I \models_{PL} \mathsf{t}(\mathcal{db}) \setminus \neg(\{L\}) \cup \{L\} && \\ &\text{hence, by Claim 3} && I \models_{PL} \mathcal{db} \bullet_{db} \{L\} && \\ &\text{and finally by Claim 1} && I = I^{\mathcal{db}\bullet_{db}\{L\}}. && \end{aligned}$$

The latter equality together with (16) shows that $\mathcal{db} \bullet_{db} \{L\} \models_{PL} \mathcal{IC}$ holds.

Property 1 of Proposition 6.1

According to Definition 6.7, we must show that $(\bigwedge(\mathsf{t}(\mathcal{db})), \bot) \notin \mathsf{t}(\mathcal{IC})$. This follows right from consistency of the database $\langle \mathcal{IC}, \mathcal{db} \rangle$ by Definition 6.1 by applying Claim 4 and choosing the literal L in that claim such that
$\mathcal{db} \bullet_{db} \{L\} = \mathcal{db}$.

Property 2 of Proposition 6.1

The property states the equality of the two sets of belief in (17) and in (18):

$$\begin{aligned} \mathsf{Bel}^{DB}_{\mathfrak{D}}(\langle \mathcal{IC}, \mathcal{db} \rangle) &= \{B \in \mathcal{L}_{PL} \mid \mathcal{db} \models_{PL} B\}, && (17)\\ \mathsf{Bel}^{NMR}_{\mathfrak{D}}(\mathsf{t}(\langle \mathcal{IC}, \mathcal{db} \rangle)) &= \{B \in \mathcal{L}_{PL} \mid (\bigwedge(\mathsf{t}(\mathcal{db})), B) \in \mathsf{t}(\mathcal{IC})\} && (18)\\ &= \{B \in \mathcal{L}_{PL} \mid \mathsf{t}(\mathcal{db}) \cup \mathcal{IC} \vdash_{pl} B\}. && \end{aligned}$$

Thus, we must show the equivalence $\mathcal{db} \models_{PL} B$ iff $\mathsf{t}(\mathcal{db}) \cup \mathcal{IC} \vdash_{pl} B$ for all $B \in \mathcal{L}_{PL}$.

Considering the only-if part, presuppose that $I^{\mathcal{db}} \models_{PL} B$ holds. Further, consider a propositional interpretation I such that $I \models_{PL} \mathsf{t}(\mathcal{db}) \cup \mathcal{IC}$. By Claim 2 we know that $I = I^{\mathcal{db}}$ and, hence, $I \models_{PL} B$ immediately follows.

Considering the if-part, Claim 2 yields $I^{\mathcal{db}} \models_{PL} \mathtt{t}(\mathcal{db})$. Moreover, the consistency of the database by Definition 6.1 says that $I^{\mathcal{db}} \models_{PL} \mathcal{IC}$ holds. By presupposition of the if-part, altogether it follows $I^{\mathcal{db}} \models_{PL} B$.

Property 3 of Proposition 6.1

To show the equality of the two epistemic states, we consider their components separately.

First, the epistemic state $\mathtt{t}(\langle \mathcal{IC}, \mathcal{db}\rangle \bullet_{db} \{L\})$ has the following two components:

- the consequence relation $\mathtt{t}(\mathcal{IC})$ since by Definition 6.4 the set $\mathcal{IC}$ is not changed by the update $\bullet_{db}$,

- the assertions

$$\mathcal{as}_1 = \begin{cases} \mathtt{t}(\mathcal{db} \bullet_{db} \{L\}) & \text{if } \mathcal{db} \bullet_{db} \{L\} \models_{PL} \mathcal{IC}, \\ \mathtt{t}(\mathcal{db}) & \text{otherwise, by Def. 6.4.} \end{cases}$$

Second, the epistemic state $\mathtt{t}(\langle \mathcal{IC}, \mathcal{db}\rangle) \bullet_{simp} L$ has the following two components:

- the consequence relation $\mathtt{t}(\mathcal{IC})$ since by (6.11) the operator $\bullet_{simp}$ does not change the consequence relation $\mathtt{t}(\mathcal{IC})$,

- the assertions

$$\begin{aligned} \mathcal{as}_2 &= *_{simp}(\mathtt{t}(\mathcal{IC}), L, \mathtt{t}(\mathcal{db})) \\ &= \begin{cases} \mathtt{t}(\mathcal{db}) \setminus \neg(\{L\}) \cup \{L\} & \text{if } (\bigwedge(\mathtt{t}(\mathcal{db}) \setminus \neg(\{L\}) \cup \{L\}), \bot) \notin \mathtt{t}(\mathcal{IC}) \\ \mathtt{t}(\mathcal{db}) & \text{otherwise, by (6.11).} \end{cases} \end{aligned}$$

The sets of assertions $\mathcal{as}_1$ and $\mathcal{as}_2$ are equal, using Claim 4 and then Claim 3. This shows Property 3 of Proposition 6.1.

□(proof Prop. 6.1)

Lemma 7.1

Let $\mathcal{Seq} = \langle \mathcal{U}_1, \ldots, \mathcal{U}_{k-1} \rangle$ be a sequence of sets of literals with each set $\mathcal{U}_i$ being defined over distinct atoms. Further, let $\langle \mathcal{db}_1, \ldots, \mathcal{db}_k \rangle$ be a sequence of database instances such that for all $i = 1, \ldots, k-1$ it holds

$$\mathcal{db}_i \not\models_{PL} L \text{ for all } L \in \mathcal{U}_i \qquad \text{and} \qquad \mathcal{db}_{i+1} = \mathcal{db}_i \bullet \mathcal{U}_i.$$

Then, for all formulas $A \in \mathcal{L}_{PL}$ it holds for all $i = 1, \ldots, k$

$$\mathcal{db}_i \models_{PL} A \qquad \text{iff} \qquad \mathcal{db}_k \models_{PL} \mathsf{neg}(A, \Delta\mathcal{Up}[\mathcal{Seq}][i]).$$

Proof We prove this lemma by induction on k.
Base case: $k = 1$

Nothing is to show.
Inductive case: $k - 1 \to k$

We apply the induction hypothesis unto the sequence $\langle \mathcal{db}_2, \ldots, \mathcal{db}_k \rangle$ of database instances with updates $\mathcal{U}_2$ to $\mathcal{U}_{k-1}$ defining $\Delta\mathcal{Up}[\langle \mathcal{U}_2, \ldots, \mathcal{U}_{k-1} \rangle]$ by Definition 7.2. This definition immediately implies that $\Delta\mathcal{Up}[\langle \mathcal{U}_2, \ldots, \mathcal{U}_{k-1} \rangle] = \langle \Delta\mathcal{Up}[\mathcal{Seq}][2], \ldots, \Delta\mathcal{Up}[\mathcal{Seq}][k] \rangle$ holds. Hence, we obtain that for all formulas A and for all $i = 2, \ldots, k$ it holds

$$\mathcal{db}_i \models_{PL} A \quad \text{iff} \quad \mathcal{db}_k \models_{PL} \mathsf{neg}(A, \Delta\mathcal{Up}[\mathcal{Seq}][i]). \tag{19}$$

By presupposition we generated $\mathcal{db}_2$ by $\mathcal{db}_2 = \mathcal{db}_1 \bullet \mathcal{U}_1$ where $\mathcal{db}_1 \not\models_{PL} L$ for all $L \in \mathcal{U}_1$. Knowing this, we argue that for all formulas A it holds

$$\begin{array}{lll}
 & \mathcal{db}_1 \models_{PL} A & \\
\text{iff} & \mathcal{db}_2 \models_{PL} \mathsf{neg}(A, \mathcal{U}_1) & \text{by Lem. 6.3} \\
\text{iff} & \mathcal{db}_k \models_{PL} \mathsf{neg}(\mathsf{neg}(A, \mathcal{U}_1), \Delta\mathcal{Up}[\mathcal{Seq}][2]) & \text{by (19)} \\
\text{iff} & \mathcal{db}_k \models_{PL} \mathsf{neg}(\mathsf{neg}(A, \mathsf{At}(\mathcal{U}_1)), \Delta\mathcal{Up}[\mathcal{Seq}][2]) & \text{by Lem. 6.2} \\
\text{iff} & \mathcal{db}_k \models_{PL} \mathsf{neg}(A, (\Delta\mathcal{Up}[\mathcal{Seq}][2] \setminus \mathsf{At}(\mathcal{U}_1)) \cup (\mathsf{At}(\mathcal{U}_1) \setminus \Delta\mathcal{Up}[\mathcal{Seq}][2])) & \text{by Lem. 6.2} \\
\text{iff} & \mathcal{db}_k \models_{PL} \mathsf{neg}(A, \Delta\mathcal{Up}[\mathcal{Seq}][1]) & \text{by Def. 7.2.}
\end{array}$$

□(proof Lem. 7.1)

Proposition 7.1

Let cexec *be a control function of Def. 4.3 with domain* $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}$*. Further, let* $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ *be the system of defender* $\mathcal{D}$ *and attacker* $\mathcal{A}$ *defined by* cexec*. Then, for all points* $(r, m) \in \mathcal{PT}(\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}})$ *and for all* $S \in \mathcal{L}_{PL}$

$$\begin{array}{ll} \mathcal{db}(r,m) \not\models_{PL} \mathsf{ccp}(S, \Delta\mathcal{Up}[r,m]) & \textit{of Def. 7.3 holds} \\ \qquad\textit{iff} & \\ \qquad r \in R^{CCP}_{S,m} & \textit{of (4.2) holds.} \end{array}$$

Proof We rewrite the definition of $R^{CCP}_{S,m}$ in (4.2) for control functions with domain $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}$ where $S \in \mathcal{L}_{PL}$ and $m \in \mathbb{N}_0$:

$$R^{CCP}_{S,m} = \{r' \in \mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}} \mid \text{for all } n \leq m : \quad S \notin \mathsf{Bel}^{DB}_{\mathcal{D}}(\mathcal{IC}(r', n), \mathcal{db}(r', n))\}. \tag{20}$$

Next, writing u_i, we refer to the time before the request with the i-th effective update $\mathcal{U}_i$ in run r before time m, that is the $u_i + 1$-th request $\mathsf{up}(\mathcal{L})$ such that $\mathcal{U}_i \subseteq \mathcal{L}$ and

$$\mathcal{db}(r, u_i) \not\models_{PL} L \text{ for all } L \in \mathcal{U}_i \quad \text{ and } \quad \mathcal{db}(r, u_i + 1) = \mathcal{db}(r, u_i) \bullet \mathcal{U}_i. \tag{21}$$

We set $u_k = u_{k-1} + 1$, that is the time of the request of the last effective update. Moreover, for all times $j \leq m$ we have that

$$\mathcal{db}(r, j) = \mathcal{db}(r, u_i) \qquad\qquad \text{for some } i \in \{1, \ldots k\}. \tag{22}$$

and, in particular, $\mathcal{db}(r, m) = \mathcal{db}(r, u_k)$.

Only-If Part

We presuppose that $\mathcal{db}(r, u_k) = \mathcal{db}(r, m) \not\models_{PL} \mathsf{ccp}(S, \Delta\mathcal{Up}[r, m])$ holds which implies

$$\mathcal{db}(r, u_k) \models_{PL} \neg\mathsf{neg}(S, \Delta\mathcal{Up}[r, m][1]) \land \ldots \land \neg\mathsf{neg}(S, \Delta\mathcal{Up}[r, m][k]) \tag{23}$$
$$= \mathsf{neg}(\neg S, \Delta\mathcal{Up}[r, m][1]) \land \ldots \land \mathsf{neg}(\neg S, \Delta\mathcal{Up}[r, m][k]) \qquad \text{by Def. 6.5.}$$

Finally, we use Lemma 7.1 with (21) and (23) and, this way, obtain $\mathcal{db}(r, u_i) \models_{PL} \neg S$ for $i = 1, \ldots, k$. Thus, we conclude together with (22) that $r \in R^{CCP}_{S,m}$ of (20) holds.

If-Part

Now, we presuppose that $r \in R^{CCP}_{S,m}$ of (20) holds which implies

$$\mathcal{db}(r, u_i) \models_{PL} \neg S \qquad \text{for all } i = 1, \ldots, k$$

and hence with Lemma 7.1

$$\mathcal{db}(r, u_k) \models_{PL} \mathsf{neg}(\neg S, , \Delta\mathcal{Up}[r, m][i]) \qquad \text{for all } i = 1, \ldots, k.$$

As we have seen in the only-if part, this means that
$\mathcal{db}(r, m) \not\models_{PL} \mathsf{ccp}(S, \Delta\mathcal{Up}[r, m])$.

□(proof Prop. 7.1)

Proposition 7.3

Procedure 2 outputs a state of Def. 7.1.

Proof We have to show that the output $\langle \mathcal{IC}, \mathcal{db}_t \rangle$ is database according to Definition 6.1 and that the output $\mathcal{V}_t$ is a database view according to Definition 6.2.
First, the integrity constraints are ensured by Procedure 2 in line 26 before the database update in line 34. There is no other way the database instance may be updated by the procedure. Therefore, $\langle \mathcal{IC}, \mathcal{db}_t \rangle$ is a database.
Finally, we argue that $\mathcal{C}_t$ is an instance view of database instance $\mathcal{db}_t$ and to this aim distinguish the following cases.

Case 1 The request triggered a successful update $\mathcal{U} \subseteq \mathcal{L}_{PL}$ with $\mathcal{db}_{t-1} \not\models_{PL} L$ for all $L \in \mathcal{U}$.
Procedure 2 updates the database instance $\mathcal{db}_{t-1}$ to $\mathcal{db}_t = \mathcal{db}_{t-1} \bullet \mathcal{U}$.

Because $\mathcal{V}_{t-1}$ is a database view of the database $\langle \mathcal{IC}, \mathcal{db}_{t-1} \rangle$ by Definition 7.1, we obtain

$$\mathcal{db}_{t-1} \models_{PL} \mathsf{skeptical}(\mathcal{V}_{t-1}) \vdash_{pl} \mathcal{C}_{t-1}.$$

By Lemma 6.3 the satisfaction relation $\mathcal{db}_{t-1} \models_{PL} \mathcal{C}_{t-1}$ is equivalent to

$$\mathcal{db}_t \models_{PL} \mathcal{C}_{t-1} \odot \mathcal{U} = \mathcal{C}_t.$$

Case 2 There has not been a successful update.

Then $\mathcal{db}_t = \mathcal{db}_{t-1}$ holds. Like in the previous case, by Definition 7.1 we know that $\mathcal{db}_{t-1} \models_{PL} \mathsf{skeptical}(\mathcal{V}_{t-1}) \vdash_{pl} \mathcal{C}_{t-1}$ holds. Let $\mathcal{U}$ denote the outstanding updates of the request. Procedure 2 may only add the formula $\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})$ to the instance view $\mathcal{C}_{t-1}$ in line 28. Before that, in line 26 the procedure has checked that $\mathcal{db}_{t-1} \bullet \mathcal{U} \models_{PL} \neg \bigwedge(\mathcal{IC})$ holds and, hence, we conclude with Lemma 6.3 that $\mathcal{db}_t = \mathcal{db}_{t-1} \models_{PL} \mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})$ holds. The lemma may be applied because $\mathcal{U}$ contains the outstanding updates so that $\mathcal{db}_{t-1} \not\models_{PL} L$ for all $L \in \mathcal{U}$. Altogether, it follows $\mathcal{db}_t \models_{PL} \mathcal{C}_t$.

□(proof Prop. 7.3)

Lemma 7.2

Proc. 1 and Proc. 2 maintain the following two invariants:

$$\begin{aligned} \mathsf{ccp}(S, \Delta\mathcal{Up}[s]) &\notin \mathsf{skeptical}(\mathcal{V}) && \textit{if } S \in \textit{conf}(CCP), \textit{ and} \\ S &\notin \mathsf{skeptical}(\mathcal{V}) && \textit{if } S \in \textit{conf}(TCP). \end{aligned}$$

Proof

Case 1 The request θ is $\mathtt{que}(A)$.

Then, Procedure 1 explicitly ensures the invariants in line 5.

Case 2 The request θ is $\mathtt{up}(\mathcal{L})$.

Case 2–a Procedure 2 terminates until line 25.

Then the procedure only updates the view when invoking Procedure 1 and we can use the same argument as in Case 1.

Case 2–b Procedure 2 terminates in line 30.

Then the invariants are enforced via the conditions checked in line 22.

Case 2–c Procedure 2 terminates after line 30.

Then the invariants are enforced via the conditions checked in line 12.

□(proof Lem. 7.2)

Proposition 7.4

Let cexec *be a control function of Def. 4.3 with domain* $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}$ *under Assumption 7. Further, let* $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ *be the system of defender* $\mathcal{D}$ *and attacker* $\mathcal{A}$ *defined by* cexec*. If* cexec *satisfies* ***(Soundness of Attacker View)*** *of Def. 7.5 and* ***(No Loss of Information)*** *of Def. 7.6, then* cexec *correctly simulates operator* $\mathcal{K}_{\mathcal{A}}$ *in the sense of Def. 7.4.*

Proof Let $(r, m) \in \mathcal{PT}(\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}})$ be a point of the system of defender $\mathcal{D}$ and attacker $\mathcal{A}$ defined by the control function cexec. We will show the equivalence required for the correctness of simulation according to Definition 7.4.

If-Part of Correctness Requirement

The proof of the if part does not need that control function satisfies **(Soundness of Attacker View)** and **(No Loss of Information)**. According to Definition 7.4 of the correctness of simulation, we presuppose that there exists run $r' \in \mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ with $(r', m) \in \mathcal{K}_{\mathcal{A}}(r, m)$ and denote $\mathcal{db} = \mathcal{db}(r', m)$. Then, we show that $\mathcal{db} \models_{PL} \mathsf{skeptical}(\mathcal{V}(r, m))$ follows. Recall the definition of the operator $\mathcal{K}_{\mathcal{A}}$ in (2.3) which models the attacking agent $\mathcal{A}$'s reasoning about system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of Definition 4.5. By those definitions, the presupposition implies that $\mathcal{V}(r', m) = \mathcal{V}(r, m)$ holds. We write $\mathcal{V}(r', m) = \mathcal{V}(r, m) = \langle \mathcal{IC}, \mathcal{C} \rangle$. By Definition 7.1, the pair $\langle \mathcal{IC}, \mathcal{C} \rangle$ is a database view of the database $\langle \mathcal{IC}, \mathcal{db} \rangle$ in point (r', m). In particular, this implies that $\mathcal{db}(r', m) = \mathcal{db} \models_{PL} \mathcal{C} \cup \mathcal{IC} \vdash_{pl} \mathsf{skeptical}(\mathcal{V}(r, m))$ holds.

Only-If-Part of Correctness Requirement

Consider a run r at time m in system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$. We show the only-if part by induction on time m.

Base case: $m = 0$

According to Definition 7.4 of the correctness of simulation, we presuppose a database instance $\mathcal{db}$ satisfying $\mathcal{db} \models_{PL} \mathsf{skeptical}(\mathcal{V}(r, 0))$. Then, we show that there exists a run $r' \in \mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ such that $\mathcal{db} = \mathcal{db}(r', 0)$ and $(r', 0) \in \mathcal{K}_{\mathcal{A}}(r, 0)$.

The presupposition implies that the state

$$st_{\mathfrak{D}} = \langle \mathcal{IC}(r,0),\ \mathcal{db},\ \mathit{conf}(r,0),\ \langle\rangle,\ \mathcal{V}(r,0)\rangle$$

is a state of Definition 7.1 satisfying `pre` of (7.1). Thus, we may define the required run r' starting in state $st_{\mathfrak{D}}$ with any sequence of requests following Definition 4.5 of system $\mathcal{R}^{\mathsf{cexec}}_{\mathfrak{D},\mathcal{A}}$.

Inductive case: $m \to m+1$

Since the considered control function has domain $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}$ of Definition 7.1, we may write

$$\mathcal{db}(r,m+1) = \mathcal{db}(r,m) \bullet \mathcal{U} \quad \text{and} \quad \mathcal{db}(r,m) \quad \not\models_{PL} L \quad \text{for all } L \in \mathcal{U} \tag{24}$$

where $\mathcal{U}$ might be empty.

First, showing the only-if part, we presuppose a database instance $\mathcal{db}$ that satisfies $\mathcal{db} \models_{PL} \mathsf{skeptical}(\mathcal{V}(r,m+1))$ and then by **(No Loss of Information)** of Definition 7.6 we argue as follows

$$\begin{aligned} \mathcal{db} &\models_{PL} \mathsf{skeptical}(\mathcal{V}(r,m+1)) \\ &\supseteq \mathsf{skeptical}(\mathcal{V}(r,m) \odot \mathcal{U}) && \text{by Def. 7.6} \\ &\supseteq \begin{cases} \mathcal{U} & \text{by Def. 6.6,} \\ (\mathcal{C}(r,m) \cup \mathcal{IC}) \odot \mathcal{U} & \text{by Def. 6.6.} \end{cases} \end{aligned} \tag{25}$$

For constructing the required run r', we will apply the induction hypothesis on the database instance $\mathcal{db} \bullet \neg(\mathcal{U})$ reversing the effective updates $\mathcal{U}$.[4] To this end, we show that this instance is a model of $\mathsf{skeptical}(\mathcal{V}(r,m))$ in the following.

First, from (25) it follows that $\mathcal{db} \not\models_{PL} L$ for all $L \in \neg(\mathcal{U})$ so that we may apply Lemma 6.1 and, hence, obtain

$$\mathcal{db} = (\mathcal{db} \bullet \neg(\mathcal{U})) \bullet \mathcal{U}. \tag{26}$$

With this equation, we rewrite parts of (25) to

$$(\mathcal{db} \bullet \neg(\mathcal{U})) \bullet \mathcal{U} \models_{PL} (\mathcal{C}(r,m) \cup \mathcal{IC}) \odot \mathcal{U}.$$

[4] Note that, if $\mathcal{U}$ is empty, then the considerations, which prepare the application of the induction hypothesis in the following, are trivial.

Applying Lemma 6.3 on the latter result, it follows

$$\mathcal{db} \bullet \neg(\mathcal{U}) \models_{PL} \mathcal{C}(r,m) \cup \mathcal{IC} \vdash_{pl} \mathsf{skeptical}(\mathcal{V}(r,m)). \tag{27}$$

After these considerations, by (27) we may apply the induction hypothesis on $\mathcal{db} \bullet \neg(\mathcal{U})$. The hypothesis yields that there exists a run r' such that

$$\mathcal{db} \bullet \neg(\mathcal{U}) = \mathcal{db}(r',m) \qquad \text{and} \qquad (r',m) \in \mathcal{K}_{\mathcal{A}}(r,m). \tag{28}$$

Thus, (r',m) is a possible point at point (r,m) as defined by (2.3) so that that the following two local states of $\mathcal{D}$ fulfill the requirements of Definition 7.5 of **(Soundness of Attacker View)**

$$\begin{aligned} r_{\mathcal{D}}(m) &= \langle \quad \mathcal{IC}(r,m), \quad \mathcal{db}(r,m), \quad \mathit{conf}(r,m), \quad \mathcal{H}^{DB}(r,m), \quad \mathcal{V}(r,m)\rangle \\ r'_{\mathcal{D}}(m) &= \langle \quad \mathcal{IC}(r,m), \quad \mathcal{db} \bullet \neg(\mathcal{U}), \quad \mathit{conf}(r,m), \quad \mathcal{H}^{DB}(r,m), \quad \mathcal{V}(r,m)\rangle. \end{aligned}$$

Note that the sets of integrity constraints are the same in the points (r,m) and (r',m) because by Definition 7.1 each set is contained in the respective attacker view. Let us assume that runs r and r' process the same request at time $m+1$.[5] Proposition 7.4 to be proven presupposes that the control function maintains a sound attacker view in the sense of Definition 7.5. Hence, we use the contraposition of that property saying here that for all sets $\mathcal{U}$ of literals over pairwise distinct atoms it holds,

$$\begin{array}{lcr} \text{if} & \mathcal{db}(r,m+1) = \mathcal{db}(r,m) \bullet \mathcal{U} & \text{implies} \\ & \mathcal{db}(r',m) \bullet \mathcal{U} \models_{PL} \mathsf{skeptical}(\mathcal{V}(r,m+1)), & \\ \text{then} & \mathcal{H}^{DB}(r',m+1) = \mathcal{H}^{DB}(r,m+1) & \text{and} \\ & \mathcal{V}(r',m+1) = \mathcal{V}(r,m+1). & \end{array}$$

The implication in the premise holds by (28), (26) and (25) for the set $\mathcal{U}$ being the effective updates in run r at time $m+1$ according to (24). Altogether, it follows that $\mathcal{H}^{DB}(r',m+1) = \mathcal{H}^{DB}(r,m+1)$ and $\mathcal{V}(r',m+1) = \mathcal{V}(r,m+1)$. This shows $(r',m+1) \in \mathcal{K}_{\mathcal{A}}(r,m+1)$ by (2.3) and Definition 4.5 of system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ and Definition 7.1 of the states in the implemented system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$.

[5] Otherwise, we may redefine r' such that the $(m+1)$-th request in run r' is the request in the history of run r at time $m+1$ without loosing the validity of (28).

Finally, since the history $\mathcal{H}^{DB}(r', m+1)$ records the effective updates on database instance $db(r', m) = db \bullet \neg(\mathcal{U})$ by Assumption 7, which are $\mathcal{U}$ by $\mathcal{H}^{DB}(r', m+1) = \mathcal{H}^{DB}(r, m+1)$, we have $db(r', m+1) = db(r', m) \bullet \mathcal{U} = db$ by (26).

□(proof Prop. 7.4)

Lemma 7.3

The control functions Proc. 1 and Proc. 2 maintain a sound attacker view.

Proof We will prove the lemma by contraposition in the following cases.

$\boxed{\textbf{Case } 1}$ The request θ is $\mathsf{que}(A)$.
In this case, there is no update so that $db_t = db_{t-1}$ holds. Hence, for proving the contraposition of the lemma, we presuppose

$$db'_{t-1} \models_{PL} \mathsf{skeptical}(\mathcal{V}_t) \supseteq \mathcal{C}_t. \tag{29}$$

Next, we consider the execution of Procedure 1 on $\mathcal{D}$'s state $s_{\mathcal{D}}$ with database instance db_{t-1} after the request $\mathsf{que}(A)$, comparing it with the execution on state $s_{\mathcal{D}}'$ with database instance db'_{t-1}. The evaluation of all conditions until line 6 does not depend on the database instance. If the procedure terminates after line 6, then the notification *note* is the evaluation of the query in db_{t-1} and included into the instance view $\mathcal{C}_t$ in line 9 or line 13 respectively. Hence, by (29) the query results in the same evaluation on database instance db'_{t-1}. Altogether, the procedure outputs the same history and view when executed on state $s'_{\mathcal{D}}$.

$\boxed{\textbf{Case } 2}$ The request θ is $\mathsf{up}(\mathcal{L})$ but there is no successful update on $s_{\mathcal{D}}$ and request θ.
In this case, $db_t = db_{t-1}$ holds so that, for proving the contraposition of the lemma, we presuppose

$$db'_{t-1} \models_{PL} \mathsf{skeptical}(\mathcal{V}_t) \supseteq \mathcal{C}_t. \tag{30}$$

Next, we compare the execution of Procedure 2 on both states. First, when executed on $s_{\mathcal{D}}$, the procedure computes the set $\mathcal{U} \subseteq \mathcal{L}$. This computation uses Procedure 1 so that we may reuse the arguments of Case 1 to show that the histories and views are equal until line 10 of Procedure 2 and that the same set $\mathcal{U} \subseteq \mathcal{L}$ is computed on both input states.

After line 10 the procedure does not use the database instance until line 26. As there is no successful update on $s_{\mathfrak{D}}$ and θ, it must hold that $\mathcal{IC} \not\subseteq \mathsf{Bel}_{\mathfrak{D}}^{DB}(\langle \mathcal{IC}, \mathcal{db}_{t-1} \bullet \mathcal{U}\rangle)$. Thus, the procedure in line 28 ensures that $\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U}) \in \mathcal{C}_t$ holds. Together with (30), we obtain $\mathcal{db}'_{t-1} \models_{PL} \mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})$. We apply Lemma 6.3 on the latter result. The application of the lemma is possible because it holds $\mathcal{db}'_{t-1} \not\models L$ for all $L \in \mathcal{U}$ by (30) since $\neg(\mathcal{U}) \subseteq \mathcal{C}_t$ is ensured by the procedure. The lemma yields $\mathcal{db}'_{t-1} \bullet \mathcal{U} \models_{PL} \neg \bigwedge(\mathcal{IC})$ which implies $\mathcal{IC} \not\subseteq \mathsf{Bel}_{\mathfrak{D}}^{DB}(\langle \mathcal{IC}, \mathcal{db}'_{t-1} \bullet \mathcal{U}\rangle)$. The rest of the computation does not depend on the database instance so that finally we have $\mathcal{H}_t^{DB'} = \mathcal{H}_t^{DB}$ and $\mathcal{V}'_t = \mathcal{V}_t$.

Case 3 The request θ is $\mathsf{up}(\mathcal{L})$ and there is a successful update $\mathcal{U} \subseteq \mathcal{L}$ on $s_{\mathfrak{D}}$ and request θ.
In this case, we know that $\mathcal{db}_t = \mathcal{db}_{t-1} \bullet \mathcal{U}$ holds. For showing the contraposition, we thus presuppose

$$\mathcal{db}'_{t-1} \bullet \mathcal{U} \models_{PL} \mathsf{skeptical}(\mathcal{V}_t). \tag{31}$$

We argue like in Case 2 until line 26. As there is a successful update on $s_{\mathfrak{D}}$ and θ, it must hold that $\mathcal{IC} \subseteq \mathsf{Bel}_{\mathfrak{D}}^{DB}(\langle \mathcal{IC}, \mathcal{db}_{t-1} \bullet \mathcal{U}\rangle)$. By (31), we further know that $\mathcal{IC} \subseteq \mathsf{Bel}_{\mathfrak{D}}^{DB}(\langle \mathcal{IC}, \mathcal{db}'_{t-1} \bullet \mathcal{U}\rangle)$ because $\mathsf{skeptical}(\mathcal{V}_t)$ contains $\mathcal{IC}$. Thus, we conclude like in Case 2.

□(proof Lem. 7.3)

Theorem 7.1

The control functions Proc. 1 and Proc. 2 preserve continuous/temporary confidentiality towards agent $\mathcal{A}$ in the sense of Def. 4.6.

Proof The proof uses Theorem 4.1: Confidentiality policies are translated to a single possibility policy in system $\mathcal{R}_{\mathfrak{D},\mathcal{A}}^{\mathsf{cexec}}$ and temporary/continuous confidentiality is enforced through policy-based secrecy of Definition 3.3. The translation is found in Definition 4.7.

Let thus (r, m) be a point in system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ of Definition 4.5 and $I \in \mathtt{policy}_{\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}}(r, m)$ a piece of information for the protection of some confidential belief $S \in \mathit{conf}(r, m)$. Policy-based secrecy requires

$$\mathcal{K}_{\mathcal{A}}(r, m) \cap I \neq \emptyset. \tag{32}$$

We distinguish between the cases whether S is a temporary or a continuous confidential belief.

$\boxed{\textbf{Case } 1}$ $S \in \mathit{conf}(TCP)(r, m)$.
In this case, by Definition 4.7 the set I is the set of points

$$I^{TCP}_S = \{(r', m') \mid S \notin \mathsf{Bel}^{DB}_{\mathcal{D}}(\mathcal{IC}(r', m'), \mathcal{db}(r', m'))\}. \tag{33}$$

The invariant of Lemma 7.2 ensures that $S \notin \mathsf{skeptical}(\mathcal{V}(r, m))$ holds where we need that $\mathit{conf}(TCP)(r, m) = \mathit{conf}(TCP)(r, 0)$ holds by Assumption 5. Let $\mathcal{db}$ be a witness of the latter fact; formally, $\mathcal{db}$ satisfies

$$\mathcal{db} \models_{PL} \mathsf{skeptical}(\mathcal{V}(r, m)) \qquad\qquad \mathcal{db} \models_{PL} \neg S. \tag{34}$$

By Lemma 7.5, Procedure 1 and Procedure 2 correctly simulate operator $\mathcal{K}_{\mathcal{A}}$ by means of the view $\mathcal{V}$ in the sense of Def. 7.4. Hence, there exists a run $r' \in \mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ such that

$$\mathcal{db} = \mathcal{db}(r', m) \qquad \text{and} \qquad (r', m) \in \mathcal{K}_{\mathcal{A}}(r, m). \tag{35}$$

Recalling (33), we further observe that $(r', m) \in I^{TCP}_S$ because by (35) it holds $\mathcal{db}(r', m) = \mathcal{db}$ and by (34) it holds $\mathcal{db} \models_{PL} \neg S$. Together with (35), this shows (32).

$\boxed{\textbf{Case } 2}$ $S \in \mathit{conf}(CCP)(r, m)$.
The line of argumentation is similar to that of Case 1. First, in this case, by Definition 4.7 the set I is the set of points

$$\mathcal{PT}(R^{CCP}_{S,m}) \text{ with the property } R^{CCP}_{S,m} \subseteq \mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}} \text{ of runs of (4.2).} \tag{36}$$

The invariant of Lemma 7.2 ensures that
$\mathsf{ccp}(S, \Delta\mathcal{Up}[r, m]) \notin \mathsf{skeptical}(\mathcal{V}(r, m))$ holds where we need that

$conf(TCP)(r,m) = conf(TCP)(r,0)$ holds by Assumption 5. Let $\mathcal{db}$ be a witness of the negative set membership; formally, $\mathcal{db}$ satisfies

$$\mathcal{db} \models_{PL} \mathsf{skeptical}(\mathcal{V}(r,m)) \qquad \mathcal{db} \not\models_{PL} \mathsf{ccp}(S, \Delta\mathcal{Up}[r,m]). \tag{37}$$

Like in Case 1, the correctness of the simulation of operator $\mathcal{K}_{\mathcal{A}}$ of Lemma 7.5 with (37) guarantees that there exists a run $r' \in \mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$ satisfying (35). In particular, if we recall the definition of the operator $\mathcal{K}_{\mathcal{A}}$ in (2.3) in the context of system $\mathcal{R}^{\mathsf{cexec}}_{\mathcal{D},\mathcal{A}}$, we immediately see that run r' outputs the same history as run r until time m. Hence, it holds $\Delta\mathcal{Up}[r',m] = \Delta\mathcal{Up}[r,m]$ since those sequences are obtained from the histories in the runs until time m. With these considerations, we apply Lemma 7.1 to run r' at time m, using (37) and $\mathcal{db}(r',m) = \mathcal{db}$ of (35), and obtain $r' \in R^{CCP}_{S,m}$ which together with (35) proves (32).

$\square$(proof Thm. 7.1)

Lemma 7.6

Let cexec *be a control function of Def. 4.3 with domain* $\mathcal{St}^{DB} \times \mathcal{Req}^{DB}_{\mathrm{up}}$ *fulfilling Asmp. 7. Let the view* $\mathcal{V}_t$ *in the output state* s_t *contain confidential information, that is there exists* $S \in \mathcal{L}_{PL}$ *such that*

$$\begin{array}{llll} & (S \in \mathsf{skeptical}(\mathcal{V}_t) & \textit{and} & S \in conf(TCP)) \\ \textit{or} & (\mathsf{ccp}(S, \Delta\mathcal{Up}[s_t]) \in \mathsf{skeptical}(\mathcal{V}_t) & \textit{and} & S \in conf(CCP)). \end{array}$$

Then, cexec *violates the confidentiality requirements of Def. 4.6.*

Proof We prove this lemma by contraposition. Assume that cexec preserves confidentiality according to Definition 4.6 so that after the t-th request in the sequence $\mathcal{Int}$ with initial state $s_{\mathcal{D}}$ there exists an alternative initial state $s'_{\mathcal{D}} \in \mathcal{St}^{DB}$ with the properties required by the definition. Since this state is indistinguishable from the actual state $s_{\mathcal{D}}$, after the t-th request the views $\mathcal{V}'_t$ and $\mathcal{V}_t$ are equal and so are the histories $\mathcal{H}^{DB'}_t$ and $\mathcal{H}^{DB}_t$. Further, by Definition 7.1 cexec ensures that $\mathcal{V}'_t$ is a database view of the database $\langle \mathcal{IC}, \mathcal{db}'_t \rangle$ so that it holds

$$\mathcal{db}'_t \models_{PL} \mathsf{skeptical}(\mathcal{V}'_t) = \mathsf{skeptical}(\mathcal{V}_t) \tag{38}$$

$\boxed{\textbf{Case } 1}$ $S \in \mathit{conf}(TCP)$.
In the alternative situation $\mathcal{D}$'s current belief is safe, that is $S \notin \mathsf{Bel}_{\mathcal{D}}^{DB}(\mathcal{IC}, \mathcal{db}'_t)$. This means by definition that $\mathcal{db}'_t \models_{PL} \neg S$ holds. Together with (38) it follows that $\mathsf{skeptical}(\mathcal{V}_t) \not\models_{pl} S$ holds so that $S \notin \mathsf{skeptical}(\mathcal{V}_t)$ also holds.

$\boxed{\textbf{Case } 2}$ $S \in \mathit{conf}(CCP)$.
In the alternative situation $\mathcal{D}$'s belief throughout the interaction is safe, that is for all $0 \leq j \leq t$ it holds $S \notin \mathsf{Bel}_{\mathcal{D}}^{DB}(\mathcal{IC}, \mathcal{db}'_j)$. Thus, the run $r^{\mathit{Int}, s'_{\mathcal{D}}}$ is an alternative run for the protection of S. Formally, we obtain $r^{\mathit{Int}, s'_{\mathcal{D}}} \in R_{S,t}^{CCP}$ by (4.2). Proposition 7.1 now implies that $\mathcal{db}'_t \not\models_{PL} \mathsf{ccp}(S, \Delta\mathcal{Up}[s'_t])$ where $\mathcal{A}$ may determine the formula $\mathsf{ccp}(S, \Delta\mathcal{Up}[s'_t])$ on the basis of Assumption 7. Hence, together with (38) it follows $\mathsf{ccp}(S, \Delta\mathcal{Up}[s'_t]) \notin \mathsf{skeptical}(\mathcal{V}_t)$. Moreover, we know that $\Delta\mathcal{Up}[s'_t] = \Delta\mathcal{Up}[s_t]$ since the histories after the t-th request are equal.

$\Box$(proof Lem. 7.6)

Theorem 7.2

Procedure 2 for controlled view update execution is locally optimal.

Proof We proceed with a proof by cases, under which condition Procedure 2 terminates. In Section 7.5, we assume that the procedure does not terminate in Case 1, but that $\mathcal{A}$ knows the set $\mathcal{U}$ of outstanding updates without a confidentiality violation, cf. (7.6).

$\boxed{\textbf{Case } 1}$ Procedure 2 terminates in Case 2.
In this case, Procedure 2 outputs the view $\mathcal{V}_t = \mathcal{V}_{t-1}$ and $\Delta^{\mathsf{cexec}} = \emptyset$ holds.

$\boxed{\textbf{Case } 1\text{–a}}$ cexec' successfully updates $\mathcal{db}_{t-1}$ by $\mathcal{U}$.
The control function cexec' satisfies **(No Loss of Information)** of Definition 7.6 and hence ensures

$$\mathsf{skeptical}(\mathcal{V}_{t-1} \odot \mathcal{U}) \subseteq \mathsf{skeptical}(\mathcal{V}_t^{\mathsf{cexec}'}).$$

But Case 2 of Procedure 2 detected that the updated view $\mathcal{V}_{t-1} \odot \mathcal{U}$ contains confidential information and hence the view $\mathcal{V}_t^{\mathsf{cexec}'}$. Lemma 7.6 shows that cexec' violates the confidentiality requirements of Definition 4.6.

Case 1–b cexec' does not update db_{t-1}.
Then by **(Cooperativeness)** it follows
$\mathsf{skeptical}(\mathcal{V}_t^{\mathsf{cexec}'}) \subseteq \mathsf{skeptical}(\mathcal{V}_{t-1}) = \mathsf{skeptical}(\mathcal{V}_t^{\mathsf{cexec}})$ because the condition in line 12 of Procedure 2 holds for the input in the considered case of termination of Procedure 2.

Case 2 Procedure 2 terminates in Case 4 after the if-statement in line 17 of Case 3 did not apply.
In this case, control function cexec updates the database instance so that $\Delta^{\mathsf{cexec}'} \subseteq \mathcal{U} = \Delta^{\mathsf{cexec}}$. If equality holds, then by **(Cooperativeness)** of cexec' we obtain the following, knowing that cexec computes $\mathcal{V}_t^{\mathsf{cexec}} = \mathcal{V}_{t-1} \odot \mathcal{U}$:

$$\mathsf{skeptical}(\mathcal{V}_t^{\mathsf{cexec}'}) \subseteq \mathsf{skeptical}(\mathcal{V}_{t-1} \odot \mathcal{U}) = \mathsf{skeptical}(\mathcal{V}_t^{\mathsf{cexec}}).$$

Case 3 Procedure 2 terminates after the if-statement in line 18 applied.
Then cexec does not update the database instance so that $\Delta^{\mathsf{cexec}} = \emptyset$. The condition of the if-statement in line 18 implies that
$\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U}) \in \mathsf{skeptical}(\mathcal{V}_{t-1})$ is true. By Definition 7.1, the input to cexec' satisfies $db_{t-1} \models_{PL} \mathsf{skeptical}(\mathcal{V}_{t-1})$ because $\mathcal{V}_{t-1}$ is a database view of the database $\langle \mathcal{IC}, db_{t-1} \rangle$. From this it follows that
$db_{t-1} \models_{PL} \mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})$ holds. Hence, by Lemma 6.3 we obtain that $db_{t-1} \bullet \mathcal{U} \models_{PL} \neg \bigwedge(\mathcal{IC})$ holds. Since cexec' preserves atomicity and consistency, we know that $\Delta^{\mathsf{cexec}'} = \emptyset$ must hold.

Finally, we have to compare the computed views. Procedure cexec computes $\mathcal{V}_t^{\mathsf{cexec}} = \mathcal{V}_{t-1}$ and, further, **(Cooperativeness)** of cexec' ensures

$$\begin{aligned}\mathsf{skeptical}(\mathcal{V}_t^{\mathsf{cexec}'}) &\subseteq \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}_{t-1} \cup \{\mathsf{neg}(\neg \textstyle\bigwedge(\mathcal{IC}), \mathcal{U})\}\rangle) = \mathsf{skeptical}(\mathcal{V}_{t-1}) \\ &= \mathsf{skeptical}(\mathcal{V}_t^{\mathsf{cexec}}).\end{aligned}$$

Case 4 Procedure 2 terminates after the if-statement in line 22 applied.
Thus, the procedure exits in Case 3 because a notification of integrity violation harms confidentiality. In this situation, there is a possible conflict between

integrity and confidentiality and

$$\Delta^{\mathsf{cexec}} = \emptyset \qquad \text{and} \qquad \mathcal{V}_t \qquad = \mathcal{V}_{t-1}. \tag{39}$$

In the following we look closer at the execution of cexec'. By Definition 7.1, agent $\mathcal{A}$ is aware that cexec' requires and maintains $\mathcal{V}$ to be a database view and further ensures the validity of the integrity constraints in the database instance. Thus, from agent $\mathcal{A}$'s point of view there are the following three sets of input database instances

$$DB_1 = \quad \{\mathit{db}'_{t-1} \models_{PL} \mathcal{C}_{t-1} \cup \mathcal{IC} \mid \mathit{db}'_{t-1} \bullet \mathcal{U} \not\models_{PL} \mathcal{IC}\}$$

Integrity check will fail on db'_{t-1}.

$$\begin{aligned} DB_2 = \quad & \{\mathit{db}'_{t-1} \models_{PL} \mathcal{C}_{t-1} \cup \mathcal{IC} \mid \mathit{db}'_{t-1} \bullet \mathcal{U} \models_{PL} \mathcal{IC} \text{ and} \\ & \mathsf{cexec}'(\mathcal{IC}, \mathit{db}'_{t-1}, \mathit{conf}, \mathcal{H}^{DB}_{t-1}, \mathcal{V}_{t-1}) = (\mathcal{IC}, \mathit{db}'_{t-1}, \ldots)\} \end{aligned}$$

Integrity check will pass on db'_{t-1}, but cexec' will not perform the update.

$$\begin{aligned} DB_3 = \quad & \{\mathit{db}'_{t-1} \models_{PL} \mathcal{C}_{t-1} \cup \mathcal{IC} \mid \mathit{db}'_{t-1} \bullet \mathcal{U} \models_{PL} \mathcal{IC} \text{ and} \\ & \mathsf{cexec}'(\mathcal{IC}, \mathit{db}'_{t-1}, \mathit{conf}, \mathcal{H}^{DB}_{t-1}, \mathcal{V}_{t-1}) = (\mathcal{IC}, \mathit{db}'_{t-1} \bullet \mathcal{U}, \ldots)\} \end{aligned}$$

Integrity check will pass on db'_{t-1} and cexec' will perform the update.

There are no further sets, since cexec' ensures atomicity and consistency of the ACID requirements. Further, we remark that due to consistency it holds that

$$\text{if } \mathit{db}_{t-1} \in DB_1 \text{ then } \mathsf{cexec}'(\mathcal{IC}, \mathit{db}_{t-1}, \ldots) = (\mathcal{IC}, \mathit{db}_{t-1}, \ldots). \tag{40}$$

Because the execution of cexec' is deterministic, it holds that

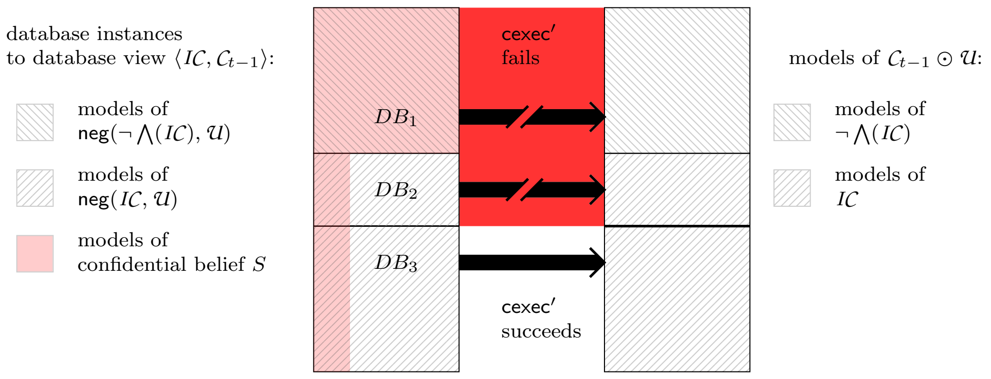
$$\mathsf{Mod}(\mathcal{C}_{t-1} \cup \mathcal{IC}) = DB_1 \uplus DB_2 \uplus DB_3 \text{ with disjoint union } \uplus\,. \tag{41}$$

Here, we use the notation $\mathsf{Mod}(\mathcal{M})$ for the set of models of a set $\mathcal{M} \subseteq \mathcal{L}_{PL}$ of formulas. Using Lemma 6.3, we obtain

$$DB_1 = \mathsf{Mod}(\mathcal{C}_{t-1} \cup \mathcal{IC} \cup \{\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})\}) \tag{42}$$

$$DB_2 \cup DB_3 = \mathsf{Mod}(\mathcal{C}_{t-1} \cup \mathcal{IC} \cup \mathsf{neg}(\mathcal{IC}, \mathcal{U})) \tag{43}$$

Figure A.3: Possible execution steps of cexec' at time $t-1$ considered by $\mathcal{A}$ when it reasons about system $\mathcal{R}_{\mathcal{D},\mathcal{A}}^{\mathsf{cexec}'}$, defined by control function cexec', at time $t-1$ on basis of the database view $\langle \mathcal{IC}, \mathcal{C}_{t-1} \rangle$


database instances
to database view $\langle \mathcal{IC}, \mathcal{C}_{t-1} \rangle$:
models of $\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})$
models of $\mathsf{neg}(\mathcal{IC}, \mathcal{U})$
models of confidential belief S
DB_1
DB_2
DB_3
cexec' fails
cexec' succeeds
models of $\mathcal{C}_{t-1} \odot \mathcal{U}$:
models of $\neg \bigwedge(\mathcal{IC})$
models of $\mathcal{IC}$

Further, because agent $\mathcal{A}$ knows the input, except the database instance, and the implementation of cexec', it can determine for each possible input db'_{t-1} to which set DB_i it belongs. The overall situation is shown in Figure A.3.

Claim 1 $\quad DB_2 \neq \emptyset$

Assume $DB_2 = \emptyset$. Let us consider that the input instance db_{t-1} to procedures cexec and cexec' is in DB_1 so that the integrity check will fail on this instance. This is possible because in the considered case of termination of procedure cexec (Procedure 2) the if-statement in line 17 did not apply. Because of the remark in (40), procedure cexec' does not update the database instance db_{t-1} whereas for all $db'_{t-1} \in DB_3$ the procedure succeeds with the update. Note that for the last conclusion we need implicitly that $DB_3 \cap (DB_1 \cup DB_2) = \emptyset$. By Assumption 7, procedure cexec' truthfully records the effective updates in the history so that

$$\mathcal{H}_t^{DB'} \neq \mathcal{H}_t^{DB''} \text{ with}$$

$$\begin{aligned} &\mathsf{cexec}'(\mathcal{IC}, db_{t-1}, \ldots) = (\ldots, \mathcal{H}_t^{DB'}, conf, \mathcal{V}'_t), && db_{t-1} \in DB_1 \text{ and} \\ &\mathsf{cexec}'(\mathcal{IC}, db'_{t-1}, \ldots) = (\ldots, \mathcal{H}_t^{DB''}, conf, \mathcal{V}''_t), && db'_{t-1} \in DB_3 \end{aligned}$$

Hence, by **(Soundness of Attacker View)** it follows that

$$db'_{t-1} \not\models_{PL} \mathsf{skeptical}(\mathcal{V}'_t) \text{ for all } db'_{t-1} \in DB_3. \tag{44}$$

To conclude, we argue in the following that the execution of cexec' on input instance $\mathcal{db}_{t-1}$ reveals to agent $\mathcal{A}$ that the instance has failed the integrity check, that is, it reveals that $\mathcal{db}_{t-1} \in DB_1$ holds:
It holds by **(No Loss of Information)** that $\mathsf{skeptical}(\mathcal{V}_{t-1}) \subseteq \mathsf{skeptical}(\mathcal{V}_t'')$. This means together with assumption $DB_2 = \emptyset$ that

$$\mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}_t'')) \subseteq \mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}_{t-1})) = \mathsf{Mod}(\mathcal{C}_{t-1} \cup \mathcal{IC}) = DB_1 \cup DB_3.$$

Now by (44) and by (42), it follows that

$$\mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}_t'')) \subseteq DB_1 = \mathsf{Mod}(\mathcal{C}_{t-1} \cup \mathcal{IC} \cup \{\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})\}).$$

We rewrite this equation to

$$\mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}_{t-1} \cup \{\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})\}\rangle) \subseteq \mathsf{skeptical}(\mathcal{V}_t'').$$

Finally, because the if-statement in line 22 of Procedure 2 applied to the input instance view $\mathcal{C}_{t-1}$, the last equation contradicts **(Confidentiality)** of procedure cexec' because of Lemma 7.6.

□(proof claim 1)

Claim 2 $DB_3 = \emptyset$

Assume the contrary and, now, let us consider that the input instance $\mathcal{db}_{t-1}$ to procedures cexec and cexec' is in DB_3. Next, we compare the execution of procedure cexec' on $\mathcal{db}_{t-1}$ to its execution on an instance $\mathcal{db}'_{t-1} \in DB_2$, which is possible according to Claim 1. Since cexec' is deterministic, it successfully updates $\mathcal{db}_{t-1}$, but not $\mathcal{db}'_{t-1}$. By Assumption 7, it truthfully records the effective updates in the history so that it follows that

$$\begin{aligned} &\mathcal{H}_t^{DB'} \neq \mathcal{H}_t^{DB''} && \text{with} \\ &\mathsf{cexec}'(\mathcal{IC}, \mathcal{db}_{t-1}, \ldots) = (\mathcal{IC}, \mathcal{db}_{t-1} \bullet \mathcal{U}, \mathit{conf}, \mathcal{H}_t^{DB'}, \mathit{conf}, \mathcal{V}_t'') && \text{and} \\ &\mathsf{cexec}'(\mathcal{IC}, \mathcal{db}'_{t-1}, \ldots) = (\mathcal{IC}, \mathcal{db}'_{t-1}, \mathit{conf}, \mathcal{H}_t^{DB''}, \mathit{conf}, \mathcal{V}_t'''). \end{aligned}$$

Hence, by **(Soundness of Attacker View)** we obtain

$$\mathcal{db}'_{t-1} \bullet \mathcal{U} \not\models_{PL} \mathsf{skeptical}(\mathcal{V}_t''). \tag{45}$$

Next, we argue that agent $\mathcal{A}$ gains more information than it requested through meta-inference, which may not happen by **(Cooperativeness)**. Because $\mathcal{db}'_{t-1}$ is in DB_2, by (43) and Lemma 6.3 it follows

$$\mathcal{db}'_{t-1} \bullet \mathcal{U} \in \mathsf{Mod}(\mathsf{neg}(\mathcal{C}_{t-1} \cup \mathcal{IC}, \mathcal{U}) \cup \mathcal{IC}).$$

Since we assumed by (7.6) that $\neg(\mathcal{U}) \subseteq \mathcal{C}_{t-1}$ it further holds that

$$\mathsf{Mod}(\mathsf{neg}(\mathcal{C}_{t-1} \cup \mathcal{IC}, \mathcal{U}) \cup \mathcal{IC}) = \mathsf{Mod}(\mathsf{neg}(\mathcal{C}_{t-1} \cup \neg(\mathcal{U}) \cup \mathcal{IC}, \mathcal{U}) \cup \mathcal{IC}) =$$
$$\mathsf{Mod}(\mathcal{IC} \cup \mathsf{neg}(\mathcal{C}_{t-1} \cup \mathcal{IC}, \mathcal{U}) \cup \mathcal{U}) = \mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}_{t-1} \odot \mathcal{U})).$$

Hence, together with (45) we have

$$\mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}_{t-1} \odot \mathcal{U})) \not\subseteq \mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}''_t)).$$

Finally, we rewrite the latter equation to

$$\mathsf{skeptical}(\mathcal{V}''_t) \not\subseteq \mathsf{skeptical}(\mathcal{V}_{t-1} \odot \mathcal{U})$$

which contradicts **(Cooperativeness)**.

□(proof claim 2)

By Claim 2 and the definition of DB_j $j = 1, 2$ and (39) it follows that

$$\Delta^{\mathsf{cexec}} = \emptyset = \Delta^{\mathsf{cexec}'}. \tag{46}$$

Still, we have to show

$$\mathsf{skeptical}(\mathcal{V}''_t) \subseteq \mathsf{skeptical}(\mathcal{V}_{t-1}), \tag{47}$$

where $\mathcal{V}_{t-1} = \mathcal{V}_t$ is the view output by cexec according to (39) and $\mathcal{V}''_t$ the view output by cexec' on input instance $\mathcal{db}_{t-1}$ and input view $\mathcal{V}_{t-1}$. We distinguish between the cases whether updating the input instance $\mathcal{db}_{t-1}$ satisfies the integrity constraints or not.

Case 4–a $\mathcal{db}_{t-1} \in DB_1$.

Assume now that the inclusion in (47) does not hold, which is, equivalently,

$$\mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}_{t-1})) \not\subseteq \mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}''_t)).$$

Then, there exists a $\mathit{db}'_{t-1} \in \mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}_{t-1})) \setminus \mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}'_t))$. This could not be db_{t-1}, that is $\mathit{db}'_{t-1} \neq \mathit{db}_{t-1}$, because $\mathcal{V}'_t$ is a database view of the database $\langle \mathcal{IC}, \mathit{db}_{t-1} \rangle$ by Definition 7.1 of the in- and output state of cexec' so that it holds $\mathit{db}_{t-1} \models_{PL} \mathsf{skeptical}(\mathcal{V}'_t)$.

Let $\mathcal{V}'''_t$ denote the view output by cexec' on input instance db'_{t-1}. This view is different from $\mathcal{V}'_t$ for the following reason. By Definition 7.1 $\mathcal{V}'''_t$ is a database view of the database $\langle \mathcal{IC}, \mathit{db}'_t \rangle$ and hence it holds $\mathit{db}'_t \models_{PL} \mathsf{skeptical}(\mathcal{V}'''_t)$. Further, by (46) we know that $\mathit{db}'_{t-1} = \mathit{db}'_t$ holds and thus
$\mathit{db}'_{t-1} \models_{PL} \mathsf{skeptical}(\mathcal{V}'''_t)$ holds, but $\mathit{db}'_{t-1} \not\models_{PL} \mathsf{skeptical}(\mathcal{V}'_t)$ by the choice of db'_{t-1}.

Because the views are different and visible to agent $\mathcal{A}$ (Definition 4.5), the agent rules out db_{t-1} as a possible input instance after cexec' has executed on input instance db'_{t-1}. Further, that execution did not update db'_{t-1} successfully by (46). Therefore, by **(Soundness of Attacker View)** it follows that

$$\mathit{db}_{t-1} \not\models_{PL} \mathsf{skeptical}(\mathcal{V}'''_t). \tag{48}$$

Finally, we will see that $\mathcal{V}'''_t$ reveals other information than the integrity violation because it rules out db_{t-1} and
$\mathit{db}_{t-1} \in DB_1 = \mathsf{Mod}(\mathcal{C}_{t-1} \cup \mathcal{IC} \cup \{\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})\})$ by Case 4–a and (42). The revelation of this additional information is not allowed by **(Cooperativeness)**. The technical argument is as follows:

$$\begin{aligned}
&\mathsf{Mod}(\mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}_{t-1} \cup \{\mathsf{neg}(\neg \textstyle\bigwedge(\mathcal{IC}), \mathcal{U})\}\rangle)) \\
&= \mathsf{Mod}(\mathcal{C}_{t-1} \cup \mathcal{IC} \cup \{\mathsf{neg}(\neg \textstyle\bigwedge(\mathcal{IC}), \mathcal{U})\}) = DB_1 \\
&\not\subseteq \mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}'''_t)) \qquad\qquad \text{by (48).}
\end{aligned}$$

But this contradicts **(Cooperativeness)**.

■ **Case** 4–b $\mathit{db}_{t-1} \in DB_2$.

Again, assume $\mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}_{t-1})) \not\subseteq \mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}'_t))$. There exist $\mathit{db}'_{t-1} \in DB_1$ because in the considered case of termination of procedure cexec (Procedure 2) the if-statement in line 17 did not apply. Let $\mathcal{V}'''_t$ denote the view output by cexec' on this instance. If it holds that $\mathcal{V}'''_t = \mathcal{V}'_t$ we can follow the line of argumentation of Case 4–a with db'_{t-1} which leads to a contradiction. Therefore, let $\mathcal{V}'''_t \neq \mathcal{V}'_t$. Then, similar to Case 4–a, the procedure cexec' does not update db'_{t-1} successfully, but updates db_{t-1}, so that **(Soundness of**

Attacker View) yields $\mathcal{db}_{t-1} \not\models_{PL} \mathsf{skeptical}(\mathcal{V}_t'')$. Finally, since $\mathcal{db}_{t-1} \models \mathsf{skeptical}(\mathcal{V}_{t-1})$ it holds that $\mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}_{t-1})) \not\subseteq \mathsf{Mod}(\mathsf{skeptical}(\mathcal{V}_t''))$ so that now we can conclude following the line of argumentation of Case 4–a with $\mathcal{db}_{t-1}' \in DB_1$.

By Claim 2 there are no other input instances to the database view $\mathcal{V}_{t-1}$ than considered in Case 4–a and Case 4–b.

$\boxed{\textbf{Case } 5\quad}$ Procedure 2 cexec terminates after the if-statement in line 26 applied.
Then, the procedure terminates with a notification of integrity violation and thus $\Delta^{\mathsf{cexec}} = \emptyset$. Since the procedure cexec' fulfills atomicity and consistency and the integrity check failed, cexec' cannot perform the update successfully and thus $\Delta^{\mathsf{cexec}'} = \emptyset$.

Procedure cexec computes $\mathcal{V}_t = \langle \mathcal{IC}, \mathcal{C}_{t-1} \cup \{\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})\}\rangle$. Since cexec' ensures **(Cooperativeness)**, it outputs a view $\mathcal{V}_t'$ such that

$$\mathsf{skeptical}(\mathcal{V}_t') \subseteq \mathsf{skeptical}(\langle \mathcal{IC}, \mathcal{C}_{t-1} \cup \{\mathsf{neg}(\neg \bigwedge(\mathcal{IC}), \mathcal{U})\}\rangle)$$

which proves $\mathsf{skeptical}(\mathcal{V}_t') \subseteq \mathsf{skeptical}(\mathcal{V}_t)$.

$\boxed{\textbf{Case } 6\quad}$ Procedure 2 terminates after line 33.
Then, the procedures passes the integrity check and updates the instance successfully which can be treated by Case 2.

$\square$(proof Thm. 7.2)

Preliminaries to the Proof of Theorem 8.1

In this section, we will introduce and verify lemmata and propositions to prepare the proof of Thm. 8.1 in the way that we sketched on page 174. First, we show in Prop. 8.1 that the existence of possible consequence relations under a class $\mathfrak{C}$ axiomatized by **Ax** may be tested by deduction with **Ax**. To simplify the proofs, we will make use of the following deduction theorem.

Proposition 8.2 (Deduction Theorem).
Cf. [82]

Let $\mathbf{Ax}$ *be an allowed set of axiom and rule schemes (Def. 8.9) and* $M \subset \mathcal{L}^*_{CL}$ *and* $F, G \in \mathcal{L}^*_{CL}$*. Then, if* $M \cup \{F\} \Vdash_{\mathbf{Ax}} G$ *holds then also* $M \Vdash_{\mathbf{Ax}} F \Rightarrow G$ *holds.*

The proof from [82] can be transfered to the deduction system of Def. 8.10 with only slight modifications.

Proposition 8.1

Let $\mathcal{V}$ *be a view and* $\mathcal{C}$ *a class of consequence relations axiomatized by an allowed set* $\mathbf{Ax}$ *of axiom and rules schemes (Asmp. 9). Then, there exists a possible consequence relation under* $\mathcal{V}$ *and* $\mathcal{C}$ *iff* $CL(\mathcal{V}) \nVdash_{\mathbf{Ax}} (\top \rightarrow \bot)$.

Only-If Part

By contradiction.

Assume: there exists a possible consequence relation $\mathrel{|\!\sim}$ under $\mathcal{V}$ and $\mathcal{C}$ (Def. 8.1).

Assume: $CL(\mathcal{V}) \Vdash_{\mathbf{Ax}} (\top \rightarrow \bot)$.

$$\emptyset \Vdash_{\mathbf{Ax}} \bigwedge(CL(\mathcal{V})) \Rightarrow (\top \rightarrow \bot) \quad \text{deduction thm.} \tag{49}$$

$$\emptyset \vdash_{\mathcal{C}} \bigwedge(CL(\mathcal{V})) \Rightarrow (\top \rightarrow \bot) \quad \text{(49) and } \mathcal{C} \text{ axiomatized by } \mathbf{Ax} \tag{50}$$

$$\mathrel{|\!\sim} \models_{CL} \bigwedge(CL(\mathcal{V})) \Rightarrow (\top \rightarrow \bot) \quad \text{(50), Def. 8.1 (}\mathcal{C}\text{ Defined)} \tag{51}$$

$$\mathrel{|\!\sim} \models_{CL} \bigwedge(CL(\mathcal{V})) \quad \text{Def. 8.1 (}\mathcal{B}\text{ Compatible, }\mathcal{C}\text{ Consistent)} \tag{52}$$

$$\mathrel{|\!\sim} \models_{CL} (\top \rightarrow \bot) \quad \text{(51), (52)} \tag{53}$$

Similarly, since $\neg(\top \rightarrow \bot) \in \mathbf{Ax}$ (Def. 8.9), since the class $\mathcal{C}$ is axiomatized by $\mathbf{Ax}$ and since $\mathrel{|\!\sim} \in \mathcal{C}$ (Def. 8.1, $\mathcal{C}$ Defined), we obtain $\mathrel{|\!\sim} \models_{CL} \neg(\top \rightarrow \bot)$ which contradicts (53).

If Part

Assume: there does not exist a possible consequence relation under $\mathcal{V}$ and $\mathfrak{C}$. (Def. 8.1)

Show: $CL(\mathcal{V}) \Vdash_{\mathbf{Ax}} (\top \to \bot)$.

By assumption, there does not exist a consequence relation $\mid\!\sim$ such that $\mid\!\sim \in \mathfrak{C}$ (Def. 8.1, $\mathfrak{C}$ Defined) and $\mid\!\sim \models_{CL} CL(\mathcal{V})$ (Def. 8.1, $\mathcal{B}$ Compatible and $\mathcal{C}$ Consistent). By definition of $\mathfrak{C}$-entailment, it holds that $CL(\mathcal{V}) \vdash_{\mathfrak{C}} (\top \to \bot)$. Because $\mathfrak{C}$ is axiomatized by $\mathbf{Ax}$, it holds that $CL(\mathcal{V}) \Vdash_{\mathbf{Ax}} (\top \to \bot)$.

$\square$(proof Prop. 8.1)

Lemmata

The key to the proof of Theorem 8.1 will be Proposition 8.3 saying that the following two problems are equivalent: (1) assume the visible part $\mathcal{C}_0$ as $\mathcal{D}$'s initial assertions, and after the interaction with history $\mathcal{H}^{NMR}$, find a possible consequence relation under $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})$; (2) find a set $\mathcal{E} \subset_{\mathit{fin}} \mathcal{L}_{PL}$, assume $\mathcal{C}_0 \cup \mathcal{E}$ as $\mathcal{D}$'s initial assertions, and then find a possible consequence relation under $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})$.

The proof of Proposition 8.3 builds on Proposition 8.1. The essential step in that proof is to translate a valid proof of $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}) \Vdash_{\mathbf{Ax}} (\top \to \bot)$ (non existence of a possible consequence relation under $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})$ by Proposition 8.1) to a valid proof of $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR}) \Vdash_{\mathbf{Ax}} (\top \to \bot)$ for any $\mathcal{E} \subset_{\mathit{fin}} \mathcal{L}_{PL}$.

Our approach is as follows. First, we will define a transformation from formulas in $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})$ to formulas in $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})$. Then, two lemmata give the desired translation of valid proofs. As the final preparation for the proof of Theorem 8.1, we will present and prove Proposition 8.3.

Given $\mathcal{E} \subset_{\mathit{fin}} \mathcal{L}_{PL}$ we introduce transformations between conditional expressions of the form $A \to B$ and of the form $A \wedge \bigwedge(\mathcal{E}) \to B$ in the following Definition 8.12. Let $\mathcal{L}^{*}_{CL}(\mathcal{E})$ denote the conditional language where in Definition 8.7 basic formulas are replaced by $A \wedge \bigwedge(\mathcal{E}) \to B$ with $A, B \in \mathcal{L}_{PL}$.

Definition 8.12 (Transformation from $\mathcal{L}^*_{CL}$ to $\mathcal{L}^*_{CL}(\mathcal{E})$)**.**

The transformation $\mathsf{t}_{\mathcal{E}}$ *is a function* $\mathsf{t}_{\mathcal{E}} : \mathcal{L}^*_{CL} \to \mathcal{L}^*_{CL}(\mathcal{E})$ *inductively defined on the structure of* $\mathcal{L}^*_{CL}$ *starting with* $\mathsf{t}_{\mathcal{E}}(A \to B) = A \wedge \bigwedge(\mathcal{E}) \to B$*.*

For the translation of a valid proof in the deduction system, we start with the translation of substitution instances of axiom schemes in **Ax** which are allowed schema formulas.

Lemma 8.1 (Translation of Axiom Schemes)**.**

Let $\mathbf{\Lambda} \in \mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL}))$ *be an allowed schema formula (Def. 8.8) and* $\mathsf{sub} : \mathcal{X}_{PL} \to \mathcal{L}_{PL}$ *a substitution. Let* $\mathcal{X}^{Pos}_{PL}, \mathcal{X}^{Neg}_{PL} \subseteq \mathcal{X}_{PL}$ *be a marking of* $\mathbf{\Lambda}$ *as required by Def. 8.8.*

$$\textit{Define } \mathsf{sub}_{\mathcal{E}} : \lambda \mapsto \begin{cases} \mathsf{sub}(\lambda) \wedge \bigwedge(\mathcal{E}) & \lambda \in \mathcal{X}^{Pos}_{PL}, \\ \mathsf{sub}(\lambda) \vee \neg \bigwedge(\mathcal{E}) & \lambda \in \mathcal{X}^{Neg}_{PL}, \\ \mathsf{sub}(\lambda) & \textit{otherwise.} \end{cases}$$

Then, it holds that $\emptyset \Vdash_{\mathbf{Ax}} \mathsf{sub}_{\mathcal{E}}(\mathbf{\Lambda}) \Rightarrow \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\mathbf{\Lambda}))$*.*

Proof We give a valid proof $G_1, \ldots, G_m$ of $\mathsf{sub}_{\mathcal{E}}(\mathbf{\Lambda}) \Rightarrow \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\mathbf{\Lambda}))$ from $\emptyset$ by induction on the structure of $\mathbf{\Lambda}$. Note that $\mathbf{\Lambda}$ is in conjunctive normal form. For the special case that $\mathbf{\Lambda}$ is of the form $(\neg)(\Phi \to \Gamma)$ we will present three essential results before we do the induction. For these results, the following tautologies of propositional logic will be useful:

$$((A \Rightarrow B) \wedge (C \Rightarrow D)) \Rightarrow (A \vee C \Rightarrow B \vee D), \tag{54}$$

$$((A \Rightarrow B) \wedge (C \Rightarrow D)) \Rightarrow (A \wedge C \Rightarrow B \wedge D). \tag{55}$$

Claim (56)

First, the following equivalence holds for $\mathbf{\Lambda} = (\neg)(\Phi \to \Gamma)$

$$\vdash_{pl} \mathsf{sub}_{\mathcal{E}}(\Phi) \Leftrightarrow (\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E})). \tag{56}$$

Proof of (56).

In (56), we consider allowed schema formulas of the form $(\neg)(\Phi \to \Gamma)$ with $\Phi, \Gamma \in \mathcal{L}_{PL}(\mathcal{X}_{PL})$ in conjunctive normal form. There must exist at least one conjunct Θ_d of Φ that meets the requirements of Def. 8.8 with the above marking $\mathcal{X}_{PL}^{Pos}$ and $\mathcal{X}_{PL}^{Neg}$. We proceed by induction on the number l of literals of Θ_d and show

$$\vdash_{pl} \mathsf{sub}_{\mathcal{E}}(\Theta_d) \Leftrightarrow \mathsf{sub}(\Theta_d) \wedge \bigwedge(\mathcal{E}). \tag{57}$$

$\boxed{\textbf{Case } 1}$ $l = 1$

$\boxed{\textbf{Case } 1\text{–a}}$ $\Theta_d = \lambda_1$.
Then, $\mathsf{sub}_{\mathcal{E}}(\Theta_d) = \mathsf{sub}_{\mathcal{E}}(\lambda_1) = \mathsf{sub}(\lambda_1) \wedge \bigwedge(\mathcal{E}) = \mathsf{sub}(\Theta_d) \wedge \bigwedge(\mathcal{E})$.

$\boxed{\textbf{Case } 1\text{–b}}$ $\Theta_d = \neg\lambda_1$.
Then, $\mathsf{sub}_{\mathcal{E}}(\Theta_d) = \neg\mathsf{sub}_{\mathcal{E}}(\lambda_1) = \neg(\mathsf{sub}(\lambda_1) \vee \neg \bigwedge(\mathcal{E}))$ and
$\neg(\mathsf{sub}(\lambda_1) \vee \neg \bigwedge(\mathcal{E})) \Leftrightarrow \neg\mathsf{sub}(\lambda_1) \wedge \bigwedge(\mathcal{E})$. Finally,
$\neg\mathsf{sub}(\lambda_1) \wedge \bigwedge(\mathcal{E}) = \mathsf{sub}(\Theta_d) \wedge \bigwedge(\mathcal{E})$

$\boxed{\textbf{Case } 2}$ $l \to l + 1$.

$\mathsf{sub}_{\mathcal{E}}(\Theta_d)$
$= \mathsf{sub}_{\mathcal{E}}((\neg)\lambda_1 \vee \ldots \vee (\neg)\lambda_l) \vee (\neg)\mathsf{sub}_{\mathcal{E}}(\lambda_{l+1})$
and $\mathsf{sub}_{\mathcal{E}}((\neg)\lambda_1 \vee \ldots \vee (\neg)\lambda_l) \vee (\neg)\mathsf{sub}_{\mathcal{E}}(\lambda_{l+1})$
$\Leftrightarrow (\mathsf{sub}((\neg)\lambda_1 \vee \ldots \vee (\neg)\lambda_l) \wedge \bigwedge(\mathcal{E})) \vee (\neg)\mathsf{sub}_{\mathcal{E}}(\lambda_{l+1})$ induct. hyp.
$\Leftrightarrow (\mathsf{sub}((\neg)\lambda_1 \vee \ldots \vee (\neg)\lambda_l) \wedge \bigwedge(\mathcal{E})) \vee ((\neg)\mathsf{sub}(\lambda_{l+1}) \wedge \bigwedge(\mathcal{E}))$ case $l = 1$
$\Leftrightarrow \mathsf{sub}(\Theta_d) \wedge \bigwedge(\mathcal{E})$

For the other conjuncts $\Theta_o = (\neg)\lambda_1 \vee \ldots \vee (\neg)\lambda_l$ in Φ, $o \neq d$, by definition of $\mathsf{sub}_{\mathcal{E}}$ and induction on l we show

$$\mathsf{sub}_{\mathcal{E}}(\Theta_o) \wedge \bigwedge(\mathcal{E}) \Leftrightarrow \mathsf{sub}(\Theta_o) \wedge \bigwedge(\mathcal{E}) : \tag{58}$$

$\boxed{\textbf{Case } 1}$ $l = 1$, $\Theta_o = (\neg)\lambda_1$.
The cases of $\lambda_1 \in \mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg}$ are analogous to Θ_d above. Consider
$\lambda_1 \in \mathcal{X}_{PL} \setminus (\mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg})$. Then,
$\mathsf{sub}_{\mathcal{E}}(\Theta_0)\wedge\bigwedge(\mathcal{E}) = (\neg)\mathsf{sub}_{\mathcal{E}}(\lambda_1)\wedge\bigwedge(\mathcal{E}) = (\neg)\mathsf{sub}(\lambda_1)\wedge\bigwedge(\mathcal{E}) = \mathsf{sub}(\Theta_o)\wedge\bigwedge(\mathcal{E})$.

$\boxed{\textbf{Case } 2}$ $l \to l+1$.

Analogous to the inductive case of Θ_d above.

Finally, we finish the proof of (56) as follows.

$$\begin{aligned}
\mathsf{sub}_{\mathcal{E}}(\Phi) &= \bigwedge_j \mathsf{sub}_{\mathcal{E}}(\Theta_j) \\
\text{and } & \bigwedge_j \mathsf{sub}_{\mathcal{E}}(\Theta_j) \\
&\Leftrightarrow \bigwedge_{j \neq d} \mathsf{sub}_{\mathcal{E}}(\Theta_j) \wedge \mathsf{sub}(\Theta_d) \wedge \bigwedge(\mathcal{E}) && \text{by (57)} \\
&\Leftrightarrow \bigwedge_{j \neq d} (\mathsf{sub}_{\mathcal{E}}(\Theta_j) \wedge \bigwedge(\mathcal{E})) \wedge \mathsf{sub}(\Theta_d) \wedge \bigwedge(\mathcal{E}) \\
&\Leftrightarrow \bigwedge_{j \neq d} (\mathsf{sub}(\Theta_j) \wedge \bigwedge(\mathcal{E})) \wedge \mathsf{sub}(\Theta_d) \wedge \bigwedge(\mathcal{E}) && \text{by (58)} \\
&\Leftrightarrow \mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E})
\end{aligned}$$

□(proof (56))

Claim (59)

The second result for the proof of Lem. 56 is the following implication for the case of $\mathbf{\Lambda} = (\Phi \to \Gamma)$

$$\vdash_{pl} \mathsf{sub}_{\mathcal{E}}(\Gamma) \Rightarrow \mathsf{sub}(\Gamma). \tag{59}$$

Proof of (59).

In (59), we consider allowed schema formulas of the form $\Phi \to \Gamma$ with $\Phi, \Gamma \in \mathcal{L}_{PL}(\mathcal{X}_{PL})$ in conjunctive normal form. We proceed by induction on the structure of Γ and show

$\vdash_{pl} \mathsf{sub}_{\mathcal{E}}(\Gamma) \Rightarrow \mathsf{sub}(\Gamma)$.

Literals.

$\boxed{\textbf{Case } 1}$ $\Gamma = \lambda \in \mathcal{X}_{PL}$.
In this case, λ occurs under an even number of negations so that either $\lambda \in \mathcal{X}_{PL} \setminus (\mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg})$ or $\lambda \in \mathcal{X}_{PL}^{Pos}$ because $\mathbf{\Lambda}$ complies with the marking.

$\boxed{\textbf{Case } 1\text{–a}}$ $\lambda \in \mathcal{X}_{PL} \setminus (\mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg})$.
By definition, $\mathsf{sub}_{\mathcal{E}}(\lambda) = \mathsf{sub}(\lambda)$.

$\boxed{\textbf{Case } 1\text{–b}}$ $\lambda \in \mathcal{X}_{PL}^{Pos}$.
By definition, $\mathsf{sub}_{\mathcal{E}}(\lambda) = \mathsf{sub}(\lambda) \wedge \bigwedge(\mathcal{E})$ and $\mathsf{sub}(\lambda) \wedge \bigwedge(\mathcal{E}) \Rightarrow \mathsf{sub}(\lambda)$.

$\boxed{\textbf{Case } 2}$ $\Gamma = \neg\lambda$.
In this case, λ occurs under an odd number of negations so that either $\lambda \in \mathcal{X}_{PL} \setminus (\mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg})$ or $\lambda \in \mathcal{X}_{PL}^{Neg}$ because $\mathbf{\Lambda}$ complies with the marking.

$\boxed{\textbf{Case } 2\text{–a}}$ $\lambda \in \mathcal{X}_{PL} \setminus (\mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg})$.
By definition, $\mathsf{sub}_{\mathcal{E}}(\lambda) = \mathsf{sub}(\lambda)$.

$\boxed{\textbf{Case } 2\text{–b}}$ $\lambda \in \mathcal{X}_{PL}^{Neg}$.
By definition, $\mathsf{sub}_{\mathcal{E}}(\neg\lambda) = \neg(\mathsf{sub}(\lambda) \vee \neg \bigwedge(\mathcal{E}))$ and $\neg(\mathsf{sub}(\lambda) \vee \neg \bigwedge(\mathcal{E})) \Rightarrow \mathsf{sub}(\neg\lambda)$.

Compound Formulas.

$\boxed{\textbf{Case } 1}$ $\Gamma = \Gamma_1 \vee \Gamma_2$.
We apply (54) to the induction hypothesis $\vdash_{pl} \mathsf{sub}_{\mathcal{E}}(\Gamma_i) \Rightarrow \mathsf{sub}(\Gamma_i)$ for $i = 1, 2$.

$\boxed{\textbf{Case } 2}$ $\Gamma = \Gamma_1 \wedge \Gamma_2$.
We apply (55) to the induction hypothesis $\vdash_{pl} \mathsf{sub}_{\mathcal{E}}(\Gamma_i) \Rightarrow \mathsf{sub}(\Gamma_i)$ for $i = 1, 2$.

$\square$(proof (59))

Claim (60)

The second result for the proof of Lem. 56 is the following implication for the case of $\mathbf{\Lambda} = \neg(\Phi \rightarrow \Gamma)$

$$\vdash_{pl} \mathsf{sub}(\Gamma) \Rightarrow \mathsf{sub}_{\mathcal{E}}(\Gamma). \tag{60}$$

Proof of (60).

In (60), we consider allowed schema formulas of the form $\neg(\Phi \to \Gamma)$ with $\Phi, \Gamma \in \mathcal{L}_{PL}(\mathcal{X}_{PL})$ in conjunctive normal form. We proceed by induction on the structure of Γ and show

$\vdash_{pl} \mathsf{sub}(\Gamma) \Rightarrow \mathsf{sub}_{\mathcal{E}}(\Gamma)$.

Literals.

Case 1 $\Gamma = \lambda \in \mathcal{X}_{PL}$.

In this case, λ occurs under an odd number of negations so that either $\lambda \in \mathcal{X}_{PL} \setminus (\mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg})$ or $\lambda \in \mathcal{X}_{PL}^{Neg}$ because $\mathbf{\Lambda}$ complies with the marking.

Case 1–a $\lambda \in \mathcal{X}_{PL} \setminus (\mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg})$.

By definition, $\mathsf{sub}_{\mathcal{E}}(\lambda) = \mathsf{sub}(\lambda)$.

Case 1–b $\lambda \in \mathcal{X}_{PL}^{Neg}$.

By definition, $\mathsf{sub}_{\mathcal{E}}(\lambda) = \mathsf{sub}(\lambda) \vee \neg \bigwedge(\mathcal{E})$ and $\mathsf{sub}(\lambda) \vee \neg \bigwedge(\mathcal{E}) \Leftarrow \mathsf{sub}(\lambda)$.

Case 2 $\Gamma = \neg\lambda$.

In this case, λ occurs under an even number of negations so that either $\lambda \in \mathcal{X}_{PL} \setminus (\mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg})$ or $\lambda \in \mathcal{X}_{PL}^{Pos}$ because $\mathbf{\Lambda}$ complies with the marking.

Case 2–a $\lambda \in \mathcal{X}_{PL} \setminus (\mathcal{X}_{PL}^{Pos} \cup \mathcal{X}_{PL}^{Neg})$.

By definition, $\mathsf{sub}_{\mathcal{E}}(\lambda) = \mathsf{sub}(\lambda)$.

Case 2–b $\lambda \in \mathcal{X}_{PL}^{Pos}$.

By definition, $\mathsf{sub}_{\mathcal{E}}(\neg\lambda) = \neg(\mathsf{sub}(\lambda) \wedge \bigwedge(\mathcal{E}))$ and

$\neg(\mathsf{sub}(\lambda) \wedge \bigwedge(\mathcal{E})) \Leftrightarrow \neg\mathsf{sub}(\lambda) \vee \neg \bigwedge(\mathcal{E}) \Leftarrow \mathsf{sub}(\neg\lambda)$.

Compound Formulas.

Case 1 $\Gamma = \Gamma_1 \vee \Gamma_2$.

We apply (54) to the induction hypothesis $\vdash_{pl} \mathsf{sub}(\Gamma_i) \Rightarrow \mathsf{sub}_{\mathcal{E}}(\Gamma_i)$.

Case 2 $\Gamma = \Gamma_1 \wedge \Gamma_2$.

We apply (55) to the induction hypothesis $\vdash_{pl} \mathsf{sub}(\Gamma_i) \Rightarrow \mathsf{sub}_{\mathcal{E}}(\Gamma_i)$.

$\square$(proof (60))

Proof of Lemma 8.1

Next, we prove Lem. 8.1 by induction on the structure of $\mathbf{\Lambda}$. In the proof, PL abbreviates the application of possibly several axiom and rule schemes of the outer propositional calculus.

Literals $(\neg)(\Phi \to \Gamma)$, $\Phi, \Gamma \in \mathcal{L}_{PL}(\mathcal{X}_{PL})$.

$\boxed{\textbf{Case } 1}$ $\mathbf{\Lambda} = \Phi \to \Gamma$.

We construct a valid proof of $\mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\Phi \to \Gamma))$ from $\{\mathsf{sub}_{\mathcal{E}}(\Phi \to \Gamma)\}$:

(i)	$\mathsf{sub}_{\mathcal{E}}(\Phi \to \Gamma)$	
(ii)	$\mathsf{sub}_{\mathcal{E}}(\Phi \to \Gamma) \Leftrightarrow (\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}_{\mathcal{E}}(\Gamma))$	(56), (LLE)
(iii)	$(\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}_{\mathcal{E}}(\Gamma))$	$(i), (ii)$, PL
(iv)	$(\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}_{\mathcal{E}}(\Gamma)) \Rightarrow (\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}(\Gamma))$	(59), (RW)
(v)	$\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}(\Gamma) = \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\Phi \to \Gamma))$	$(iii), (iv)$, (MP)

We conclude this case with the deduction theorem in Prop. 8.2.

$\boxed{\textbf{Case } 2}$ $\mathbf{\Lambda} = \neg(\Phi \to \Gamma)$.

We give a valid proof of $\mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\neg(\Phi \to \Gamma)))$ from $\{\mathsf{sub}_{\mathcal{E}}(\neg(\Phi \to \Gamma))\}$:

(i)	$\mathsf{sub}_{\mathcal{E}}(\neg(\Phi \to \Gamma))$	
(ii)	$\mathsf{sub}_{\mathcal{E}}(\Phi \to \Gamma) \Leftrightarrow (\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}_{\mathcal{E}}(\Gamma))$	(56), (LLE)
(iii)	$\neg(\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}_{\mathcal{E}}(\Gamma))$	$(i), (ii)$, PL
(iv)	$(\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}(\Gamma)) \Rightarrow (\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}_{\mathcal{E}}(\Gamma))$	(60), (RW)
(v)	$\neg(\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}_{\mathcal{E}}(\Gamma)) \Rightarrow \neg(\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}(\Gamma))$	PL
(vi)	$\neg(\mathsf{sub}(\Phi) \wedge \bigwedge(\mathcal{E}) \to \mathsf{sub}(\Gamma)) = \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\neg(\Phi \to \Gamma)))$	$(iii), (v)$, (MP)

Again, we finish the case with Prop. 8.2.

Compound Formulas. $\mathbf{\Lambda} = \mathbf{\Lambda_1} \vee \mathbf{\Lambda_2}$ ($\wedge$ respectively).

(i)	$\mathsf{sub}_{\mathcal{E}}(\mathbf{\Lambda_1}) \Rightarrow \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\mathbf{\Lambda_1}))$	induction hypothesis
(ii)	$\mathsf{sub}_{\mathcal{E}}(\mathbf{\Lambda_2}) \Rightarrow \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\mathbf{\Lambda_2}))$	induction hypothesis
(iii)	$\mathsf{sub}_{\mathcal{E}}(\mathbf{\Lambda}) \Rightarrow \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\mathbf{\Lambda}))$	$(i), (ii), (54)$ ((55) resp.)

$\square$(proof Lem. 8.1)

Lemma 8.2 (Translation of Proofs)**.**

Let $M \subset_{fin} \mathcal{L}^*_{CL}$ and $F_i, F \in \mathcal{L}^*_{CL}$ such that the sequence $F_1, \ldots, F_n$ is a valid proof of F from M with respect to an allowed set **Ax** *of axioms and rules schemes, Def. 8.9. Further, let $\mathcal{E} \subset_{fin} \mathcal{L}_{PL}$. Then, there exists a valid proof $G_1, \ldots, G_m$, $m \geq n$, of $\mathsf{t}_{\mathcal{E}}(F)$ from $\mathsf{t}_{\mathcal{E}}(M) \cup \{\neg(\bigwedge(\mathcal{E}) \rightarrow \bot)\}$ with respect to* **Ax** *with subsequence $\mathsf{t}_{\mathcal{E}}(F_1), \ldots, \mathsf{t}_{\mathcal{E}}(F_n) = G_m$.*

Proof We proceed by induction on the length n of proof $F_1, \ldots, F_n$.
Base case. We consider all deductions of F_1 in Def. 8.10.

$\boxed{\textbf{Case } 1}$ $F_1 \in M$.
Then, $\mathsf{t}_{\mathcal{E}}(F_1) \in \mathsf{t}_{\mathcal{E}}(M)$.

$\boxed{\textbf{Case } 2}$ $F_1 = \mathsf{sub}((\alpha \rightarrow \lambda) \Leftrightarrow (\beta \rightarrow \lambda))$ with $\mathsf{sub} : \mathcal{X}_{PL} \rightarrow \mathcal{L}_{PL}$ and $\vdash_{pl} \mathsf{sub}(\alpha) \Leftrightarrow \mathsf{sub}(\beta)$.

$$\text{Define } \mathsf{sub}_{\mathcal{E}} : \gamma \mapsto \begin{cases} \mathsf{sub}(\gamma) \wedge \bigwedge(\mathcal{E}) & \gamma = \alpha \text{ or } \gamma = \beta \\ \mathsf{sub}(\gamma) & \text{otherwise} \end{cases}$$

Then, $\mathsf{t}_{\mathcal{E}}(F_1) = \mathsf{sub}_{\mathcal{E}}((\alpha \rightarrow \lambda) \Leftrightarrow (\beta \rightarrow \lambda))$ following the inductive definitions of sub, $\mathsf{t}_{\mathcal{E}}$ and $\mathsf{sub}_{\mathcal{E}}$. $\mathsf{t}_{\mathcal{E}}(F_1)$ is a valid proof because $\vdash_{pl} \mathsf{sub}_{\mathcal{E}}(\alpha) \Leftrightarrow \mathsf{sub}_{\mathcal{E}}(\beta)$ follows from $\vdash_{pl} \mathsf{sub}(\alpha) \Leftrightarrow \mathsf{sub}(\beta)$.

$\boxed{\textbf{Case } 3}$ $F_1 = \mathsf{sub}((\alpha \rightarrow \beta) \Rightarrow (\alpha \rightarrow \lambda))$ with $\mathsf{sub} : \mathcal{X}_{PL} \rightarrow \mathcal{L}_{PL}$ and $\vdash_{pl} \mathsf{sub}(\beta) \Rightarrow \mathsf{sub}(\lambda)$.

$$\text{Define } \mathsf{sub}_{\mathcal{E}} : \gamma \mapsto \begin{cases} \mathsf{sub}(\gamma) \wedge \bigwedge(\mathcal{E}) & \gamma = \alpha \\ \mathsf{sub}(\gamma) & \text{otherwise} \end{cases}$$

Then, $\mathsf{t}_{\mathcal{E}}(F_1) = \mathsf{sub}_{\mathcal{E}}((\alpha \rightarrow \beta) \Rightarrow (\alpha \rightarrow \lambda))$. $\mathsf{t}_{\mathcal{E}}(F_1)$ is a valid proof because $\vdash_{pl} \mathsf{sub}_{\mathcal{E}}(\beta) \Rightarrow \mathsf{sub}_{\mathcal{E}}(\lambda)$ follows from $\vdash_{pl} \mathsf{sub}(\beta) \Rightarrow \mathsf{sub}(\lambda)$.

$\boxed{\textbf{Case } 4 \quad}$ $F_1 = \mathsf{sub}(\mathbf{\Lambda})$ where $\mathbf{\Lambda} \in \mathbf{Ax} \cap \mathcal{L}^*_{CL}(\mathcal{L}_{PL}(\mathcal{X}_{PL}))$ is allowed (Def. 8.8) and $\mathsf{sub} : \mathcal{X}_{PL} \to \mathcal{L}_{PL}$.
Then, Lem. 8.1 provides us with a substitution $\mathsf{sub}_{\mathcal{E}} : \mathcal{X}_{PL} \to \mathcal{L}_{PL}$ such that $\Vdash_{\mathbf{Ax}} \mathsf{sub}_{\mathcal{E}}(\mathbf{\Lambda}) \Rightarrow \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\mathbf{\Lambda}))$ holds. With the result of this lemma, we finish the case with a valid proof of $\mathsf{t}_{\mathcal{E}}(F_1)$ from $\emptyset$ in $\mathbf{Ax}$:

$$\begin{array}{llr}
(i) & \mathsf{sub}_{\mathcal{E}}(\mathbf{\Lambda}) & \text{axiom } \mathbf{\Lambda} \\
 & \ldots & \\
(ii) & \mathsf{sub}_{\mathcal{E}}(\mathbf{\Lambda}) \Rightarrow \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\mathbf{\Lambda})) & \text{Lem. 8.1} \\
(iii) & \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\mathbf{\Lambda})) = \mathsf{t}_{\mathcal{E}}(F_1) & (i), (ii), (\text{MP})
\end{array}$$

$\boxed{\textbf{Case } 5 \quad}$ $F_1 = \neg(\top \to \bot)$.

$$\begin{array}{llr}
(i) & \neg(\bigwedge(\mathcal{E}) \to \bot) & \\
(ii) & (\top \wedge \bigwedge(\mathcal{E}) \to \bot) \Leftrightarrow (\bigwedge(\mathcal{E}) \to \bot) & (\text{LLE}) \\
 & \neg(\top \wedge \bigwedge(\mathcal{E}) \to \bot) = \mathsf{t}_{\mathcal{E}}(F_1) & (i), (ii), \text{PL}
\end{array}$$

$\boxed{\textbf{Case } 6 \quad}$ $F_1 = \mathsf{sub}(\Lambda)$ where $\Lambda \in \mathcal{L}_{PL}(\mathcal{X}_{CL})$ and $\mathsf{sub} : \mathcal{X}_{CL} \to \mathcal{L}^*_{CL}$.
Then, define $\mathsf{sub}_{\mathcal{E}} : \mathcal{X}_{CL} \to \mathcal{L}^*_{CL}$ by $\mathsf{sub}_{\mathcal{E}} : \lambda \mapsto \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\lambda))$. By the inductive definition of $\mathsf{sub}_{\mathcal{E}}$, $\mathsf{t}_{\mathcal{E}}$ and sub, it holds that $\mathsf{sub}_{\mathcal{E}}(\Lambda) = \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\Lambda)) = \mathsf{t}_{\mathcal{E}}(F_1)$.

Inductive case:

We consider all deductions of F_n based on at least one predecessor F_j, $j < n$. In the cases, where the last deduction step does not depend on any predecessor in the proof sequence, we can argue as we did in the base case.
By the induction hypothesis, there exists a valid proof $G_1, \ldots, G_m$ of $\mathsf{t}_{\mathcal{E}}(F_{n-1})$ from $\mathsf{t}_{\mathcal{E}}(M) \cup \{\neg(\bigwedge(\mathcal{E}) \to \bot)\}$ with subsequence $\mathsf{t}_{\mathcal{E}}(F_1), \ldots, \mathsf{t}_{\mathcal{E}}(F_{n-1})$ and $m \geq n - 1$.
$F_n = \mathsf{sub}(\Lambda_{p+1})$ and there exists a rule scheme $\mathsf{If}\ \Lambda_1, \ldots, \Lambda_p\ \mathsf{then}\ \Lambda_{p+1} \in \mathbf{Ax}$ with $\Lambda_i \in \mathcal{L}_{PL}(\mathcal{X}_{CL})$ and a substitution $\mathsf{sub} : \mathcal{X}_{CL} \to \mathcal{L}^*_{CL}$ such that

$$\text{for all } j = 1, \ldots, p \text{ there exists } k_j < n \text{ with } F_{k_j} = \mathsf{sub}(\Lambda_j).$$

Define $\mathsf{sub}_{\mathcal{E}} : \mathcal{X}_{CL} \to \mathcal{L}^*_{CL}$ by $\mathsf{sub}_{\mathcal{E}} : \lambda \mapsto \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\lambda))$.

By the inductive definition of $\mathsf{sub}_{\mathcal{E}}$, $\mathsf{t}_{\mathcal{E}}$ and sub, it holds that

$$\mathsf{sub}_{\mathcal{E}}(\Lambda_{p+1}) = \mathsf{t}_{\mathcal{E}}(\mathsf{sub}(\Lambda_{p+1})) = \mathsf{t}_{\mathcal{E}}(F_n) \qquad \text{and } \mathsf{sub}_{\mathcal{E}}(\Lambda_j) = \mathsf{t}_{\mathcal{E}}(F_{k_j}) \text{ for all } j = 1, \dots, p.$$

By the induction hypothesis, for all $j = 1, \dots, p$ there exists $k'_j \leq m$ such that $\mathsf{sub}_{\mathcal{E}}(\Lambda_j) = \mathsf{t}_{\mathcal{E}}(F_{k_j}) = G_{k'_j}$. Hence, $G_1, \dots, G_m, \mathsf{t}_{\mathcal{E}}(F_n)$ is a valid proof with subsequence $\mathsf{t}_{\mathcal{E}}(F_1), \dots, \mathsf{t}_{\mathcal{E}}(F_n)$.

□(proof Lem. 8.2)

Proposition 8.3

Proposition 8.3 (Possible Consequence Relations under Hidden Assertions)**.**

Let $\mathcal{V} = \langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ be a view. Further, let $\mathfrak{C}$ be a class of consequence relations axiomatized by an allowed set **Ax** *of axiom and rule schemes as in Asmp. 9. Then, there exists a possible consequence relation under $\mathcal{V}$ and $\mathfrak{C}$ iff there exist a set $\mathcal{E} \subset_{fin} \mathcal{L}_{PL}$ (not necessarily a possible extension) such that there exists a possible consequence relation under $\mathcal{V}_{\mathcal{E}} = \langle \mathcal{B}^+_{\mathcal{E}}, \mathcal{B}^-_{\mathcal{E}}, \mathcal{C}_{\mathcal{E}} \rangle$ and $\mathfrak{C}$ where*

1. *for all $P, C \in \mathcal{L}_{PL}$ and $s \in \{+, -\}$:*
   $(P, C) \in \mathcal{B}^s$ iff $(P \wedge \bigwedge(\mathcal{E}), C) \in \mathcal{B}^s_{\mathcal{E}}$,
2. *$\mathcal{C}_{\mathcal{E}} = \mathcal{C} \cup \mathcal{E}$.*

Proof We may rewrite the equivalence to the two contrapositives. Then, the non-existence of possible consequence relation can be expressed by deduction with $\Vdash_{\mathbf{Ax}}$ by Prop. 8.1 yielding the following claim:

$$\begin{gathered} CL(\mathcal{V}) \Vdash_{\mathbf{Ax}} (\top \to \bot) \\ \text{iff} \\ \text{for all } \mathcal{E} \subset_{fin} \mathcal{L}_{PL}, \quad CL(\mathcal{V}_{\mathcal{E}}) \Vdash_{\mathbf{Ax}} (\top \to \bot) \end{gathered} \tag{61}$$

If-Part of (61)

The if-part of the claim (61) immediately follows when choosing $\mathcal{E} = \emptyset$. By definition in (6.4) and (8.3), every $F \in CL(\mathcal{V}_\emptyset')$ is of the form $F = (\neg)(\Phi \wedge \top \to \Gamma)$. In a valid proof $F_1, \ldots, F_n$ of $\top \to \bot$ from $CL(\mathcal{V}_\emptyset')$, we just replace every formula $F_i \in CL(\mathcal{V}_\emptyset')$ by the following sequence.

G_{i_1}	$\mathsf{t}_\emptyset^{-1}(F_i) = (\neg)(\Phi \to \Gamma)$	$CL(\mathcal{V})$
G_{i_2}	$\Phi \to \Gamma \Leftrightarrow (\Phi \wedge \top \to \Gamma)$	(LLE)
$G_{i_3}, \ldots, G_{i_k} := F_i$	F_i	G_{i_1}, G_{i_2}, PL

Here, PL abbreviates the application of possibly several axiom and rule schemes of the outer propositional calculus. The sequence $\langle G_1, \ldots, G_n \mid G_i = F_i \text{ if } F_i \notin CL(\mathcal{V}_\emptyset') \text{ else } G_i = G_{i_1}, \ldots, G_{i_k} \rangle$ is a valid proof of $\top \to \bot$ from $\mathsf{t}_\emptyset^{-1}(CL(\mathcal{V}_\emptyset')) = CL(\mathcal{V})$.

Only-If Part of (61)

The only-if-part of the claim (61) is almost immediate from Lem. 8.2 that translates a valid proof of the left-hand side of (61) to a valid proof of the right-hand side for any $\mathcal{E} \subset_{fin} \mathcal{L}_{PL}$. Using Lem. 8.2, we construct a valid proof from $CL(\mathcal{V}_\mathcal{E}')$ for arbitrary $\mathcal{E} \subset_{fin} \mathcal{L}_{PL}$.

(i)	$\bigwedge(\mathcal{E}) \to \bot$	hypothesis
(ii)	$\bigwedge(\mathcal{C}) \wedge \bigwedge(\mathcal{E}) \to \bot$	(IMP),(LLE)
(iii)	$(\bigwedge(\mathcal{E}) \to \bot) \Rightarrow (\bigwedge(\mathcal{C}) \wedge \bigwedge(\mathcal{E}) \to \bot)$	$(i), (ii)$, Prop. 8.2
(iv)	$\neg(\bigwedge(\mathcal{C}) \wedge \bigwedge(\mathcal{E}) \to \bot) \Rightarrow \neg(\bigwedge(\mathcal{E}) \to \bot)$	(iii), PL
(v)	$\neg(\bigwedge(\mathcal{C}) \wedge \bigwedge(\mathcal{E}) \to \bot)$	$CL(\mathcal{V}_\mathcal{E}')$
(vi)	$\neg(\bigwedge(\mathcal{E}) \to \bot)$	$(iv), (v)$, (MP)
	$\ldots$	
(vii)	$\mathsf{t}_\mathcal{E}(\top \to \bot) = \top \wedge \bigwedge(\mathcal{E}) \to \bot$	$CL(\mathcal{V}_\mathcal{E}'), (vi)$, Lem. 8.2
$(viii)$	$\bigwedge(\mathcal{E}) \to \bot$	(vii), (LLE),PL
(ix)	$\top \to \bot$	$(vi), (viii)$, PL

□(proof Prop. 8.3)

Theorem 8.1

Let $\mathcal{V}_0 = \langle \emptyset, \emptyset, \mathcal{C}_0 \rangle$ be the initial view as in Asmp. 8, $\mathfrak{C}$ a class of consequence relations axiomatized by **Ax** *as in Asmp. 9 and $\mathcal{H}^{NMR}$ a history. Then,*

$$\mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) = \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR}).$$

Inclusion $\subseteq$

By contraposition.

Assume: $B \notin \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$.

Show: $B \notin \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}))$.

By assumption and Def. 8.5, there exists a possible extension $\mathcal{E} \subset_{\mathit{fin}} \mathcal{L}_{PL}$ under $\mathcal{V}_0$ and $\mathfrak{C}$ such that $B \notin \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR}))$. Recall that $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})$ denotes the view computed in an interaction with history $\mathcal{H}^{NMR}$ assuming that the assertions $\mathcal{E}$ are hidden.

It follows that there exists a possible extension $\mathcal{E}$ under $\mathcal{V}_0$ and $\mathfrak{C}$ and, by Def. 8.2, that there exists a possible consequence relation $\mathrel{|\!\sim}$ under $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})$ and $\mathfrak{C}$ such that $\bigwedge(\mathcal{C}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})) \not\mathrel{|\!\sim} B$.

Hence, there exists a set $\mathcal{E} \subset_{\mathit{fin}} \mathcal{L}_{PL}$ and a possible consequence relation under $\mathcal{V}_{\mathcal{E}}$ and $\mathfrak{C}$ where

$$\begin{aligned}\mathcal{V}_{\mathcal{E}} := \langle & \mathcal{B}^+(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR}),\\ & \mathcal{B}^-(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR}) \cup \{(\bigwedge(\mathcal{C}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})), B)\}, \mathcal{C}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}^{NMR})\rangle.\end{aligned}$$

By Prop. 8.3, there exists a possible consequence relation under $\mathcal{V}$ and $\mathfrak{C}$ where

$$\begin{aligned}\mathcal{V} := \langle & \mathcal{B}^+(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}),\\ & \mathcal{B}^-(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}) \cup \{(\bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})), B)\}, \mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})\rangle.\end{aligned}$$

Inclusion $\supseteq$

Assume: $B \in \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$.

Show: $B \in \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}))$.

$\boxed{\textbf{Case } 1\quad}$ $\emptyset$ is a possible extension under $\mathcal{V}_0$ and $\mathfrak{C}$.
Then, by Def. 8.2 it holds that $\mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \supseteq$ $\mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$.

$\boxed{\textbf{Case } 2\quad}$ $\emptyset$ is not a possible extension under $\mathcal{V}_0$ and $\mathfrak{C}$.
Then, by Def. 8.3 there does not exist a possible consequence relation under $\mathcal{V}_0$ and $\mathfrak{C}$. Hence, following Def. 8.1 and Asmp. 8, there does not exist $\mid\!\sim\, \in \mathfrak{C}$ such that $\bigwedge(\mathcal{C}_0) \mid\!\not\sim \bot$. By definition, this means that $\emptyset \vdash_{\mathfrak{C}} \bigwedge(\mathcal{C}_0) \to \bot$. By Asmp. 9, **Ax** is complete for $\mathfrak{C}$ so that $\emptyset \Vdash_{\mathbf{Ax}} \bigwedge(\mathcal{C}_0) \to \bot$. We continue with a valid proof of $\bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \to \bot$ from $\emptyset$ in **Ax**:

$$(i) \qquad \bigwedge(\mathcal{C}_0) \to \bot$$

$$(ii) \qquad \bigwedge(\mathcal{C}_0) \wedge \bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \to \bot$$

by (i),(IMP),(MP)

$$(iii) \qquad \bigwedge(\mathcal{C}_0) \wedge \bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \to \bot \Leftrightarrow \bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \to \bot$$

by (LLE), $\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}) \supseteq \mathcal{C}_0$

$$(iv) \qquad \bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \to \bot$$

by (ii),(iii), PL

Since **Ax** is sound for $\mathfrak{C}$, there does not exist $\mid\!\sim\, \in \mathfrak{C}$ such that $\bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \mid\!\not\sim \bot$. Hence by Def. 8.1, there does not exist a possible consequence relation under $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})$ and $\mathfrak{C}$ so that by Def. 8.2 it holds that $\mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) = \mathcal{L}_{PL} \supseteq \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$.

$\square$(proof Thm. 8.1)

Corollary 8.2

Let $\mathfrak{C}$ be axiomatized by an allowed set **Ax** *of axioms and rule schemes (Asmp. 9). Then, for all $B \in \mathcal{L}_{PL}$ it holds that $B \in \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$*

iff $CL(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \Vdash_{\mathbf{Ax}} \bigwedge(\mathcal{C}) \rightarrow B$.

Proof Theorem 8.1 shows that $B \in \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}_0, \mathcal{H}^{NMR})$ iff $B \in \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}))$. The claim now follows right from the definitions as follows. By Def. 8.2, the skeptical entailment $B \in \mathsf{skeptical}_{\mathfrak{C}}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}))$ is equivalent to the fact that all possible consequence relations $\mathrel{|\!\sim}$ under $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})$ and $\mathfrak{C}$ yield $\bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \mathrel{|\!\sim} B$. By Def. 8.1 and (8.3), the latter is equivalent to the fact that all consequence relations $\mathrel{|\!\sim} \in \mathfrak{C}$ that satisfy $\mathrel{|\!\sim} \models_{CL} CL(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}))$ also satisfy $\mathrel{|\!\sim} \models_{CL} \bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \rightarrow B$ and, hence, $CL(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \vdash_{\mathfrak{C}} \bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \rightarrow B$. By Asmp. 9, the set **Ax** of axiom and rule schemes is an axiomatization of $\mathfrak{C}$ and, hence, the latter is equivalent to $CL(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \Vdash_{\mathbf{Ax}} \bigwedge(\mathcal{C}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR})) \rightarrow B$.

$\square$(proof Cor. 8.2)

Lemma for the Proof of Theorem 8.2

As we indicated in the proof sketch of Thm. 8.2, the proof of this theorem uses the following lemma. This lemma states that, if agent $\mathcal{A}$ assumes $\mathfrak{D}$'s initial assertions as correctly (that is, chooses $\mathcal{E} = \mathit{as}$), then, by applying Proc. 3 and Proc. 4, it approximates $\mathfrak{D}$'s fixed reasoning $\mathrel{|\!\sim}_{\mathfrak{D}}$ correctly, that is, $\mathcal{B}^+(\mathcal{V}_0, \mathit{as}, \mathcal{H}^{NMR}) \subseteq \mathrel{|\!\sim}_{\mathfrak{D}}$ and $\mathcal{B}^-(\mathcal{V}_0, \mathit{as}, \mathcal{H}^{NMR}) \subseteq \mathrel{|\!\not\sim}_{\mathfrak{D}}$.

Lemma 8.3 (Correct Approximation).

Let $\mathcal{H}^{NMR}$ be the history of an interaction Int with $\mathfrak{D}$'s fixed reasoning $\mathrel{|\!\sim}_{\mathfrak{D}}$ and initial assertions $\mathit{as} \subset_{\mathit{fin}} \mathcal{L}_{PL}$. Further, let $\mathfrak{C}$ be a class of consequence relations as in Asmp. 9 and $\mathcal{V}_0 = \langle \emptyset, \emptyset, \mathcal{C}_0 \rangle$ an initial view such that the precondition `pre` (8.1) *on page 164 is satisfied. Then,*

1. *$\mathrel{|\!\sim}_{\mathfrak{D}}$ is a possible consequence relation under $\mathcal{V}(\mathcal{V}_0, \mathit{as}, \mathcal{H}^{NMR})$ and $\mathfrak{C}$, and*

2. *the set of formulas $\mathcal{C}(\mathcal{V}_0, \mathit{as}, \mathcal{H}^{NMR})$ is logically equivalent to $\mathit{as}_{\mathit{Int}}$ where $\mathit{as}_{\mathit{Int}}$ are the assertions after the last request in Int.*

Proof For clarity of the presentation, we denote the initial assertions by $\mathcal{as}_0$. We prove the lemma by induction on the length k of $\mathcal{H}^{NMR}$.
Base case: $k = 0$

Here, the history is empty and $\mathcal{V}(\mathcal{V}_0, \mathcal{as}_0, \mathcal{H}^{NMR}) = \langle\emptyset, \emptyset, \mathcal{C}_0 \cup \mathcal{as}_0\rangle$. The precondition $\mathcal{as}_0 \vdash_{pl} F$ for all $F \in \mathcal{C}_0$ in the predicate `pre` ensures the second claim, that is,

$$\bigwedge(\mathcal{as}_0) \Leftrightarrow \bigwedge(\mathcal{C}_0 \cup \mathcal{as}_0) \text{ and } \bigwedge(\mathcal{C}_0 \cup \mathcal{as}_0) = \bigwedge(\mathcal{C}(\mathcal{V}_0, \mathcal{as}_0, \mathcal{H}^{NMR})). \quad (62)$$

The first claim follows from the following properties of $\mid\!\sim_{\mathfrak{D}}$ by Def. 8.1:
Property ($\mathfrak{C}$ Defined) holds by `pre`.
Property ($\mathcal{B}$ Compatible) immediately follows from $\mathcal{B}^+ = \emptyset \subseteq \mid\!\sim_{\mathfrak{D}}$ and $\mathcal{B}^- = \emptyset \not\mid\!\sim_{\mathfrak{D}}$.
Property ($\mathcal{C}$ Consistent) is justified as follows. By Def. 6.7, the assertions $\mathcal{as}_0$ are not contradictory under $\mid\!\sim_{\mathfrak{D}}$: $\bigwedge(\mathcal{as}_0) \not\mid\!\sim_{\mathfrak{D}} \perp$ and, hence, by definition

$$\mid\!\sim_{\mathfrak{D}} \models_{CL} \neg(\bigwedge(\mathcal{as}_0) \rightarrow \perp). \quad (63)$$

We apply the rule scheme (LLE) from **Ax** (Def. 8.9), using (62), and obtain $\emptyset \Vdash_{\mathbf{Ax}} (\bigwedge(\mathcal{C}(\mathcal{V}_0, \mathcal{as}_0, \mathcal{H}^{NMR})) \rightarrow \perp) \Leftrightarrow (\bigwedge(\mathcal{as}_0) \rightarrow \perp)$.
Since $\mathfrak{C}$ is axiomatized by **Ax**, it follows that
$\emptyset \vdash_{\mathfrak{C}} (\bigwedge(\mathcal{C}(\mathcal{V}_0, \mathcal{as}_0, \mathcal{H}^{NMR})) \rightarrow \perp) \Leftrightarrow (\bigwedge(\mathcal{as}_0) \rightarrow \perp)$,
and, hence, $\mid\!\sim_{\mathfrak{D}} \models_{CL} \neg(\bigwedge(\mathcal{C}(\mathcal{V}_0, \mathcal{as}_0, \mathcal{H}^{NMR})) \rightarrow \perp)$ by (63) and by $\mid\!\sim_{\mathfrak{D}} \in \mathfrak{C}$ (Property $\mathfrak{C}$ Defined). This means by definition, that
$\bigwedge(\mathcal{C}(\mathcal{V}_0, \mathcal{as}_0, \mathcal{H}^{NMR})) \not\mid\!\sim_{\mathfrak{D}} \perp$.

Inductive case: $k \rightarrow k + 1$

Without loss of generality, we consider that the $(k + 1)$-th request is a query `que`(A) answered with *true*. All the other cases can be treated with the same arguments. The second claim follows from the induction hypothesis doing the following transformations:

$$\mathcal{as}_{k+1} = \mathcal{as}_k \text{ and } \mathcal{as}_k \Leftrightarrow \mathcal{C}(\mathcal{V}_0, \mathcal{as}_0, \mathcal{H}_k^{NMR})$$
$$\text{and } \mathcal{C}(\mathcal{V}_0, \mathcal{as}_0, \mathcal{H}_k^{NMR}) = \mathcal{C}(\mathcal{V}_0, \mathcal{as}_0, \mathcal{H}_{k+1}^{NMR}). \quad (64)$$

Procedure 3 just adds the tuple $(\bigwedge(\mathcal{C}(\mathcal{V}_0, \mathcal{as}_0, \mathcal{H}_k^{NMR})), A)$ to the preceding view $\mathcal{V}(\mathcal{V}_0, as_0, \mathcal{H}_k^{NMR})$ which results in the view

$$\begin{aligned}&\mathcal{V}(\mathcal{V}_0, as_0, \mathcal{H}_{k+1}^{NMR}) = \\ &\qquad\langle \mathcal{B}^+(\mathcal{V}_0, as_0, \mathcal{H}_k^{NMR}) \cup \{(\bigwedge(\mathcal{C}(\mathcal{V}_0, as_0, \mathcal{H}_k^{NMR})), A)\}, \\ &\qquad\qquad \mathcal{B}^-(\mathcal{V}_0, as_0, \mathcal{H}_k^{NMR}), \mathcal{C}(\mathcal{V}_0, as_0, \mathcal{H}_k^{NMR})\rangle. \quad (65)\end{aligned}$$

Answering $\mathtt{que}(A)$ with *true* after the k-th request requires $A \in \mathsf{Bel}\,\mathrel{|\!\sim}_{\mathcal{D}}, as_k$ which in turn by (6.4) requires $\bigwedge(as_k) \mathrel{|\!\sim}_{\mathcal{D}} A$. Therefore, following the same arguments used in the base case, by (64) and (LLE) it holds that

$$\bigwedge(\mathcal{C}(\mathcal{V}_0, as_0, \mathcal{H}_k^{NMR})) \mathrel{|\!\sim}_{\mathcal{D}} A. \quad (66)$$

Applying the induction hypothesis, (65) and (66), we conclude that $\mathrel{|\!\sim}_{\mathcal{D}}$ is a possible consequence relation under $\mathcal{V}(\mathcal{V}_0, as_0, \mathcal{H}_{k+1}^{NMR})$ and $\mathcal{C}$.

□(proof Lem. 8.3)

Theorem 8.2

Agent $\mathcal{D}$ preserves temporary confidentiality towards $\mathcal{A}$ according to Def. 4.6 by means of the two procedures of the censor cexec *under Asmp. 8 and Asmp. 9.*

Invariant (8.5) **on page 183**

We show that Proc. 5 and Proc. 6 enforce the invariant

$$B \notin \mathsf{skeptical}_{\mathcal{C}}(\mathcal{V}) \text{ for each } B \in \mathit{conf}(TCP)$$

by induction on the number k of requests.

Base case: $k = 0$

The invariant is guaranteed by $\mathtt{pre}$ in (8.1) on page 164.

Inductive case: $k \to k+1$

$\boxed{\textbf{Case } 1}$ Proc. 5 (Controlled Query Execution).

Proc. 5 ensures the invariant in line 2 for each possible answer *ans*, that is, passing line 5 the procedure has ensured by line 2 that the following assertion holds

For each *ans* such that $\texttt{poss}(\texttt{app\_by\_que}(A, \mathit{ans}, \mathcal{V}), \mathcal{C})$ holds,

for each $B \in \mathit{conf}(\mathit{TCP}) :\ B \notin \mathsf{skeptical}_{\mathcal{C}}(\texttt{app\_by\_que}(A, \mathit{ans}, \mathcal{V}))$.

The view is only modified in line 6 by the correct answer $\texttt{eval}(A)(\mathrel{|\!\!\sim}_{\mathcal{D}}, \mathit{as})$ so that the invariant follows from the next claim which we will immediately verify:

If the censor notifies $\texttt{eval}(A)(\mathrel{|\!\!\sim}_{\mathcal{D}}, \mathit{as}) = \mathit{ans}$ in line 7,
then $\texttt{poss}(\texttt{app\_by\_que}(A, \mathit{ans}, \mathcal{V}), \mathcal{C})$ is satisfied. (67)

Since neither Proc. 5 nor Proc. 6 refine the view after a refusal, we define a history $\mathcal{H}^{NMR}_{k+1}$ of the controlled interaction considered up to the $(k+1)$-th request simply by omitting all refused requests. Doing so, we see that the censor computes the view

$$\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}_{k+1}) = \texttt{app\_by\_que}(A, \mathit{ans}, \mathcal{V}). \tag{68}$$

Lem. 8.3 implies that $\mathrel{|\!\!\sim}_{\mathcal{D}}$ is a possible consequence relation under $\mathcal{V}(\mathcal{V}_0, \mathit{as}, \mathcal{H}^{NMR}_{k+1})$ and $\mathcal{C}$. Now, by Prop. 8.3, there exists a possible consequence relation under the computed view (68) and $\mathcal{C}$ so that this view and $\mathcal{C}$ satisfy the predicate $\texttt{poss}$. $\square$ (67)

$\boxed{\textbf{Case } 2}$ Proc. 6 (Controlled Revision Execution).
In Proc. 6, the view $\mathcal{V}$ is only modified in line 10 according to the notification $\texttt{notify}$. Line 1 guarantees that notification *success* preserves the invariant but only if the predicate $\texttt{poss}(\texttt{app\_by\_rev}(A, \mathit{success}, \mathcal{V}), \mathcal{C})$ holds. Likewise, line 7 ensures the invariant for notification *failure* given that $\texttt{poss}(\texttt{app\_by\_rev}(A, \mathit{failure}, \mathcal{V}), \mathcal{C})$ holds. The predicates state that a revision by formula A may be successful or possibly fail, respectively, given the view $\mathcal{V}$. Formally, the procedure ensures that the following assertion holds:

If the censor notifies $\texttt{notify}(A)(\mathrel{|\!\!\sim}_{\mathcal{D}}, \mathit{as}) = \mathit{note}$ in line 11,
then $\texttt{poss}(\texttt{app\_by\_rev}(A, \mathit{note}, \mathcal{V}), \mathcal{C})$ is satisfied.

We can verify this claim analogously to (67), here with the view $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}^{NMR}_{k+1}) = \texttt{app\_by\_rev}(A, \mathit{note}, \mathcal{V})$.

$\square$(proof invariant (8.5))

Safe Final Situation

Let $\mathcal{V}_{Int} = \langle \mathcal{B}^{+}_{Int}, \mathcal{B}^{-}_{Int}, \mathcal{C}_{Int} \rangle$ denote the view finally computed by the censor after the sequence $\boldsymbol{Int}$ of requests. By the invariant of (8.5), there exists a possible consequence relation $\mathrel{|\!\sim}'$ under $\mathcal{V}_{Int}$ and $\mathfrak{C}$ such that $\bigwedge(\mathcal{C}_{Int}) \not\mathrel{|\!\sim}' B$. We take $\langle \mathrel{|\!\sim}', \mathcal{C}_0 \rangle$ as the initial alternative epistemic state. By construction and (6.4), it holds $B \notin \mathsf{Bel}^{NMR}_{\mathfrak{D}}(\mathrel{|\!\sim}', \mathcal{C}_{Int})$.

Indistinguishability

Let

θ_i	be the i-th request from agent $\mathcal{A}$,
$react_i$	$\mathfrak{D}$'s subsequent reaction to $\mathcal{A}$,
	when the interaction $\boldsymbol{Int}$ was started with $\mathfrak{D}$'s initial state
	$s = (\mathrel{|\!\sim}_{\mathfrak{D}}, \boldsymbol{as}_0, conf(TCP), \langle\rangle, \mathcal{V}_0\rangle$ with $\mathcal{V}_0 = \langle \emptyset, \emptyset, \mathcal{C}_0 \rangle$
$\mathcal{V}_i$	the subsequent view computed by the censor,
$\boldsymbol{as}_i$	$\mathfrak{D}$'s assertions thereafter,
$react'_i$,	
$\mathcal{V}'_i$, $\boldsymbol{as}'_i$	analogously with initial state $s' = (\mathrel{|\!\sim}', \mathcal{C}_0, conf(TCP), \langle\rangle, \mathcal{V}_0\rangle$.

We show by induction on the number i of requests, that

$$react'_i = react_i \qquad \text{and} \qquad \mathcal{V}'_i = \mathcal{V}_i \qquad \text{and} \qquad \boldsymbol{as}'_i = \mathcal{C}_i.$$

Base case: $i = 0$

Obviously, the alternative initial state s' satisfies `pre` of (8.1).
Inductive case: $(i - 1 \to i)$

$\boxed{\textbf{Case } 1}$ θ_i is a query request $\mathtt{que}(A)$.
By the induction hypothesis (equal views), for both initial epistemic states the meta-inference check in line 1-5 of Proc. 5 gives the same result because the check does not depend on the epistemic state. For the query evaluation (line 6 to end of Proc. 5), consider, without loss of generality, that

$$\mathtt{eval}(A)(\mathrel{|\!\sim}_{\mathfrak{D}}, \boldsymbol{as}_{i-1}) = true = react_i. \tag{69}$$

Then, Proc. 3 adds the tuple $(\bigwedge(\mathcal{C}_{i-1}), A)$ to $\mathcal{B}^{+}_{i-1}$ so that this tuple is also contained in the final view $\mathcal{V}_{Int}$. Because $\mathrel{|\!\sim}'$ defines a possible consequence relation under $\mathcal{V}_{Int}$, it holds that $\bigwedge(\mathcal{C}_{i-1}) \mathrel{|\!\sim}' A$ so that $A \in \mathsf{Bel} \mathrel{|\!\sim}_{\mathfrak{D}}{}', \mathcal{C}_{i-1}$

by (6.4). Applying the induction hypothesis $\mathcal{as}'_{i-1} = \mathcal{C}_{i-1}$, yields

$$\texttt{eval}(A)(\,\mid\!\sim_{\mathfrak{D}}{}', \mathcal{as}'_{i-1}) = \mathit{true} = \mathit{react}'_i.$$

Together with (69) it follows that $\mathit{react}'_i = \mathit{react}_i$ and, thus, by the induction hypothesis $\mathcal{V}'_i = \mathcal{V}_i$ and $\mathcal{C}'_i = \mathcal{C}_i$.

Case 2 θ_i is a revision request $\texttt{rev}(A)$.
The checks for disclosure of a confidential belief in line 1-6 of Proc. 6 are independent of the epistemic state. Hence, by the induction hypothesis, these checks give the same results for both epistemic states. The rest of the procedure depends on the result of $\texttt{notify}(A)(\,\mid\!\sim_{\mathfrak{D}}, \mathcal{as}_{i-1})$ and $\texttt{notify}(A)(\,\mid\!\sim_{\mathfrak{D}}{}', \mathcal{as}'_{i-1})$, respectively. Without loss of generality, we consider that

$$\texttt{notify}(A)(\,\mid\!\sim_{\mathfrak{D}}, \mathcal{as}_{i-1}) = \mathit{success} = \mathit{react}_i. \tag{70}$$

Then, Proc. 4 adds the tuple $(\bigwedge(\mathcal{C}_{i-1}) \wedge A, \bot)$ to $\mathcal{B}^-_{i-1}$ so that, with the same arguments as in Case 1, $\bigwedge(\mathcal{C}_{i-1}) \wedge A \not\mid\!\sim' \bot$. Applying the induction hypothesis $(\mathcal{as}'_{i-1} = \mathcal{C}_{i-1})$, it follows that $\bigwedge(\mathcal{as}'_{i-1}) \wedge A \not\mid\!\sim' \bot$ so that

$$\texttt{notify}(A)(\,\mid\!\sim_{\mathfrak{D}}{}', \mathcal{as}'_{i-1}) = \mathit{success} = \mathit{react}'_i.$$

Together with (70) it follows that $\mathit{react}'_i = \mathit{react}_i$ and, moreover, by the induction hypothesis and $\mathcal{as}'_i = \mathcal{as}'_{i-1} \cup \{A\} = \mathcal{C}_{i-1} \cup \{A\} = \mathcal{C}_i$ as well as $\mathcal{V}'_i = \mathcal{V}_i$.

□(proof Thm. 8.2)

Bibliography

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] Charu C. Aggarwal and Philip S. Yu, editors. *Privacy-Preserving Data Mining - Models and Algorithms*, volume 34 of *Advances in Database Systems*. Springer, 2008.

[3] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530, 1985.

[4] Mário S. Alvim, Miguel E. Andrés, and Catuscia Palamidessi. Probabilistic information flow. In *LICS*, pages 314–321. IEEE Computer Society, 2010.

[5] Krishnan S. Anand and Manu Goyal. Strategic information management under leakage in a supply chain. *Management Science*, 55(3):438–452, March 2009.

[6] G. Aldo Antonelli. Non-monotonic logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2012 edition, 2012.

[7] Musard Balliu, Mads Dam, and Gurvan Le Guernic. Encover: Symbolic exploration for information flow security. In *CSF*, pages 30–44. IEEE, 2012.

[8] Musard Balliu, Mads Dam, and Gurvan Le Guernic. Epistemic temporal logic for information flow security. *CoRR*, abs/1208.6106, 2012.

[9] François Bancilhon and Nicolas Spyratos. Update semantics of relational views. *ACM Transactions on Database Systems (TODS)*, 6(4):557–575, 1981.

[10] Mohua Banerjee and Didier Dubois. A simple modal logic for reasoning about revealed beliefs. In Claudio Sossai and Gaetano Chemello, editors, *ECSQARU 2009*, volume 5590 of *LNCS*, pages 805–816. Springer, 2009.

[11] Chitta Baral and Yan Zhang. Knowledge updates: Semantics and complexity issues. *Artificial Intelligence*, 164(1-2):209–243, 2005.

[12] Christoph Beierle and Gabriele Kern-Isberner. A conceptual agent model based on a uniform approach to various belief operations. In Bärbel Mertsching, Marcus Hund, and Muhammad Zaheer Aziz, editors, *KI 2009*, volume 5803 of *LNCS*, pages 273–280. Springer, Heidelberg, 2009.

[13] Salem Benferhat. Handling hard rules and default rules in possibilistic logic. In *Fifth International Conference on Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 1994)*, volume 945 of *Lecture Notes in Computer Science*, pages 302–310. Springer, 1994.

[14] Hassan Bezzazi, David Makinson, and Ramón Pino Pérez. Beyond rational monotony: Some strong non-Horn rules for nonmonotonic inference relations. *J. Log. Comput.*, 7(5):605–631, 1997.

[15] Joachim Biskup. *Security in Computing Systems - Challenges, Approaches and Solutions*. Springer, 2009.

[16] Joachim Biskup. Usability confinement of server reactions: Maintaining inference-proof client views by controlled interaction execution. In Shinji Kikuchi, Shelly Sachdeva, and Subhash Bhalla, editors, *DNIS 2010*, volume 5999 of *LNCS*, pages 80–106. Springer, 2010.

[17] Joachim Biskup. Inference control. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security (2nd Ed.)*, pages 600–605. Springer, 2011.

[18] Joachim Biskup. Inference-usability confinement by maintaining inference-proof views of an information system. *IJCSE*, 7(1):17–37, 2012.

[19] Joachim Biskup and Piero A. Bonatti. Controlled query evaluation for enforcing confidentiality in complete information systems. *International Journal of Information Security*, 3(1):14–27, 2004.

[20] Joachim Biskup and Piero A. Bonatti. Controlled query evaluation with open queries for a decidable relational submodel. *Annals of Mathematics and Artificial Intelligence*, 50(1-2):39–77, 2007.

[21] Joachim Biskup, Christian Gogolin, Jens Seiler, and Torben Weibert. Requirements and protocols for inference-proof interactions in information systems. In Michael Backes and Peng Ning, editors, *ESORICS 2009*, volume 5789 of *LNCS*, pages 285–302. Springer, 2009.

[22] Joachim Biskup, Christian Gogolin, Jens Seiler, and Torben Weibert. Inference-proof view update transactions with forwarded refreshments. *Journal of Computer Security*, 19(3):487–529, 2011.

[23] Joachim Biskup, Sven Hartmann, Sebastian Link, and Jan-Hendrik Lochner. Efficient inference control for open relational queries. In Sara Foresti and Sushil Jajodia, editors, *24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy XXIV (DBSec 2010)*, volume 6166 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 2010.

[24] Joachim Biskup, Gabriele Kern-Isberner, and Matthias Thimm. Towards enforcement of confidentiality in agent interactions. In Maurice Pagnucco and Michael Thielscher, editors, *NMR 2008*, pages 104–112. The University of New South Wales, 2008.

[25] Joachim Biskup, Jan-Hendrik Lochner, and Sebastian Sonntag. Optimization of the controlled evaluation of closed relational queries. In Dimitris Gritzalis and Javier Lopez, editors, *SEC 2009*, volume 297 of *IFIP*, pages 214–225. Springer, 2009.

[26] Joachim Biskup, Marcel Preuß, and Lena Wiese. On the inference-proofness of database fragmentation satisfying confidentiality constraints. In Xuejia Lai, Jianying Zhou, and Hui Li, editors, *ISC 2011*, volume 7001 of *LNCS*, pages 246–261. Springer, 2011.

[27] Joachim Biskup, Jens Seiler, and Torben Weibert. Controlled query evaluation and inference-free view updates. In Ehud Gudes and Jaideep Vaidya, editors, *DBSec*, volume 5645 of *LNCS*, pages 1–16. Springer, 2009.

[28] Joachim Biskup and Cornelia Tadros. Policy-based secrecy in the Runs & Systems framework and controlled query evaluation. In Isao Echizen, Noboru Kunihiro, and Ryôichi Sasaki, editors, *Short Paper of IWSEC 2010*, pages 60–77. IPSJ, 2010.

[29] Joachim Biskup and Cornelia Tadros. Inference-proof view update transactions with minimal refusals. In Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, Nora Cuppens-Boulahia, and Sabrina De Capitani di Vimercati, editors, *DPM 2011/SETOP 2011*, volume 7122 of *LNCS*, pages 104–121. Springer, 2012.

[30] Joachim Biskup and Cornelia Tadros. Revising belief without revealing secrets. In Thomas Lukasiewicz and Attila Sali, editors, *FoIKS 2012*, volume 7153 of *LNCS*, pages 51–70. Springer, 2012.

[31] Joachim Biskup and Cornelia Tadros. Preserving confidentiality while reacting on iterated queries and belief revisions. *Annals of Mathematics and Artificial Intelligence*, 2013. DOI 10.1007/s10472-013-9374-6.

[32] Joachim Biskup, Cornelia Tadros, and Lena Wiese. Towards controlled query evaluation for incomplete first-order databases. In Sebastian Link and Henri Prade, editors, *6th International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2010)*, volume 5956 of *Lecture Notes in Computer Science*, pages 230–247. Springer, 2010.

[33] Joachim Biskup and Torben Weibert. Confidentiality policies for controlled query evaluation. In Steve Barker and Gail-Joon Ahn, editors, *21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec 2007)*, volume 4602 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2007.

[34] Joachim Biskup and Torben Weibert. Keeping secrets in incomplete databases. *International Journal of Information Security*, 7(3):199–217, 2008.

[35] Joachim Biskup and Lena Wiese. Preprocessing for controlled query evaluation with availability policy. *Journal of Computer Security*, 16(4):477–494, 2008.

[36] Joachim Biskup and Lena Wiese. A sound and complete model-generation procedure for consistent and confidentiality-preserving databases. *Theoretical Computer Science*, 412(31):4044–4072, 2011.

[37] Richard Booth and Alexander Nittka. Reconstructing an agent's epistemic state from observations about its beliefs and non-beliefs. *J. Log. Comput.*, 18(5):755–782, 2008.

[38] Gerhard Brewka, Ilkka Niemelä, and Mirosław Truszczyński. Nonmonotonic reasoning. In Vladimir Lifschitz van Frank Harmelen and Bruce Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 239–284. Elsevier, Amsterdam, 2008.

[39] Birgit Burmeister, M. Arnold, Felicia Copaciu, and Giovanni Rimassa. BDI-agents for agile goal-oriented business processes. In Michael Berger, Bernard Burg, and Satoshi Nishiyama, editors, *AAMAS (Industry Track)*, pages 37–44, 2008.

[40] Valentina Ciriani, Sabrina de Capitani di Vimercati, Sara Foresti, Giovanni Livraga, and Pierangela Samarati. Enforcing confidentiality and data visibility constraints: An OBDD approach. In Yingjiu Li, editor, *DBSec 2011*, volume 6818 of *LNCS*, pages 44–59. Springer, 2011.

[41] Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Quantifying information flow with beliefs. *Journal of Computer Security*, 17(5):655–701, 2009.

[42] Michael R. Clarkson and Fred B. Schneider. Quantification of Integrity. *Computer Security Foundations Symposium, IEEE*, 0:28–43, 2010.

[43] Frédéric Cuppens and Alban Gabillon. Cover story management. *Data & Knowledge Engineering*, 37(2):177–201, 2001.

[44] Steven Dawson, Sabrina De Capitani di Vimercati, Patrick Lincoln, and Pierangela Samarati. Maximizing sharing of protected information. *Journal of Computer and System Sciences*, 64(3):496–541, 2002.

[45] James P. Delgrande, Didier Dubois, and Jérôme Lang. Iterated revision as prioritized merging. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 210–220. AAAI Press, 2006.

[46] Didier Dubois. Three scenarios for the revision of epistemic states. *J. Log. Comput.*, 18(5):721–738, 2008.

[47] Thomas Eiter and Thomas Lukasiewicz. Complexity results for default reasoning from conditional knowledge bases. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR 2000*, pages 62–73. Morgan Kaufmann, San Francisco, 2000.

[48] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge.* MIT Press, Cambridge, 1995.

[49] Csilla Farkas and Sushil Jajodia. The inference problem: A survey. *SIGKDD Explorations*, 4(2):6–11, 2002.

[50] Nir Friedman and Joseph Y. Halpern. Plausibility measures and default reasoning. *J. ACM*, 48(4):648–685, 2001.

[51] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4), 2010.

[52] Alban Gabillon. Multilevel databases. In Laura C. Rivero, Jorge Horacio Doorn, and Viviana E. Ferraggine, editors, *Encyclopedia of Database Technologies and Applications*, pages 386–389. Idea Group, 2005.

[53] Peter Gärdenfors. Belief revision and nonmonotonic logic: Two sides of the same coin? In *ECAI*, pages 768–773, 1990.

[54] Laura Giordano, Valentina Gliozzi, Nicola Olivetti, and Gian Luca Pozzato. Analytic tableau calculi for KLM rational logic R. In Michael Fisher, Wiebe van der Hoek, Boris Konev, and Alexei Lisitsa, editors, *JELIA*, volume 4160 of *LNCS*, pages 190–202. Springer, 2006.

[55] Laura Giordano, Valentina Gliozzi, and Gian Luca Pozzato. KLMLean 2.0: A theorem prover for KLM logics of nonmonotonic reasoning. In Nicola Olivetti, editor, *TABLEAUX 2007*, volume 4548 of *LNCS*, pages 238–244. Springer, Heidelberg, 2007.

[56] Moisés Goldszmidt and Judea Pearl. On the consistency of defeasible databases. *Artificial Intelligence*, 52(2):121–149, 1992.

[57] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–512, 2005.

[58] Joseph Y. Halpern and Kevin R. O'Neill. Secrecy in multiagent systems. *ACM Transactions on Information and System Security*, 12(1), 2008.

[59] Joseph Y. Halpern and Riccardo Pucella. Modeling adversaries in a logic for security protocol analysis. *The Computing Research Repository CoRR*, abs/cs/0607146, 2006.

[60] J.Y. Halpern. *Reasoning about Uncertainty.* MIT Press, Cambridge, 2005.

[61] Sven Ove Hansson. Logic of belief revision. In *Stanford Encyclopedia of Philosophy.* The Metaphysics Research Lab, 2011.

[62] Minghua He, Nicholas R. Jennings, and Ho fung Leung. On agent-mediated electronic commerce. *IEEE Trans. Knowl. Data Eng.*, 15(4):985–1003, 2003.

[63] Andreas Herzig. On updates with integrity constraints. In James Delgrande, Jerome Lang, Hans Rott, and Jean-Marc Tallon, editors, *Belief Change in Rational Agents: Perspectives from Artificial Intelligence, Philosophy, and Economics*, Dagstuhl Seminar Proceedings. Schloss Dagstuhl, Germany, 2005.

[64] Dominic Hughes and Vitaly Shmatikov. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.

[65] Aaron Hunter and James P. Delgrande. Iterated Belief Change: A Transition System Approach. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 460–465. Professional Book Center, 2005.

[66] Sushil Jajodia, Vijayalakshmi Atluri, Thomas F. Keefe, Catherine D. McCollum, and Ravi Mukkamala. Multilevel security transaction processing. *Journal of Computer Security*, 9(3):165–195, 2001.

[67] Sushil Jajodia and Catherine Meadows. Inference problems in multilevel secure database management systems. In Marshall D. Abrams, Sushil Jajodia, and Harold J. Podell, editors, *Information Security: An Integrated Collection of Essays*, pages 570–584. IEEE, 1995.

[68] Nicholas R. Jennings, Timothy J. Norman, Peyman Faratin, P. O'Brien, and Brian Odgers. Autonomous agents for business process management. *Applied Artificial Intelligence*, 14(2):145–189, 2000.

[69] Hirofumi Katsuno and Alberto O. Mendelzon. On the Difference between Updating a Knowledge Base and Revising It. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR 91)*, pages 387–394. 1991.

[70] Gabriele Kern-Isberner. Conditionals in nonmonotonic reasoning and belief revision - considering conditionals as agents. volume 2087 of *LNCS*. Springer, Heidelberg, 2001.

[71] Gabriele Kern-Isberner. A conceptual framework for (iterated) revision, update, and nonmonotonic reasoning. In Giacomo Bonanno, James P. Delgrande, Jérôme Lang, and Hans Rott, editors, *Formal Models of Belief Change in Rational Agents*, volume 07351 of *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl, Germany, 2007.

[72] Gabriele Kern-Isberner. Linking iterated belief change operations to nonmonotonic reasoning. In Gerhard Brewka and Jérôme Lang, editors, *KR 2008*, pages 166–176. AAAI Press, Menlo Park California, 2008.

[73] Sébastien Konieczny. Using transfinite ordinal conditional functions. In *Tenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2009)*, volume 5590 of *Lecture Notes in Computer Science*, pages 396–407. Springer, 2009.

[74] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2):167–207, 1990.

[75] Patrick Krümpelmann and Tim Janus. Angerona – a multiagent framework for logic based agents with application to secrecy preservation. Technical report, TU Dortmund University, 2013.

[76] Patrick Krümpelmann and Gabriele Kern-Isberner. On agent-based epistemic secrecy. In Riccardo Rossi and Stefan Woltran, editors, *NMR*, 2012.

[77] Daniel J. Lehmann and Menachem Magidor. What does a conditional knowledge base entail? *Artif. Intell.*, 55(1):1–60, 1992.

[78] David Makinson and Peter Gärdenfors. Relations between the logic of theory change and nonmonotonic logic. In André Fuhrmann and Michael Morreau, editors, *The Logic of Theory Change*, volume 465 of *LNCS*, pages 185–205. Springer, 1989.

[79] H. Mantel. *A uniform framework for the formal specification and verification of information flow security.* PhD thesis, Universität des Saarlandes, 2003.

[80] John McCarthy. *Programs with common sense.* Defense Technical Information Center, 1963.

[81] Yves Moinard. Plausibility structures for default reasoning. In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI 2004*, pages 853–857. IOS Press, Amsterdam, 2004.

[82] Steve Reeves and Michael Clarke. *Logic for Computer Science*. International Computer Science Series. Addison-Wesley, Wokingham, 1990.

[83] Raymond Reiter. What should a database know? *Journal of Logic Programming*, 14(1&2):127–153, 1992.

[84] Raymond Reiter. On specifying database updates. *The Journal of Logic Programming*, 25(1):53–91, 1995.

[85] Andrei Sabelfeld and David Sands. Declassification: Dimensions and principles. *Journal of Computer Security*, 17(5):517–548, 2009.

[86] Axel Schairer. Towards using possibilistic information flow control to design secure multiagent systems. In *First International Conference on Security in Pervasive Computing (SPC 2003)*, volume 2802 of *LNCS*, pages 101–115, 2003.

[87] Steven Shapiro, Maurice Pagnucco, Yves Lespérance, and Hector J. Levesque. Iterated belief change in the situation calculus. *Artificial Intelligence*, 175(1):165–192, 2011.

[88] Guillermo Ricardo Simari. A brief overview of research in argumentation systems. In Salem Benferhat and John Grant, editors, *5th International Conference on Scalable Uncertainty Management (SUM 2011)*, volume 6929 of *LNCS*, pages 81–95. Springer, 2011.

[89] Wolfgang Spohn. Ordinal conditional functions: A dynamic theory of epistemic states. In Brian Skyrms and William L. Harper, editors, *Irvine Conference on Probability and Causation*, volume II of *Causation in Decision, Belief Change, and Statistics*, pages 105–134. Kluwer, Dordrecht, 1988.

[90] Tyrone S. Toland, Csilla Farkas, and Caroline M. Eastman. The inference problem: Maintaining maximal availability in the presence of database updates. *Computers & Security*, 29(1):88–103, 2010.

[91] Johan van Benthem. Dynamic logic for belief revision. *Journal of Applied Non-Classical Logics*, 17(2):129–155, 2007.

[92] Leendert van der Torre. Logics for security and privacy. In Nora Cuppens-Boulahia, Frédéric Cuppens, and Joaquín García-Alfaro, editors, *DBSec*, volume 7371 of *LNCS*, pages 1–7. Springer, 2012.

[93] Lena Wiese. Keeping secrets in possibilistic knowledge bases with necessity-valued privacy policies. In Eyke Hüllermeier, Rudolf Kruse, and Frank Hoffmann, editors, *IPMU 2010*, volume 6178 of *LNCS*, pages 655–664. Springer, Heidelberg, 2010.

[94] Michael Wooldridge. Intelligent agents. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[95] Michael J. Wooldridge. *An Introduction to Multiagent Systems*. Wiley, Chichester, 2009.

[96] Xiaokui Xiao and Yufei Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou, editors, *ACM SIGMOD International Conference on Management of Data*, pages 689–700. ACM, 2007.

[97] Da Yong Zhang, Yong Zeng, Lingyu Wang, Hongtao Li, and Yuanfeng Geng. Modeling and evaluating information leakage caused by inferences in supply chains. *Computers in Industry*, 62(3):351–363, 2011.

[98] Dongmo Zhang, Norman Y. Foo, Thomas Andreas Meyer, and Rex Kwok. Negotiation as mutual belief revision. In Deborah L. McGuinness and George Ferguson, editors, *AAAI*, pages 317–323. AAAI Press / The MIT Press, 2004.

[99] Dongmo Zhang and Yan Zhang. A computational model of logic-based negotiation. In *AAAI*, pages 728–733. AAAI Press, 2006.

List of Figures

List of Definitions

List of Assumptions

List of Examples

List of Theorems, Propositions, Lemmata and Corollaries

Index