Developing Human-Robot Dialogue Management Formally *

Hui Shi John Bateman

Universität Bremen, Germany
shi@informatik.uni-bremen.de, bateman@uni-bremen.de

Abstract. In *shared-control systems*, such as intelligent service robots, a human operator and an automated technical system are interdependently in charge of control. Natural Language dialogues have long been acknowledged as a potentially fruitful modality for instructing, describing and negotiating in human-machine interfaces. Since shared-control systems are often embedded in safety-critical devices, formal methods are thus widely used for improving the quality of such systems. In this paper, we present a formal method based approach for dialogue management and show how it enhances the clarity of dialog modelling, provides several engineering properties (e.g., validation, test and simulation) and supports the generation of clarification subdialogues.

1 Motivation

It is important that a dialogue system be robust. But the testing of a dialogue system to ensure robustness is usually cumbersome and expensive. Moreover, if dialogue is the primary means of communication between a user and a safety critical technical system, such as an intelligent service robot or an aircraft, then the robustness and correctness of the dialogue system become indispensible. Little attention has so far been paid to these aspects of system design, however (e.g. [1]).

Some researchers in the Natural Language Processing Community have developed models of dialogue (e.g., [16]) that draw upon formalisms common in computer science, for example those relying on the idea that dialogues can be modeled as finite-state machines. In the Formal Methods Community, on the other hand, technical systems are modeled using forms of mathematical logic that can be subjected to very powerful analyses using mechanized theorem provers and model checkers. Since finite-state machines are among those formalisms used in formal methods, we can now combine ideas from both sides: modelling and controlling dialogue management using formal methods to ensure robustness and correctness.

* We gratefully acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG) through the Collaborative Research Centre SFB/TR 8 Spatial Cognition - Subproject I3-SharC.

Over the past ten years, assistive systems as well as smart rehabilitation robotics have becoming increasingly interesting for both industrial and academic research. The long-term goal of our chief experimental scenario, an autonomous wheelchair, called Rolland, is to provide an assistive system for daily use by elderly or handicapped people. Rolland makes use of laser range finding sensors and cameras to construct spatial representations of its environment, which can then form the basis for complex spatially-aware interactions.

We are currently exploring the application of the formal methods mentioned above for dialogue modeling within Rolland's dialogue system, to enable Rolland to engage in natural language dialogue with its user concerning tasks to be undertaken and destinations to be reached. Now we are exploring the use of these representations for interaction with its user; natural language has therefore become an important modality of communication with the system. This scenario brings with it some particular communication problems that must be solved. These include: *Mode confusions*, which occur if the human operator loses track of the mode transitions performed by the robot; and *knowledge disparities*, which happen if the robot's knowledge representation mismatches that of the user's. Identification of such situations is undoubtedly vital to developing a robot which does what the user expects it to do.

As an example, consider a scenario in which the wheelchair Rolland is driving down a corridor when a person suddenly steps into its path. Upon seeing the colleague, the user may decide to stop and talk for a moment, uttering "please halt". However, unbeknownst to the user, Rolland did not actually acknowledge the user's utterance, but decided to come to a stop of its own accord — having viewed the colleague as an obstacle. Thus, when the colleague moves on, the user will be surprised that the wheelchair continues on its path, despite the user not having instructed it to continue. Such mode confusions become increasingly common with more complex systems and are now well known to have potentially extremely serious consequences.

It is precisely in this area of recognizing the particular problems at hand that we are investigating how formal methods can help. Formal model checking is employed in order to detect automatically these and other confusing situations (e.g., [13]). Then, based on the automatic analysis of the conflict, subdialogues can be planned, filled with appropriate content, and generated in order to clear up such a misunderstanding as soon as it occurs.

2 The Approach

2.1 Architecture

The system architecture presented in Fig. 1 focuses on the concept of dialogue centred shared-control navigation robots, where the component *Robot controller* contains software for controlling the robot's behaviour; *Robot knowledge base* provides the robot with necessary spatial and conceptual information; *User's control model* represents the user's theory about the robot's control system; *User's knowledge model* represents the user's relevant knowledge.

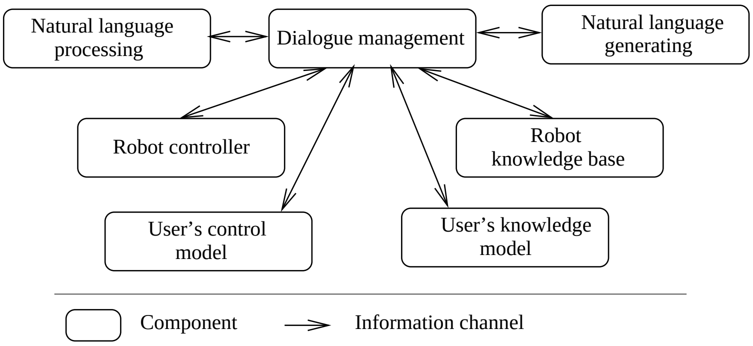

Fig. 1. A conceptual dialogue system architecture

2.2 Dialogue Modelling

Our approach to modelling dialogues is to move from empirically motivated dialogue patterns to models that can be used in computationally instantiated dialogue systems. The method followed here is to use the **CO**nversational **R**oles (COR) model [16] as a generic situation-independent dialogue model. This generic model can then be restricted as necessary so as to cover precisely particular empirically motivated discourse patterns that are revealed during our experiments. Naturally we will expect that the basis model itself may need to be extended or altered in the fact of dialogue evidence from our scenarios.

The view of dialogue given in the COR model is essentially a state transition graph. The corresponding dialogue models can be incorporated directly in Rolland's dialogue manager as Communicating Sequential Processes (CSP) as discussed in the following subsection. The COR model is broadly compatible with current *information state* approaches to dialogue (e.g., [10]), in that the nodes of the transition network represent particular abstract information states that a dialogue has reached. This means that we operate with a level of abstraction that draws from a broad information state that information that is particularly concerned with the dialogic structure; we return to this below. This is important in that it places a stronger structural emphasis on the kinds of transitions that are supported between states. Following these ideas we show in Fig. 2 a transition network giving a simplified dialogue model from our experiments with way-description tasks [4] –i.e., tasks where a user gives some description to Rolland of how to reach a destination that Rolland does not yet know. This is drawn from a real dialogic situation, albeit an extremely simple one in order to restrict the our necessary discussion here.

Here the interaction possibilities have been split into two networks, the first describing the user dialogic moves parameterized with participants, and the second giving the possiblities for the user to give instructions and additions, or for the system to request extensive feedback. In the transition network, parameterized moves, such as *instruct(user,robot)*, *add(user,robot)*, *dialogue(user,robot)*

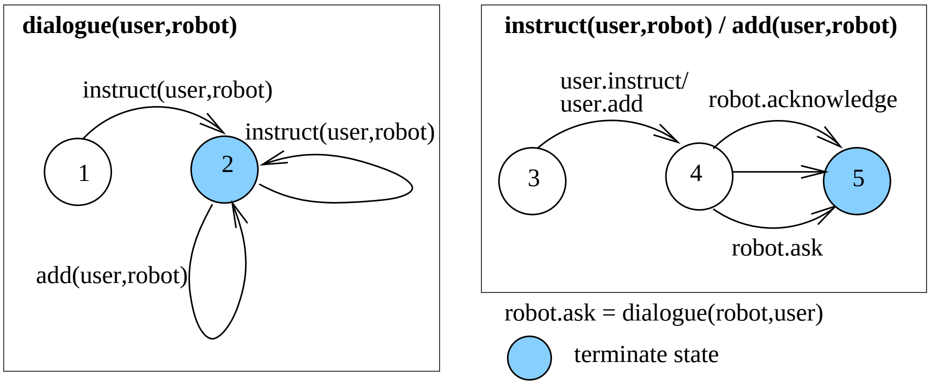

Fig. 2. A transition graph as simplified dialogue model

and *dialogue(robot,user)* are further defined as subdialogues. Moreover, moves like *user.instruct*, *user.add*, and *robot.acknowledge* are basic dialogue moves–i.e., moves that are directly implemented by other components as speech acts and actions.

2.3 Formalisation of Dialogue Model

We have chosen to apply the well developed method of *Communicating Sequential Processes* (CSP) from the formal methods community in order to model dialogue models because of its executability, good tool support, and our extensive experience with it (e.g., [14]). CSP is used for the specification of reactive systems in general; it can be seen as a very abstract, highly readable and easily maintainable language to specify finite state automata. Nevertheless, our approach is not restricted to CSP, other formal methods (e.g., [5], [3, 7], [11]) can also be applied.

In the *finite state approach* [15], dialogues are scripted utterance by utterance on a very concrete level. Each utterance leads to a new state, where various follow up utterances are allowed, each leading to a new state. In contrast to this approach, transition graph based dialogue models explain dialogue structure while abstracting from utterance details. In our dialogue model (see Fig. 2) the information about the basic dialogue acts of dialogue participants decides the dialogue structure.

Although our dialogue model is very generic, for a real dialogue management system, other information, such as *questions under discussion* and *beliefs*, must also be taken into account. As in TrindiKit [2] we use *stacks* and *queues* as data types to represent them. Each basic dialogue move leads to a state change in the dialogue model, and invokes one or more functions operating on these data structures, as well. Ususally, information states can not be captured by a finite-state model. What we do is to divide each information state into an abstract state and a content part. An *abstract state* contains only that information that is necessary for controlling dialogue transitions. For instance, if the user tells Rolland to find a route to a particular place P, then *find_place* as abstract question and P as content will be added into the questions under discussion. Fortunately, for our application and many other pratical dialogue systems, the set of abstract states is finite.

Moreover, domain specific components must also be considered. For the dialogues concerning Rolland's navigation system, a map of the environment space or a knowledge base including landmarks and their spatial relations is indispensible. The concrete information that is given in the reactions of a domain specific component, such as the *Robot's controller* or *Robot's knowledge base*, are abstracted for the purpose of the formal model into three categories: exact one positive answer to an action exists; several possible reactions exist; or no answer is possible. It is these abstract reactions, not concrete answers, that then influence dialogue transitions between the user and the robot. Communication between these components and the CSP model is provided by defining CSP 'channels' for information exchange and drivers for their implementation. The abstraction of communication data between robot and domain specific components can be extended or altered for different applications without complication.

Summarily, the formalization of the dialogue transition management consists of the following steps.

- Definition of CSP channels as interfaces to Rolland's components and to information states.
- Abstraction of communication data to achieve domain independence and to reduce the state space.
- Formalization of the dialogue control as CSP processes according to the dialogue model.

3 An implementation and some benefits

Once represented in CSP we can draw on a substantial body of methods for analysing abstract properties of our dialogue model that would otherwise not be available. Primarily among these we currently make use of FDR, (*Failures-Divergence Refinement*) [6], a model-checking tool for state machines with foundations in the theory of concurrency based upon CSP. Except for the ability to check determinism, primarily for checking security properties, its method of establishing whether a property holds is to test for the refinement (in one of the semantic models of CSP) of a transition system capturing the required specification by the candidate machine. The main ideas behind FDR are presented in [12].

Our prototypical implementation of a CSP dialogue architecture (see Fig. 3) includes mainly a state machine *Generator* using FDR; a *Validator* using FDR as model-checker; a *Simulator* with a graphical interface; and a set of interfaces for integrating other system components. The *Driver* controls internal dialogue states according to the given CSP spcification and events from both the user

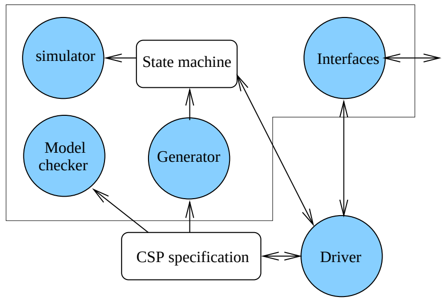

Fig. 3. The prototypical implementation

and the system. If the dialogue model is changed, all we need is to give a new CSP specification and a suitable driver. Other components remain unchanged.

The following example briefly sketches how our approach helps during the development of dialogue models. Before we could build the dialogue model presented in Fig. 2, which enforces a structuring on the dialogic tasks undertaken and explicitly represents joint dialogic projects that speakers are necessarily engaged on when enacting the interaction, two previous versions had been developed. A distinct problem with the first version derived directly from the preliminary survey of the empirical data; the problem was an apparent lack of regularity and generalization: i.e., that "speakers can do anything anytime". This version was then transfered to the kind of finite transition graph commonly used for dialogue models, where the arcs represent the dialogue acts and the nodes are information or discourse unit states. But this intermediate version in fact contained many re-occuring patterns, which were then drawn out explicitly in the transition graphs shown above. Importantly, the tranformation from one version to a new one is not always straightforward and modelling errors can occur, e.g., some dialogues accepted by a previous version may be rejected by the new one. Fortunately, with a CSP implementation we are now in the position to detect such transformation errors by verifying the equivalence of the CSP dialogue specification using the model-checker FDR. After detecting errors, we could made the necessary changes in the new version. As a result, we achieve a well-structured dialogue model, while also ensuring that this model correctly represents the empirical results, as well.

4 Conclusion

Through applying the formal method CSP the system described above now supports the validation of several properties of the dialogue system. These include such properties describing the relations between two dialogue models (inclusion, compatibility, etc.), fairness between different dialogue participants, and the absence of loops or the absence of locks which can easily be introduced by implementation errors. Moreover, the simulator provides a test and simulate environment for the whole dialogue management system from the software engineering point of view. Last but not least, the approach allows the automatic detection of the confusion situations mentioned in Section 1 above with respect to the current dialogue and robot system states. In such a situation, detailed information is added to the current information state in order to construct a (sub)dialogue to clear up the misunderstanding.

References

1. Alexandersson, J., Heisterkamp, P.: Some Notes on the complexity of Dialogues. In: Dybkjaer, L., Hasida, K. and Traum, D. (eds) Proceedings of the IJCAI'99 workshop on knowledge and reasoning in pratical dialogue systems (2000)
2. Cooper, R., Larsson, S.: Dialogue Moves and Information States. In Bunt, H.C., Thijsse, E.C.G. (eds): Proceedings of the third international workshop on computational semantics, Tilburg, Germany (1999)
3. Daws, C., Yovine, S.: Two Examples of Verification of Multirate Timed Automated With Kronos. In *Proc. 1995 IEEE Real-Time Systems Symposium.* IEEE Computer Society Press (1995) 66-77
4. Fisher, K.: First analysis of Rolland in advance way-description task corpus. Internal communication, Dec. 2004
5. Holzmann, G.J.: *Design and Validation of Computer Protocols.* Prentice-Hall (1991)
6. Formal Systemes: *Failures Divergence Refinement FDR2* Preliminary Manual. Formal Systems (Europe) Ltd (2001)
7. Henzinger, T., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic Model Checking for Real-Time System. *Information and Computation 111.* (1994) 193-244
8. Hoare C. A. R.: *Communicating Sequential Processes.* Prentice-Hall International (1985)
9. Krieg-Brückner, B., Shi, H., Ross, R.: *A Safe and Robust Apprach to Shared-control via Dialogue.* In Chinese Journal of Software, Vol.15, No. 12, 1744-1755 (2004)
10. Larsson S. and Traum D.: Information State and Dialogue Management in the TRINIDI Dialogue Move Engine Toolkit. In *Natural Language Engineering* pp. 323-340. Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering (2000)
11. McMillan, K.L.: *Symbolic Model Checking: An Approach to the State Explosion Problem.* Kluwer Academic Publishers, (1993)
12. Roscoe A. W. : *The Theory and Practice of Concurrency.* Prentice Hall (1998)
13. Rushby, J.: Using Model Checking to help Discover Mode Confusions and other Automation Surprises. *Reliability Engineering & System Safety* 75(2). Elsevier Science, 167-177 (2002)
14. Shi, H., Peleska, J., Kouvaras, M.: Combining methods for the analysis of a fault-tolerant system. In 1999 Pacific Rim International Symposium on Dependable Computing (PRDC). IEEE Computer Society (1999) 124-135
15. Sutton, S., Kayser, E.: The CSLU Rapid Prototyer: Version 1.8, Technical report. Oregon Graduate Institute, CSLU (1996)
16. Sitter, S., Stein, A.: Modeling the illocutionary aspects of information-seeking dialogues. Information Processing and Management 28, (1992) 165-180
17. Winograd, T., Flores, F.: Understanding Computers and Congnition. Addison-Wesley Professional; 1st edition (1987)