LDM: A Linear Dialogue Manager

Vladislav Maraev[⋆], **Jean-Philippe Bernardy**[⋆] **and Jonathan Ginzburg**[†]
[⋆]Centre for Linguistic Theory and Studies in Probability (CLASP),
Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg
`{vladislav.maraev,jean-philippe.bernardy}@gu.se`
[†]Laboratoire Linguistique Formelle (UMR 7110), Université de Paris
`yonatan.ginzburg@univ-paris-diderot.fr`

Abstract

For this system demonstration, we presents a modular dialog manager comprised of the following components. At the lowest level, we have implemented a proof-search engine based on linear logic. This engine features unification. In this engine we have implemented a suite of generic dialog-management rules, inspired from Ginzburg's KoS framework, such that firing a rule correspond to an information-state update of the agent. These generic rules can then be complemented by domain-specific rules. We show in particular how a question-answering agent, such as one operating an information kiosk, can be implemented.

1 Introduction

A key aspect of dialogue systems is the coherence of the system's responses. In this respect, a key component of a dialogue system is the dialogue manager, which selects appropriate system actions depending on the current state and the external context.

Two families of approaches to dialogue management can be considered: hand-crafted dialogue strategies (Allen et al., 1995; Larsson, 2002; Jokinen, 2009) and statistical modelling of dialogue (Rieser and Lemon, 2011; Young et al., 2010; Williams et al., 2017). Frameworks for hand-crafted strategies range from finite-state machines and form-filling to more complex dialogue planning and logical inference systems, such as Information State Update (ISU) (Larsson, 2002) that we employ here. Although there has been a lot of development in dialogue systems in recent years, only a few approaches reflect advancements in *dialogue theory*. Our aim is to closely integrate dialogue systems with work in theoretical semantics and pragmatics of dialogue.

We believe that it is crucial to use formal tools which are most appropriate for the task: one should be able to express the rules of various genres of dialogue in a concise way, free, to any possible extent, of irrelevant technical details. In the view of Dixon et al. (2009) this is best done by representing the information-state of the agents as updatable sets of propositions. Subsets of propositions in the information state can be treated independently, and, therefore, a suitable and flexible way to represent updates is as propositions in linear logic.

By using well-known techniques which correspond well with the intuition of information-state based dialogue management, we are able to provide a fully working prototype of the components of our framework:

1. a proof-search engine based on linear logic, modified to support inputs from external systems (representing inputs and outputs of the agent)

2. a set of rules which function as a core framework for dialogue management (in the style of KoS (Ginzburg, 2012))

3. several examples which use the above to construct potential applications of the system.

2 Linear logic as a Dialogue Management Framework

Typically, and in particular in the archetypal logic programming language prolog (Bratko, 2001), axioms and rules are expressed within the general framework of first order logic. However, several authors (Dixon et al., 2009; Martens, 2015) have proposed to use linear logic (Girard, 1995) instead. For our purpose, the crucial feature of linear logic is that hypotheses may be used *only once*.

In general, the linear arrow corresponds to *destructive state updates*. Thus, the hypotheses available for proof search correspond to the *state* of the system. In our application they will correspond to the *information state* of the dialogue participant.

This way, firing a linear rule corresponds to triggering an *action* of an agent, and a complete proof corresponds to a *scenario*, i.e. a sequence of actions, possibly involving action from several agents. However, the information state (typically in the literature and in this paper as well), corresponds to the state of a *single* agent. Thus, a scenario is conceived as a sequence of actions and updates of the information state of a single agent a, even though such actions can be attributed to any other dialogue participant b. (That is, they are a's representation of actions of b.) Scenarios can be realised as a sequence of actual actions and updates. That is, an action can result in sending a message to the outside world (in the form of speech, movement, etc.). Conversely, events happening in the outside world can result in updates of the information state (through a model of the perceptory subsystem).

In our implementation, we treat the information state as a multiset of *linear hypotheses* that can be queried. Because they are linear, these hypotheses can also be removed from the state. In particular, we have a fixed set of rules (they remain available even after being used). Each such rule manipulates a part of the information state (captured by its premisses) and leaves everything else in the state alone.

Our DM models the information-state of only one participant. Regardless, this participant can record its own beliefs about the state of other participants.In general, the core of DM is comprised of a set of linear-logic rules which depend on the domain of application. However, many rules will be domain-independent (such as generic processing of answers). We show these generic rules here, and the demo will illustrate them within an example application.

Integrating moves from NLU and NLG:

$$\begin{array}{l}
\mathit{hearAndRemember} : \\
\quad (m : \mathit{DP} \to \mathit{DP} \to \mathit{Move}) \to \\
\quad (x\ y : \mathit{DP}) \to (\mathit{ms} : \mathit{List\ Move}) \to \\
\quad \mathit{Heard}\ (m\ x\ y) \multimap \\
\quad \mathit{Moves}\ \mathit{ms} \multimap \mathit{HasTurn}\ x \multimap \\
\quad [\_ :: \mathit{Moves}\ (\mathit{Cons}\ (m\ x\ y)\ \mathit{ms}); \\
\quad\ \ \_ :: \mathit{Pending}\ (m\ x\ y); \\
\quad\ \ \_ :: \mathit{HasTurn}\ y];
\end{array}$$

$$\begin{array}{l}
\mathit{utterAndRemember} : \\
\quad (m : \mathit{DP} \to \mathit{DP} \to \mathit{Move}) \to \\
\quad (\mathit{ms} : \mathit{List\ Move}) \to (x\ y : \mathit{DP}) \to \\
\quad \mathit{Agenda}\ (m\ x\ y) \multimap \mathit{Moves}\ \mathit{ms} \multimap \\
\quad \mathit{HasTurn}\ x \multimap \\
\quad [\_ :: \mathit{Utter}\ (m\ x\ y); \\
\quad\ \ \_ :: \mathit{Moves}\ (\mathit{Cons}\ (m\ x\ y)\ \mathit{ms}); \\
\quad\ \ \_ :: \mathit{HasTurn}\ y];
\end{array}$$

$$\begin{array}{l}
\mathit{pushQUD} : \\
\quad (q : \mathit{Question}) \to (\mathit{qs} : \mathit{List\ Question}) \to \\
\quad (x\ y : \mathit{DP}) \to \mathit{Pending}\ (\mathit{Ask}\ q\ x\ y) \multimap \\
\quad \mathit{QUD}\ \mathit{qs} \multimap \mathit{QUD}\ (\mathit{Cons}\ q\ \mathit{qs})
\end{array}$$

Basic adjacency:

$$\begin{array}{l}
\mathit{counterGreeting} : \\
\quad (x\ y : \mathit{DP}) \to \mathit{HasTurn}\ x \rightarrowtriangle \\
\quad \mathit{Pending}\ (\mathit{Greet}\ y\ x) \multimap \\
\quad \mathit{Agenda}\ (\mathit{CounterGreet}\ x\ y);
\end{array}$$

Processing user replies:

$$\begin{array}{l}
\mathit{processAssert} : \\
\quad (a : \mathit{Type}) \to (x : a) \to (p : \mathit{Prop}) \qquad \to \\
\quad (\mathit{qs} : \mathit{List\ Question}) \to (\mathit{dp}\ \mathit{dp1} : \mathit{DP}) \to \\
\quad \mathit{Pending}\ (\mathit{Assert}\ p\ \mathit{dp1}\ \mathit{dp}) \qquad \multimap \\
\quad \mathit{QUD}\ (\mathit{Cons}\ (\mathit{Question}\ \mathit{dp}\ a\ x\ p)\ \mathit{qs}) \multimap \\
\qquad\quad [\_ :: \mathit{UserFact}\ p; \_ :: \mathit{QUD}\ \mathit{qs}];
\end{array}$$

$$\begin{array}{l}
\mathit{processShort} : \\
\quad (a : \mathit{Type}) \to (x : a) \to (p : \mathit{Prop}) \qquad \to \\
\quad (\mathit{qs} : \mathit{List\ Question}) \to (\mathit{dp}\ \mathit{dp1} : \mathit{DP}) \to \\
\quad \mathit{Pending}\ (\mathit{ShortAnswer}\ a\ x\ \mathit{dp1}\ \mathit{dp}) \multimap \\
\quad \mathit{QUD}\ (\mathit{Cons}\ (\mathit{Question}\ \mathit{dp}\ a\ x\ p)\ \mathit{qs}) \multimap \\
\qquad\quad [\_ :: \mathit{UserFact}\ p; \_ :: \mathit{QUD}\ \mathit{qs}];
\end{array}$$

Answering or clarifying:

$$\begin{array}{l}
\mathit{produceAnswer} : \\
\quad (a : \mathit{Type}) \to (x : a) \to_{!} (p : \mathit{Prop}) \to \\
\quad (\mathit{qs} : \mathit{List\ Question}) \to \\
\quad \mathit{QUD}\ (\mathit{Cons}\ (\mathit{Question}\ U\ a\ x\ p)\ \mathit{qs}) \multimap \\
\quad p \rightarrowtriangle \\
\quad [\_ :: \mathit{Agenda}\ (\mathit{ShortAnswer}\ a\ x\ S\ U); \\
\quad\ \ \_ :: \mathit{QUD}\ \mathit{qs}; \\
\quad\ \ \_ :: \mathit{Answered}\ (\mathit{Question}\ U\ a\ x\ p)];
\end{array}$$

$$\begin{array}{l}
\mathit{produceCR} : \\
[\quad a : \mathit{Type}; x : a; p : \mathit{Prop}; \\
\quad \mathit{qs} : \mathit{List\ Question}; \\
\quad \_ :: \mathit{QUD}\ (\mathit{Cons}\ (\mathit{Question}\ U\ a\ x\ p)\ \mathit{qs}); \\
\quad \_ :: p] \to_{?} \mathit{CR};
\end{array}$$

References

James F Allen, Lenhart K Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel Martin, Bradford Miller, Massimo Poesio, et al. 1995. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(1):7–48.

Ivan Bratko. 2001. *Prolog programming for artificial intelligence*. Pearson education.

Lucas Dixon, Alan Smaill, and Tracy Tsang. 2009. Plans, actions and dialogues using linear logic. *Journal of Logic, Language and Information*, 18(2):251–289.

Jonathan Ginzburg. 2012. *The interactive stance*. Oxford University Press.

J.-Y. Girard. 1995. *Linear Logic: its syntax and semantics*, London Mathematical Society Lecture Note Series, page 1–42. Cambridge University Press.

Kristiina Jokinen. 2009. *Constructive dialogue modelling: Speech interaction and rational agents*, volume 10. John Wiley & Sons.

Staffan Larsson. 2002. *Issue-based dialogue management*. Ph.D. thesis, University of Gothenburg.

Chris Martens. 2015. *Programming Interactive Worlds with Linear Logic*. Ph.D. thesis, Carnegie Mellon University Pittsburgh, PA.

Verena Rieser and Oliver Lemon. 2011. *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation*. Springer Science & Business Media.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.