Abstract Models for Dialogue Protocols

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

[joint work with Raquel Fernández (Potsdam)]

Dialogue Protocols

- Observation: frequently reoccurring sequences of utterance types in dialogue, e.g. *question-answer*, *proposal-acceptance*, etc.
- A *dialogue protocol* specifies the range of possible follow-ups available to a given participant at a given stage in a dialogue.
- Dialogue protocols are relevant to both *natural language* dialogue modelling and *multiagent systems:*
  - NLD: descriptive function; characterising range of unmarked follow-ups (expectations); evaluation via coverage of data
  - MAS: prescriptive function; defining simple rules for legal follow-ups; making interaction between software agents feasible
- Distinguish *protocol* (public) from *strategy* (private).

Talk Overview

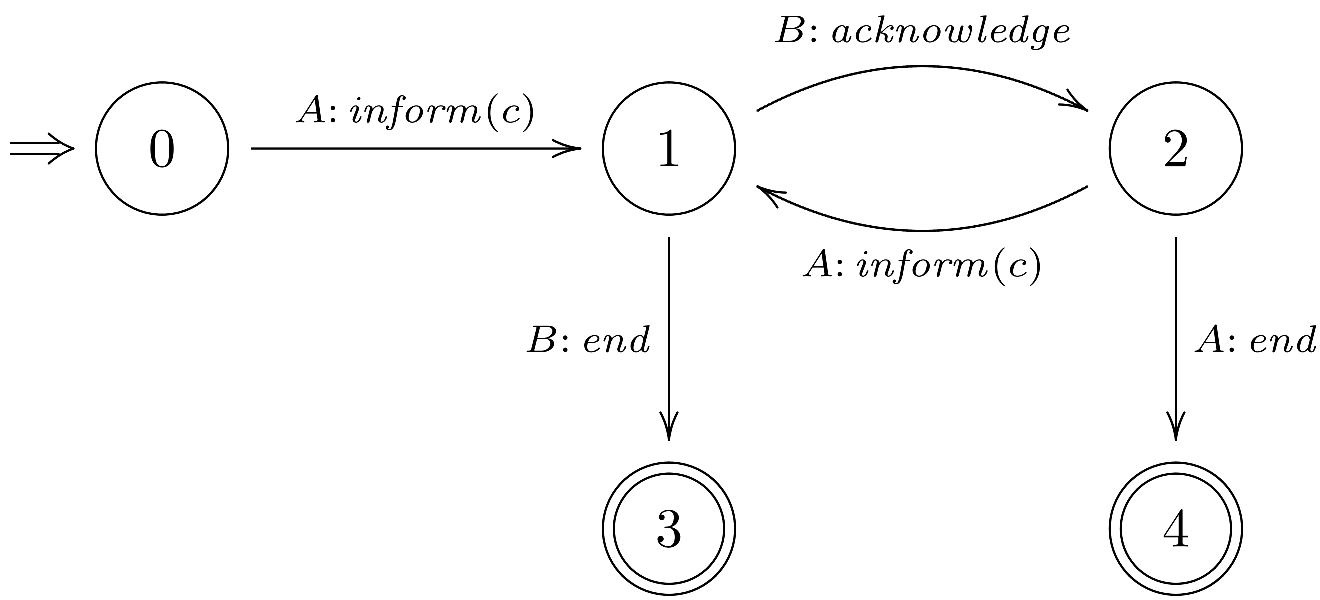
Different *features of dialogue* structure suggest different protocol models. This motivates a *hierarchy of abstract models* for dialogue protocols, to be presented in terms of different *machine models:*

- Protocols based on *deterministic finite automata*
- Enrichments of the basic model: adding a *memory component*
- A restriction of the basic model: *shallow protocols*

R. Fernández and U. Endriss. Abstract Models for Dialogue Protocols. *Journal of Logic, Language and Information*, 16(2):121–140, 2007.

Example

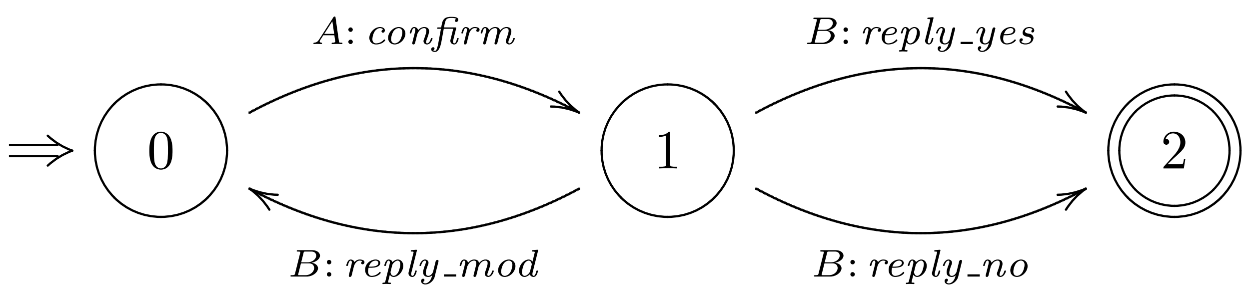
The *continuous update protocol* of Pitt & Mamdani (1999) is an example for a protocol that can be specified using a finite automaton:



J. Pitt and A. Mamdani. Communication Protocols in Multi-Agent Systems. Proc. Agents-1999 Workshop on Specifying and Implementing Conversation Policies.

Another Example

The next protocol specifies what the system (A) can expect from the user (B) in a situation where A asks B for confirmation of a previous utterance (Lewin, 1998):



I. Lewin. The Autoroute Dialogue. Technical Report CRC-073, SRI Intern., 1998.

Protocols as Finite Automata

Basic protocols are based on *deterministic finite automata* (DFAs).
A slight reformulation of the standard definition of a DFA:

- A *DFA-based protocol* is a quintuple $\langle Q, q_0, F, \mathcal{L}, \delta \rangle$, consisting of a finite set of dialogue states Q, including an initial state $q_0 \in Q$ and a set of final states $F \subseteq Q$, a (*finite*) communication language $\mathcal{L}$, and a transition function $\delta : Q \times \mathcal{L} \to Q$.

Crucially, a protocol specifies a range of possible dialogues:

- Given the current dialogue state q, an utterance u constitutes a *possible follow-up* of the dialogue iff there exists a state $q' \in Q$ such that $\delta(q, u) = q'$ holds.

- A (complete) dialogue *conforms* to a protocol iff it is *accepted* by the corresponding DFA.

Example

Replying to a question with another question (2) and asking for clarification (3) are common phenomena in dialogue:

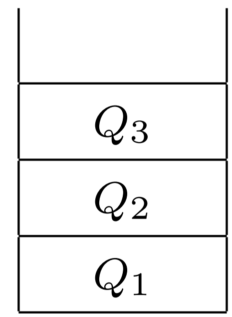
(1) A: Who should we invite?	$[Q_1]$
(2) B: Should we invite Bill?	$[Q_2]$
(3) A: Which Bill?	$[Q_3]$
(4) B: Jack's brother.	$[A_3]$
(5) A: Oh, yes.	$[A_2]$
(6) B: OK, then we should invite Gill as well.	$[A_1]$

We cannot really model this kind of phenomenon (embedded subdialogues) using our DFA-based protocols . . .

Protocols with a Stack

- We may use a *stack* to store questions:

  Questions get *pushed* onto the stack to be then *popped* by their respective answers.



- Finite automaton + stack = pushdown automaton

Protocols with Memory

Besides a stack, we could also use other *abstract data types* (ADTs) to enrich a DFA-based protocol with a memory component.

We arrive at the following definition:

- A *protocol with memory* based on a given ADT is a sextuple $\langle Q, q_0, F, \mathcal{L}, \mathcal{L}', \delta \rangle$, consisting of a finite set of dialogue states Q, including an initial state q_0 and a set of final states $F \subseteq Q$, a communication language $\mathcal{L}$, a memory alphabet $\mathcal{L}'$, and a transition function $\delta : Q \times \Gamma \times \mathcal{L} \rightarrow Q \times \Gamma$, where Γ denotes the set of all possible configurations of the memory component.

There are two restrictions on δ:

- δ is implementable in terms of the functions (e.g. *top*) and operations (e.g. *push*) of the chosen ADT.
- δ is representable as a finite subset of $(Q \times \Gamma \times \mathcal{L}) \times (Q \times \Gamma)$.

Possible Follow-ups

- Given the current dialogue state q and the current configuration of the memory component x, an utterance u constitutes a *possible follow-up* of the dialogue iff there exist a state $q' \in Q$ and a configuration $x' \in \Gamma$ such that $\delta(q, x, u) = (q', x')$.
- A (complete) dialogue *conforms* to a protocol iff it is *accepted* by the corresponding automaton.

Protocols with a Stack (again)

Ginzburg has used (something similar to) protocols with a stack as a means of modelling dialogue dynamics:

- Questions, once asked, get introduced into the so-called QUD ("*questions under discussion*").
- Assertion of a proposition p also introduces a question into QUD: $whether(p)$ — in dialogue, any contribution needs grounding.
- Once addressed, questions get removed from the QUD.
- Assuming that the last question asked is the most salient, a stack seems like the right ADT for the QUD (indeed, this is what has mostly been used for implementations).

J. Ginzburg. Interrogatives: Questions, Facts, and Dialogue. In S. Lappin (ed.), *Handbook of Contemporary Semantic Theory*, Blackwell Publishers, 1996.

Larsson *et al*. GoDiS: An Accommodating Dialogue System. Proc. NAACL-2000.

Expressive Power

Recall that DFA + stack = *pushdown automaton*. Hence:

Fact 1 *The class of dialogues conforming to protocols with a stack strictly includes that of dialogues conforming to DFA-based protocols.*

Some authors have also proposed protocols with *two stacks* (e.g. one for obligations, one for questions under discussion) $\leadsto$ Turing Machine

Discussion: Easy exercises from a computation-theoretic point of view, but interesting way of classifying complexity of dialogue management systems intended to handle dialogues with certain features.

Example

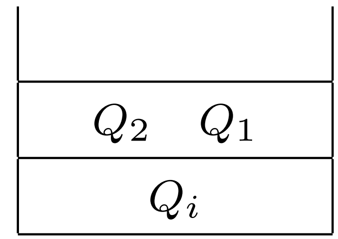
In real life, embedded *question-answer* sequences do not always follow the LIFO order suggested by a stack:

(1)	A: Where were you on the 15th?	$[Q_1]$
(2)	A: Do you remember talking to anyone after the incident?	$[Q_2]$
(3)	B: I didn't talk to anyone.	$[A_2]$
(4)	B: I was at home.	$[A_1]$
(3')	B: I was at home.	$[A_1]$
(4')	B: I didn't talk to anyone.	$[A_2]$

Protocols with a Stack of Sets

- We may use a *stack of sets* instead:

  Questions get either *pushed* on top of the stack or *inserted* into the top set.



- But how do we choose between the two operations?
  - From examples so far: different speakers $\Rightarrow$ push on top; same speaker $\Rightarrow$ insert into top set . . .
  - But the latter rule of thumb is not always correct:

(1) A: Who will you be inviting?	$[Q_1]$
(2) A: And why?	$[Q_2]$
(3) B: Mary and Bill, I guess.	$[A_1]$
(4) A: Aha.	$[Ack]$
(5) B: Yeah, (because) they are very undemanding folks.	$[A_2]$

  - Need to look at semantics: *coordination* vs. *query-extension*

Expressive Power

Fact 2 *The class of dialogues conforming to protocols with a stack is the same as that of dialogues conf. to protocols with a stack of sets.*

Two ways of proving this:

- Can translate any DFA equipped with a stack of sets with memory alphabet $\mathcal{L}'$ into a normal pushdown automaton (with a normal stack) using the power-set of $\mathcal{L}'$ as memory alphabet. ✓
- Can simulate a stack of sets using a normal but "big" stack by introducing a "separator" symbol. ✓

Protocols with a Set

- Protocols for *argumentation* modelling in multiagent systems need to express rules such as the following:

  You may only challenge an argument A if your opponent has previously asserted it.

- We may use a *set* to store arguments ("*commitment store*").
- Similar to *blackboard architecture*.

C.L. Hamblin. *Fallacies*. Methuen London, 1970.

L. Amgoud, N. Maudet, and S. Parsons. Modelling Dialogue using Argumentation. Proc. ICMAS-2000.

Expressive Power

Fact 3 *The class of dialogues conforming to DFA-based protocols is the same as the class of dialogues conforming to protocols with a set.*

Proof: The set of possible configurations of the "blackboard" is the power-set of the (finite) memory alphabet. So we can build a new DFA with a state for every pair of a state and a configuration of the original automaton (with a set component). ✓

Note that using *several sets* will also not increase expressive power.

Protocols with a List

- We can also use a *list* as an ADT to enrich a DFA-based protocol.
- Allows for storing and accessing the complete *dialogue history*.
- Most powerful, but also most costly model considered.

Expressive Power

A DFA with a stack is like a *Turing Machine*. Hence:

Fact 4 *The class of dialogues conforming to protocols with a list strictly includes that of dialogues conforming to protocols with a stack.*

Note that protocols with *several lists* would not increase expressive power any further (single-tape TMs can simulate multi-tape TMs).

Shallow Protocols

- Sometimes we might want to *restrict* the basic model . . .
- So-called *shallow protocol* are protocols where the legality of an utterance can be determined on the sole basis of the previous utterance in the dialogue.
- Example from a negotiation protocol:

$$A\!: propose \;\rightarrow\; \bigcirc\,(B\!: accept \;\vee\; B\!: reject \;\vee\; B\!: counter)$$

- Advantages: It is possible to check *a priori* whether an agent will always *conform* to a given protocol by inspecting the agent's specification (generally a very difficult problem).

U. Endriss, N. Maudet, F. Sadri, and F. Toni. Protocol Conformance for Logic-based Agents. Proc. IJCAI-2003.

Expressive Power

Formally, a DFA-based protocol is *shallow* iff the value of the transition function $\delta : Q \times \mathcal{L} \rightarrow Q$ is always uniquely identifiable given only its second argument (the utterance).

Fact 5 *The class of dialogues conforming to DFA-based protocols strictly includes the class of dialogues conforming to shallow protocols.*

Still, any DFA-based protocol can be *made* shallow by renaming transitions with the same name pointing to the same state.

Many DFA-based protocols from the literature are (almost) shallow.

Conclusion

- We have reviewed a variety of interesting dialogue features that give rise to different abstract models for dialogue protocols.
- These models have been presented either as enrichments or restrictions of our basic model:
  - basic model: deterministic finite automata
  - DFA + memory component (stack, stack of sets, set, list)
  - shallow protocols $\subset$ DFA
- Our abstract notion of a protocol provides a synthesis of work in multiagent systems and natural language dialogue.
- Connections to well-known machine models from the theory of computation offer a way of describing the complexity of dialogue.

R. Fernández and U. Endriss. Abstract Models for Dialogue Protocols. *Journal of Logic, Language and Information*, 16(2):121–140, 2007.