Temporal Logics for Representing Agent Communication Protocols

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

Talk Overview

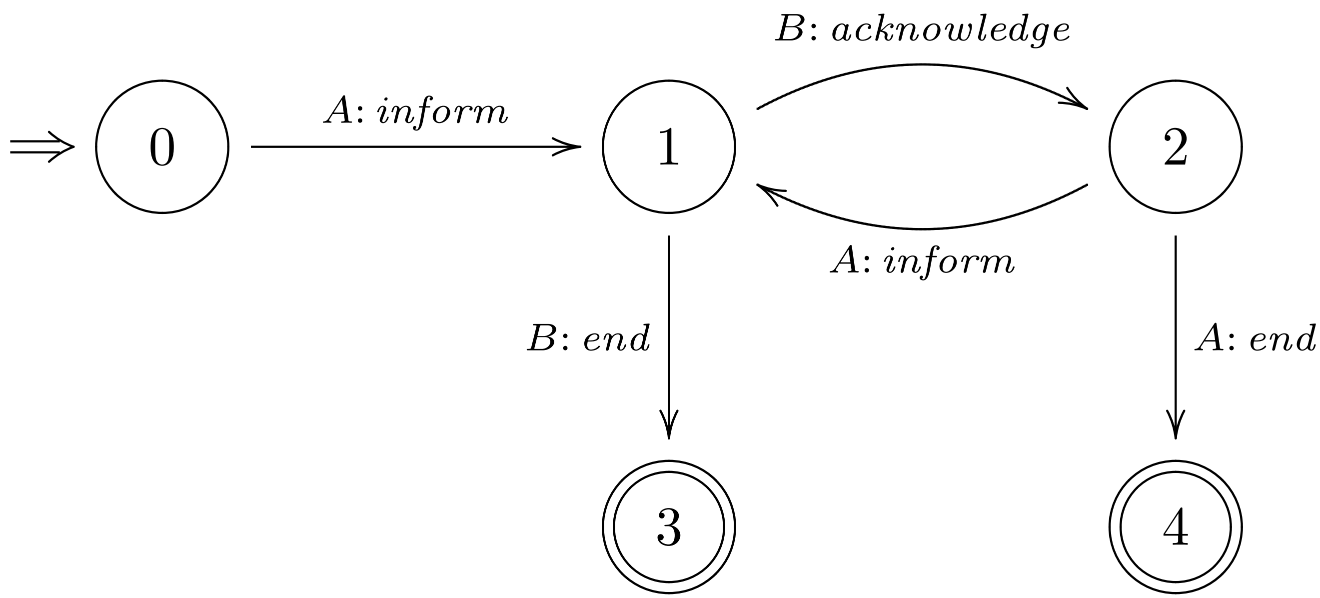
- Protocols in Convention-based Agent Communication
- Introduction to Temporal Logic
- Modelling Protocols using Linear Temporal Logic
- Two Case Studies:
  - Modelling Automata-based Protocols
  - Modelling Future Obligations
- Outlook: A Logic for Nested Protocols
- Conclusions

Communication in Open Systems

- Two schools of thought: *"mentalistic"* vs. *"conventionalist"* approach to agent communication
- *Mental* attitudes (beliefs, intentions) are useful to explain *why* agents may behave in certain ways, but (being non-verifiable) they cannot serve as a basis for building open systems that allow for meaningful communication.
- A somewhat more promising approach to agent communication relies on public norms and *conventions* as a means of specifying the rules of social interaction.
- In the convention-based approach, *protocols* specify the range of *legal follow-ups* available to the participating agents in a given dialogue (or multilogue).
- This talk is about the specification of such protocols.

Example

The "continuous update protocol" (Pitt & Mamdani, IJCAI-1999) is an example for a communication protocol that can be specified using a finite automaton:



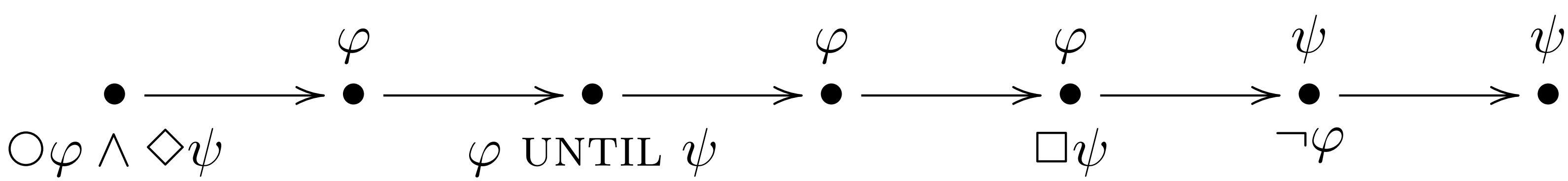
▶ We are going to get back to this one in a bit . . .

Why Temporal Logic?

- Why logic? — Because we want something formal with an unambiguous semantics.
- Why (propositional) modal logic? — Because we want something that is both computationally simple and easy to understand.
- Why not something BDI? – Because we have subscribed to the conventionalist approach (see earlier slide).
- Why not some sort of deontic logic? — Because we are not interested in analysing the nature of norms themselves.
- So why temporal logic? — Temporal logic formulas can be used to specify which sequences of utterances are legal according to a given protocol. The notion of what an agent *ought* to do is then implicit: the social conventions of communication are fulfilled, if the generated dialogue satisfies the protocol specification.

Propositional Linear Temporal Logic (PLTL)

- Syntax: We have the usual propositional connectives (such as negation and conjunction) and a number of temporal operators.
- Semantics: A *model* $\mathcal{M} = (\mathcal{T}, V)$ consists of a *frame* $\mathcal{T} = (T, <)$ and a *valuation* V mapping propositional letters to subsets of T. Here we take T to be a finite set of integers. Truth conditions:
  - p is true at point t iff $t \in V(p)$ (for propositional letters)
  - $\bigcirc\varphi$ "φ is true at the *next* point"
  - $\Diamond\varphi$ "φ is true at *some* future point"
  - $\Box\varphi$ " φ is true at *all* future points"
  - φ UNTIL ψ "ψ is true at some future point and φ *until* then"



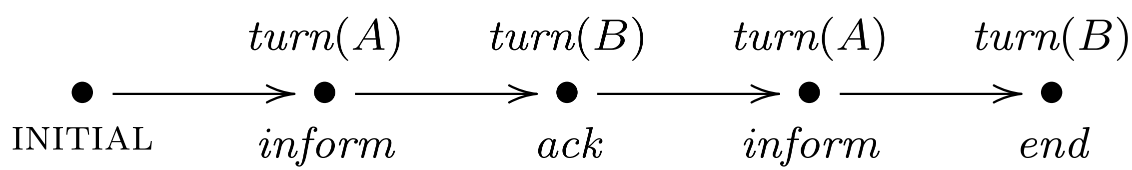
General Approach

- Specify protocols using PLTL formulas.
- Interpret dialogues as PLTL models.
- Whether or not a given dialogue $\mathcal{M}$ *conforms* to a given protocol φ can be verified using "model checking".

Models and Dialogues

Suppose the set of propositional letters includes the *performatives*, $turn(A)$ for every agent A, and the special symbol INITIAL.

Then every *dialogue* induces a *partial model* by fixing the frame and the valuation for these propositional letters. Example:



Now the problem of *conformance checking* can be described as follows:

- Given a partial model $\mathcal{M}$ (induced by a dialogue) and a formula φ (the specification of a protocol), is there a full model $\mathcal{M}'$ completing $\mathcal{M}$ such that φ is true at every point in $\mathcal{M}'$?

This problem is known as *generalised model checking* (if $\mathcal{M}$ is already a full model, then the above reduces to standard model checking).

Specifying Automata-based Protocols

- Recall the "continuous update protocol". We can model the state transition function as follows:

$$state(0) \wedge \bigcirc inform \rightarrow \bigcirc state(1)$$
$$state(1) \wedge \bigcirc ack \rightarrow \bigcirc state(2)$$
$$state(1) \wedge \bigcirc end \rightarrow \bigcirc state(3) \text{ etc.}$$

- Definition of initial and final states:

$$\textsc{initial} \leftrightarrow state(0)$$

$$\textsc{final} \leftrightarrow state(3) \vee state(4)$$
$$\textsc{final} \rightarrow \neg \bigcirc \top$$

- Still missing: How do we best specify the range of legal follow-ups for a given state?

Legality Conditions

- A first attempt to specify what are legal follow-ups from state 1:

  $$state(1) \rightarrow \bigcirc(ack \vee end)$$

  The problem with this approach is that generalised model checking will only succeed for *complete* dialogues.

- A better approach would be to use "weak" next-operators:

  $$state(1) \rightarrow \neg \bigcirc \neg(ack \vee end) \text{ etc.}$$

- Turn-taking rules can be specified in a similar fashion.

- Let φ_{cu} be the conjunction of all the above formulas. Then a (possibly incomplete) dialogue $\mathcal{M}$ is legal according to the protocol *iff* generalised model checking succeeds for φ_{cu} and $\mathcal{M}$.

- If we only want to succeed for complete dialogues, add:

  $$\text{NON-FINAL} \leftrightarrow state(0) \vee state(1) \vee state(2)$$

  $$\text{NON-FINAL} \rightarrow \bigcirc\top$$

Modelling Future Obligations

- Automata-based protocols cannot model *future obligations* such as "if you open an auction you will eventually have to close it again".
- Specifying above constraint as $(open \rightarrow \Diamond end)$ leads to similar problems as before (only complete dialogues considered legal).
  A better specification would be:

  $$open \;\rightarrow\; \textsc{pending} \;\wedge\; (\textsc{pending unless } end)$$

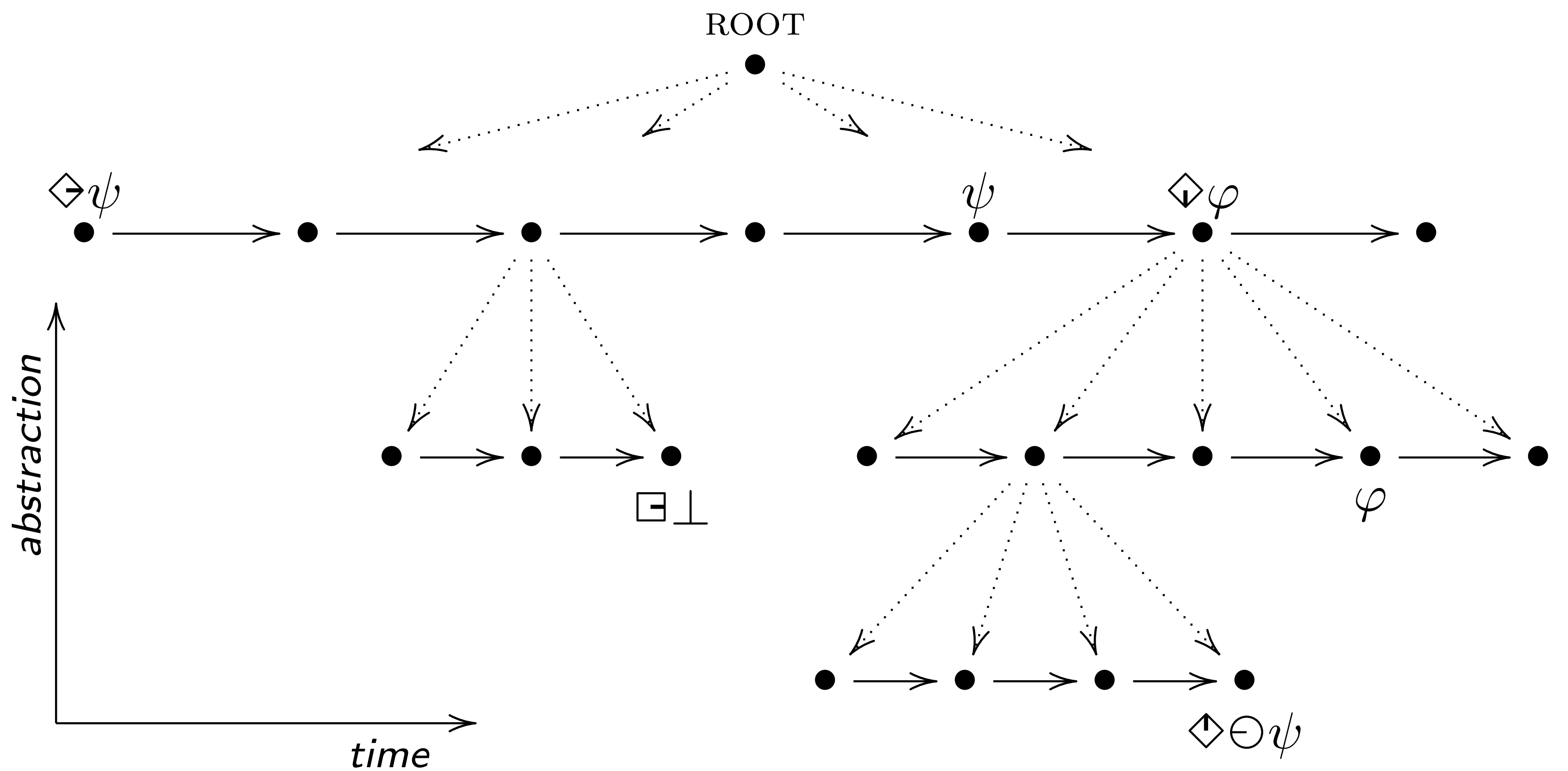
  where $\varphi \textsc{ unless } \psi \;=\; (\varphi \textsc{ until } \psi) \vee \Box\varphi$
- If we want to check that all obligations have been fulfilled, add:

  $$\textsc{pending} \rightarrow \bigcirc\top$$

Nested Protocols

- In practice, a multiagent system may specify a whole range of different protocols, and agents may use a combination of several of these during a communicative interaction.
- For instance, there may be different protocols for different types of auctions available, as well as a meta-protocol to jointly decide which of these auction protocols to use in a given situation.
- That is, we really need to be able to specify *nested* protocols.
- Such structures can be described using extended temporal logics also known as *modal logics of ordered trees* . . .

Modal Logics of Ordered Trees



Conclusions

- PLTL is a suitable logic for specifying agent communication protocols in the framework of the convention-based approach.
- Any combination of temporal constraints over utterances can be expressed in PLTL (expressive completeness).
- Conformance checking reduces to generalised model checking.
- We have identified modal logics of ordered trees as being suitable for modelling nested protocols.