Cofin 2003: Logic based development and verification of multi-agent systems

Reasoning about logic-based agent interaction protocols

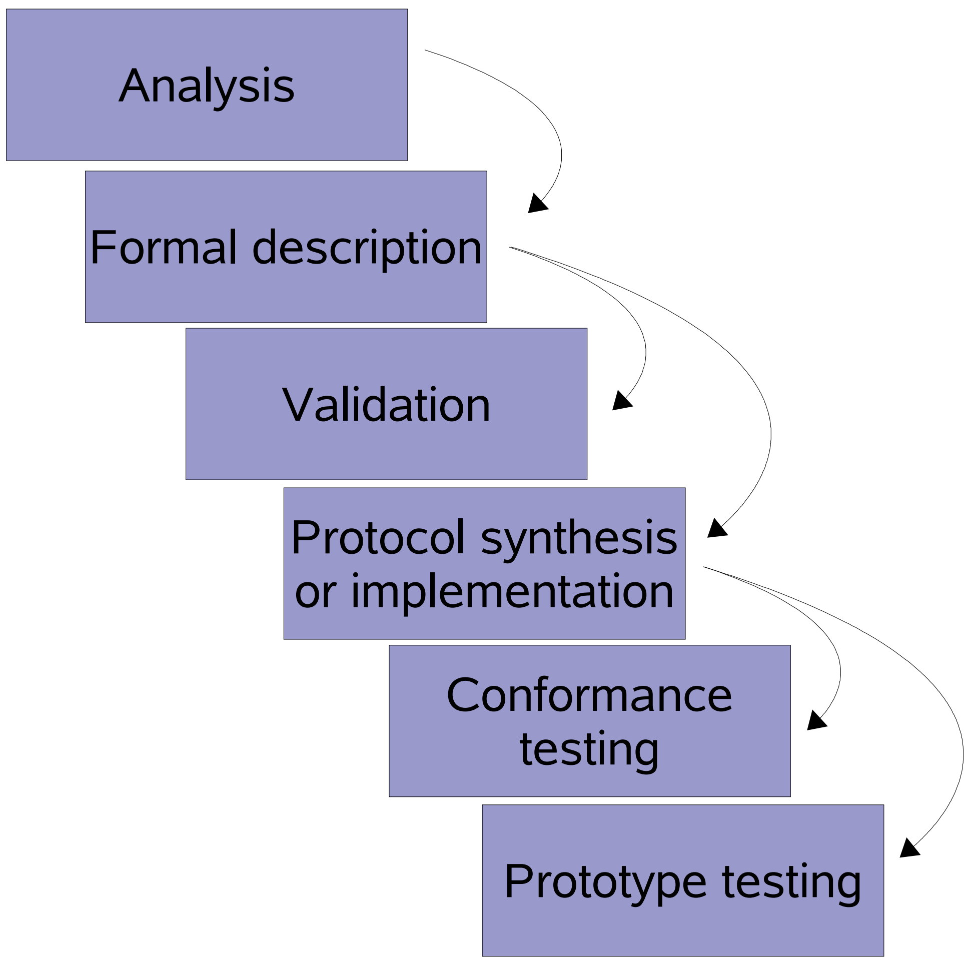
M. Baldoni, C. Baroglio, A. Martelli, V. Patti, C. Schifanella
Dipartimento di Informatica – Univ. degli Studi di Torino
C.so Svizzera, 185, I-10149 Torino (Italy)
http://www.di.unito.it/~argo
{baldoni,baroglio,mrt,patti}@di.unito.it

integration with DCaseLP in collaboration with
M. Martelli, V. Mascardi, I. Gangui
Dipartimento di Informatica e Scienze dell'Informazione
Univ. degli Studi di Genova

Protocols and MAS Engineering

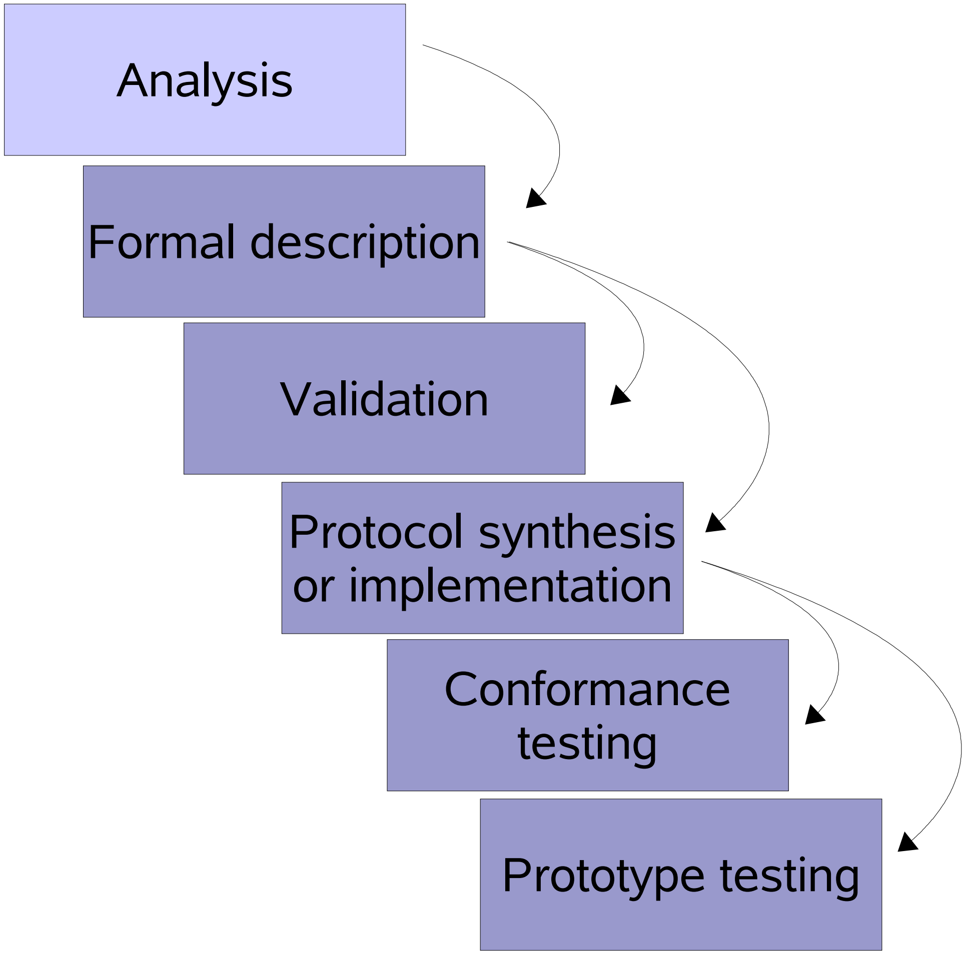
- Protocols as connective tissue of MAS
- AUML, a graphical high level modeling language for designing interactions (and protocols): abstract, does not specify the semantics of speech acts (the ACL ontology)
- Protocol implementation: no automatic translation, the abstract schema is to be completed
- Problem: verifying the conformance of an implementation to the AUML protocol
- Problem: verifying properties of the implementation

Protocols and MAS Engineering



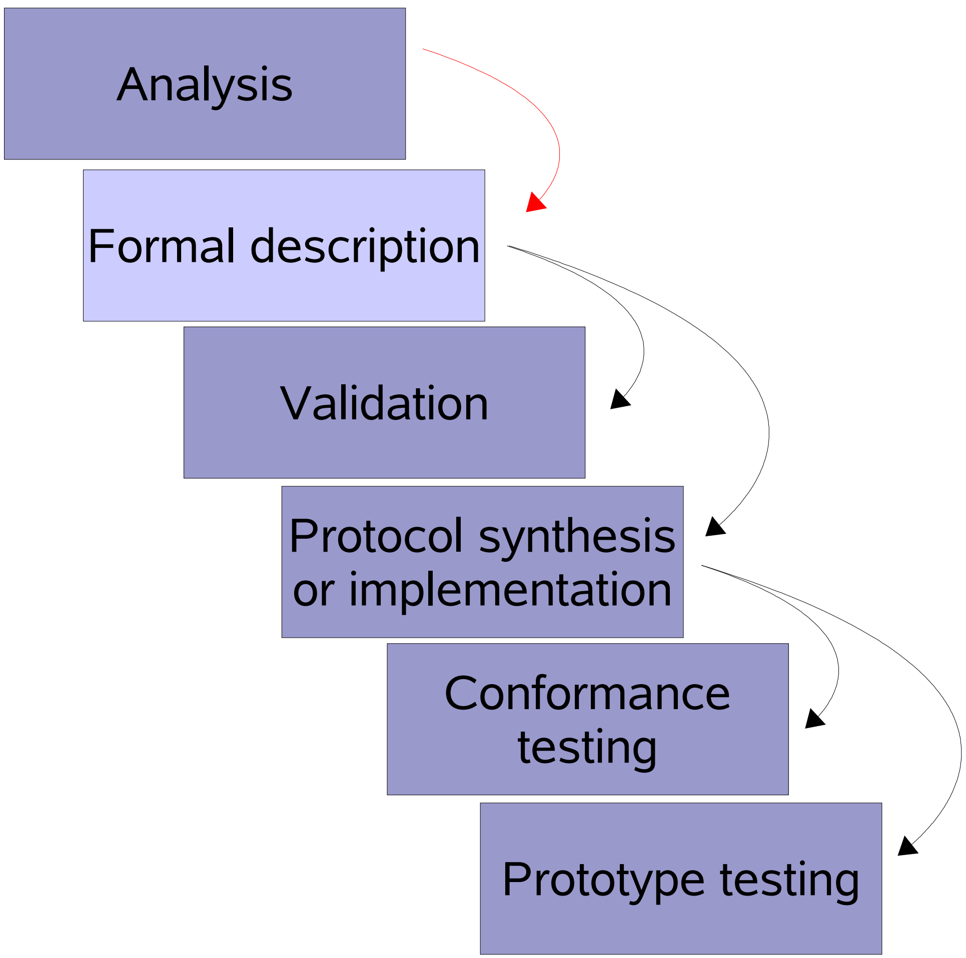
- The development process of an interaction protocol is known as interaction protocol engineering [Huget-Koning, 2003]
- Many stages are identified and described

Protocols and MAS Engineering



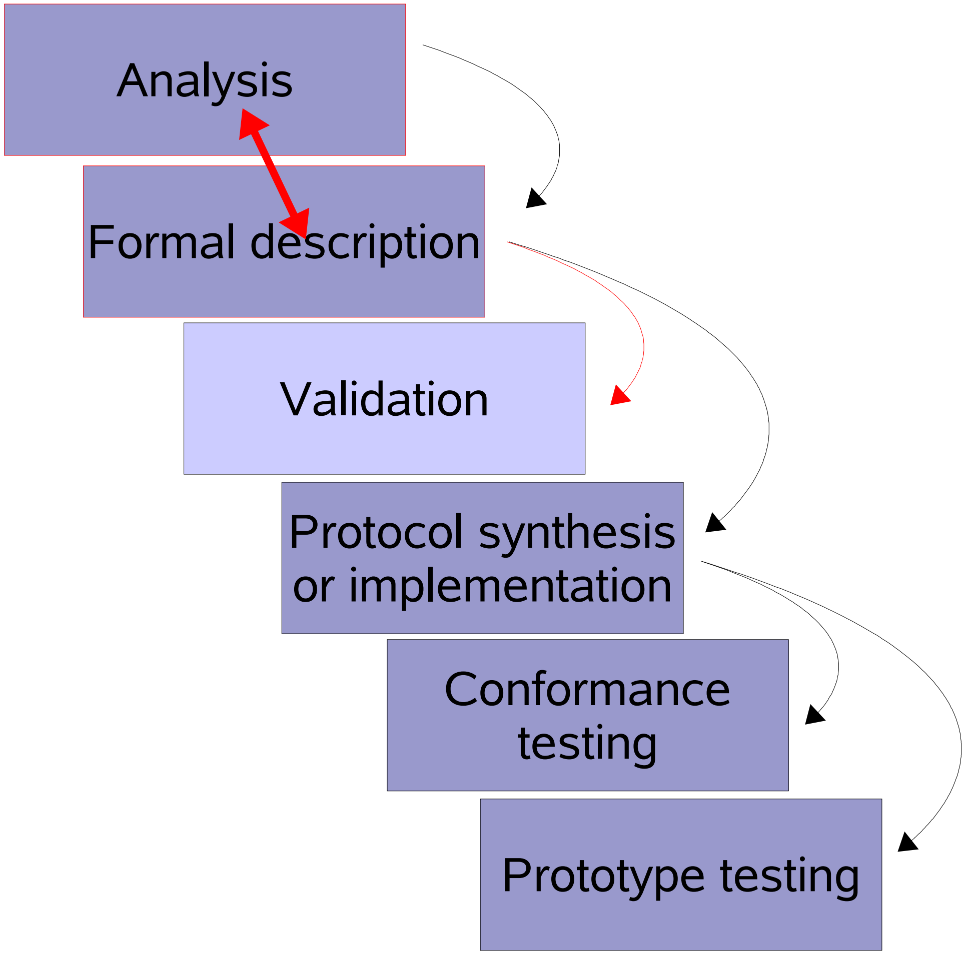
- ***Analysis***:
  all the features, that a protocol has to provide, are identified

Protocols and MAS Engineering



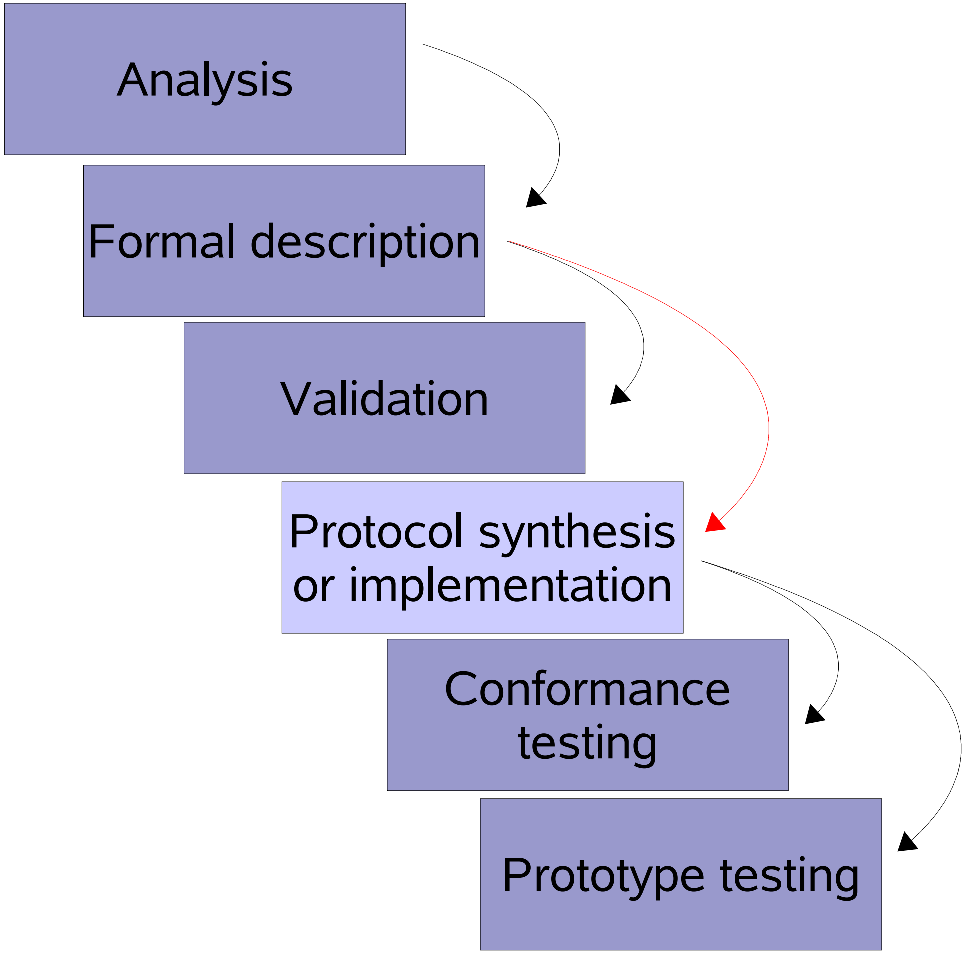
- ***Formal description***:
  a formal representation, in AUML or some other formalism, is given

Protocols and MAS Engineering



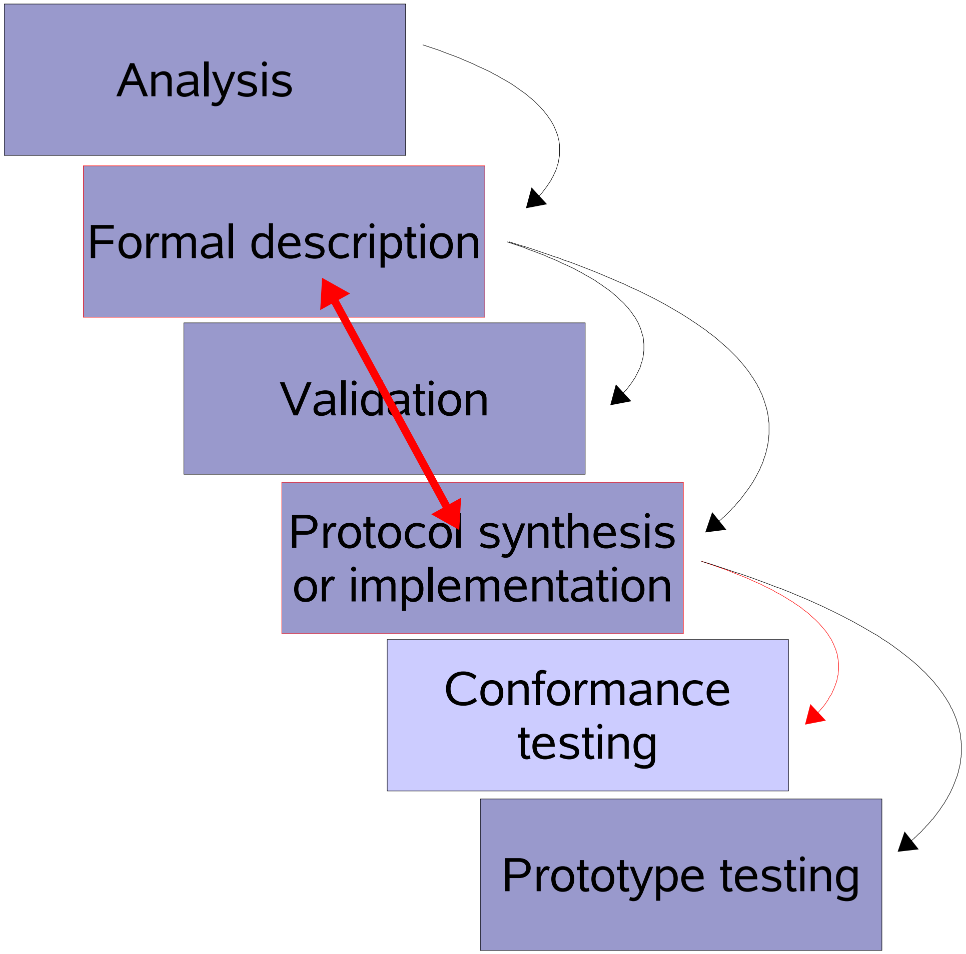
- ***Validation***:
  the formal description is validated w.r.t. the Analysis requirements (eg. model checking techniques)

Protocols and MAS Engineering



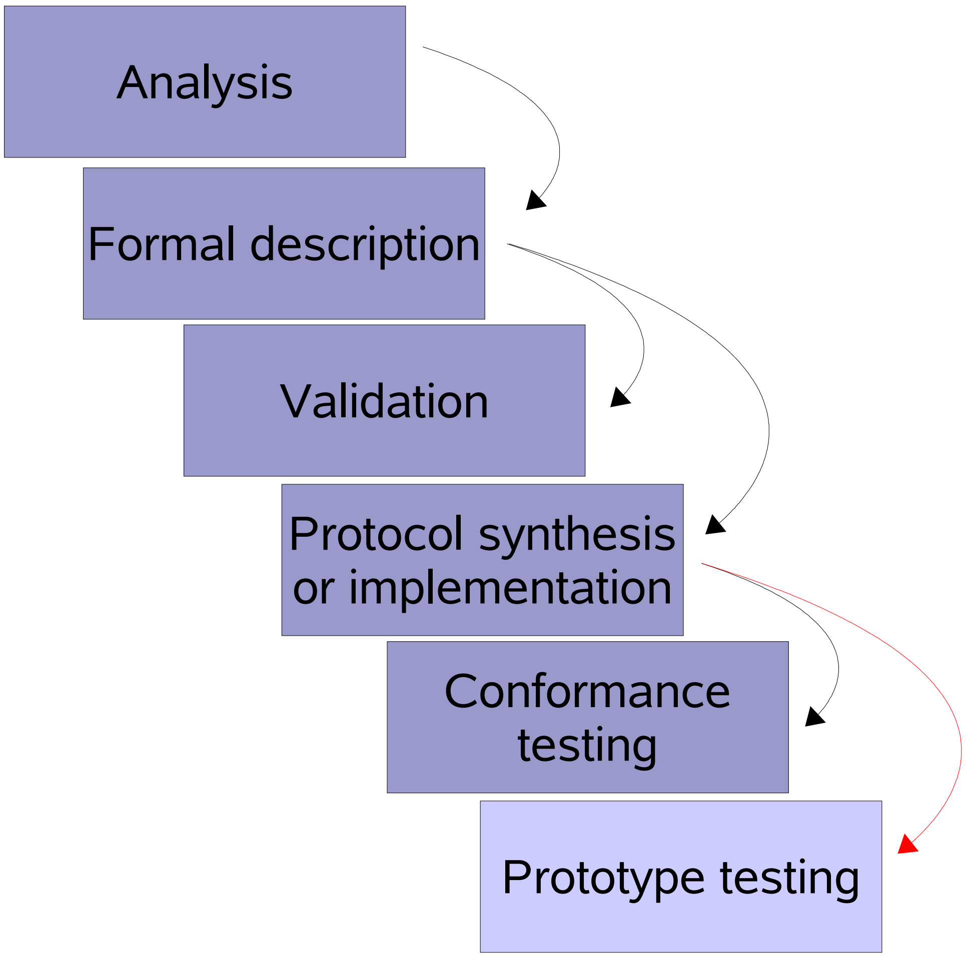
- Based on the obtained formal description, the protocol is implemented
- Alternatively:
  - A skeleton is produced in an automatic way and then it is completed by hand (in particular, by adding transitions semantics)
  - Implementation fully “by hand”

Protocols and MAS Engineering



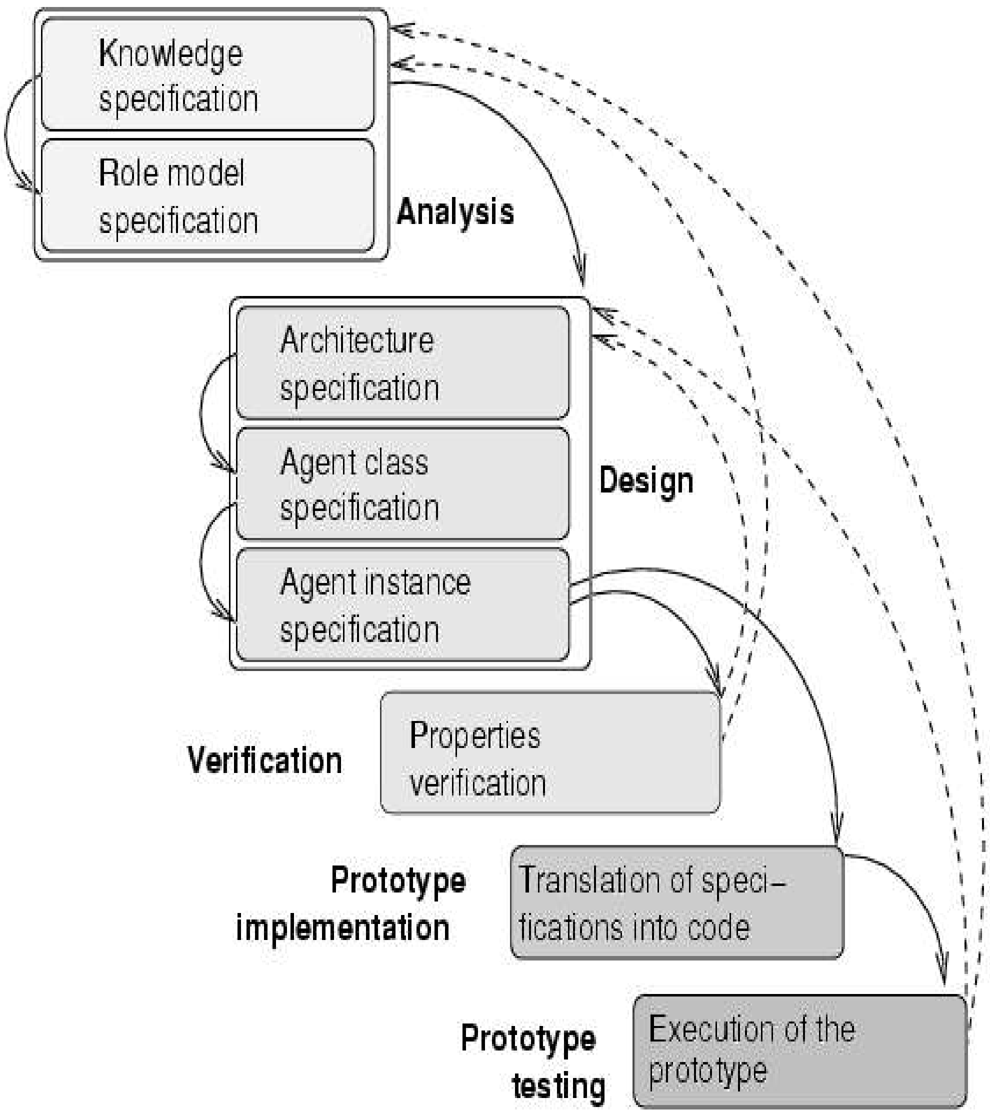
- To check if the operational version of the protocol still verifies the AUML specification
- Checking the properties of the operational version vs the properties of the formal description
- [Endriss, Maudet, Sadri, Toni, 2003, 2004]

Protocols and MAS Engineering



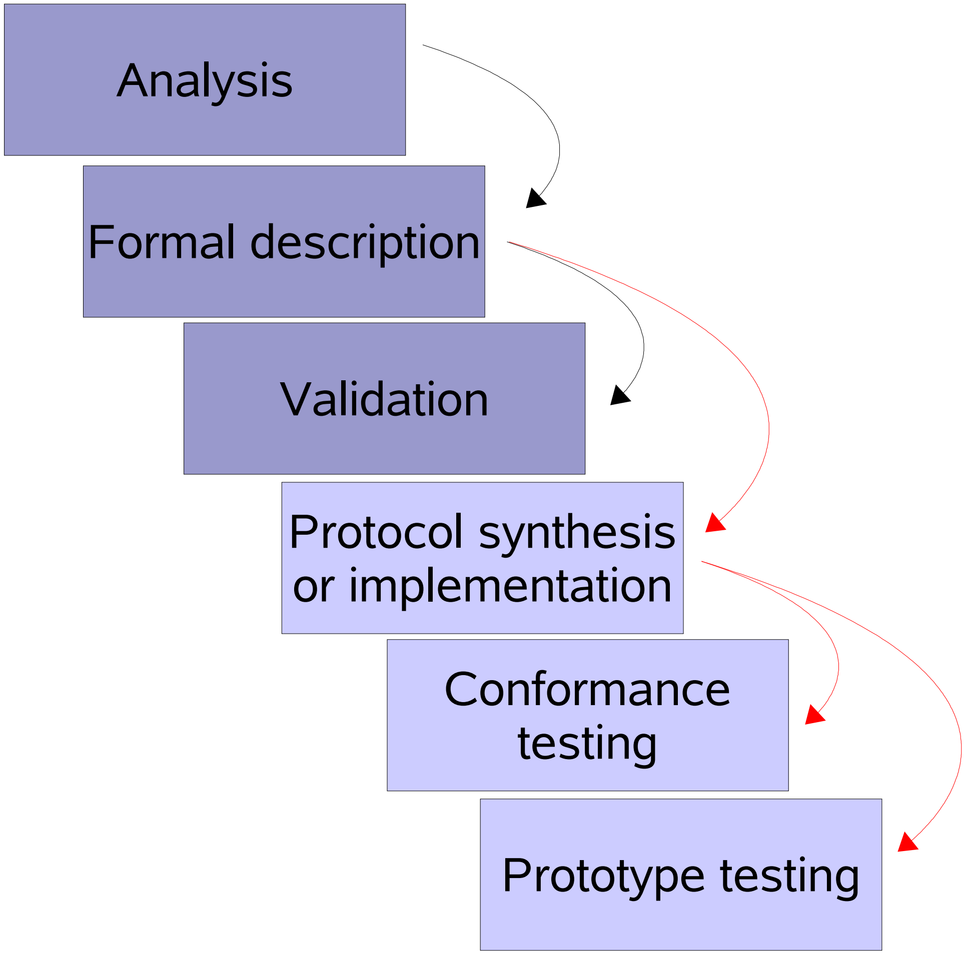
- Testing *by execution*
  - observe the running simulation (*animation*)
  - DCaseLP
- Testing *a priori*
  - Verify if a protocol implementation or a composition of protocol implementations satisfies some property

DcaseLP and its methodology



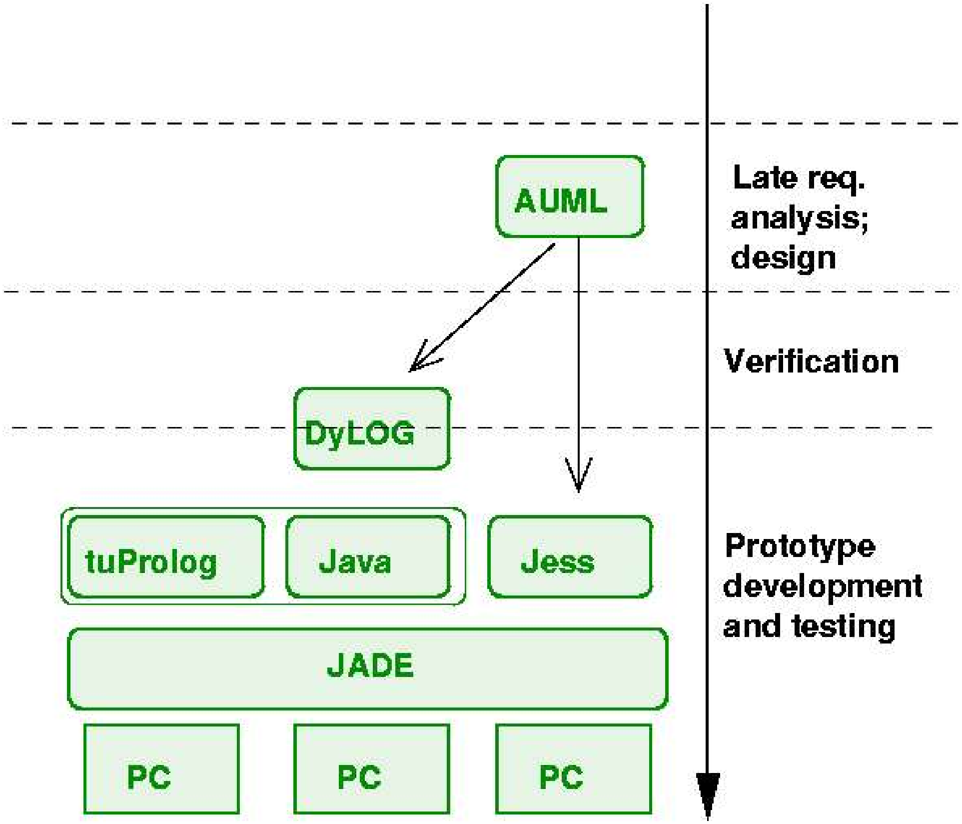
- Framework for the rapid prototyping of multi-agent systems
- It covers the engeneering stages from the requirement analysis to prototype execution
- It integrates a set of specification and implementation languages to model MAS

Integrating DyLOG in DCaseLP



- A logic language could help the designer, especially in these stages
- We propose to use DyLOG as a logic implementation language
- Because conformance verification of DyLOG protocols w.r.t. AUML protocols is quite natural
- Reasoning techniques can be applied for a *priori testing*

Integrating DyLOG in DCaseLP



- DyLOG can be used as an implementation language but it allows to verify properties of the written programs
- It is possible to verify the conformance in a natural way (as we will see soon)

Representing protocols in DyLOG

DyLOG...

- A logic language for specifying individual, communicating agents, situated in a multi agent context
- To perform hypothetical reasoning about the effects of conversations on the agents mental state
- In order to find conversation plans which are proved to respect protocols and to achieve some desired goal
- The semantic of the speech acts is specified based on mental states (taking the point of view of the agent)

DyLOG + CKit: overview

- A language to program agents, based on a *modal approach* for reasoning about actions and change
  - *Primitive actions*: preconditions and effects
  - *Sensing actions*: interaction with the world
  - *Prolog-like procedure definitions (complex actions)*: the agent's behaviour
- A domain description is used to refer to a set of primitive action definitions, a set of sensing action definitions, a set of complex action definitions, together with a set of initial observations.

DyLOG + Ckit: overview

$$DD^{agi} = (\Pi, Ckit^{agi}, S_0)$$

Π_C

a set of simple action laws to define the agent *speech acts* (inform, query, request, …)

Π_{CP}

a set of procedure axioms to specify the agent *conversation protocols*

Π_{Sget}

a set of sensing axioms to represent *messages from other agents*

- *speech acts and conversation policies are, as well, represented as primitive actions, sensing actions and procedure definitions of a DyLOG agent theory*

Inclusion axioms to represent procedures

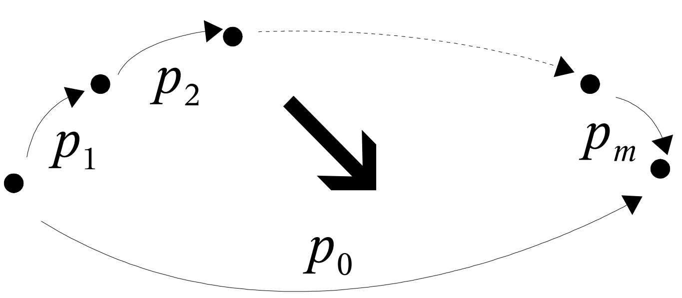
$$[p_0]\varphi \supset [p_1][p_2]\cdots[p_m]\varphi$$

Inclusion axiom

$$\langle p_0\rangle\varphi \subset \langle p_1\rangle\langle p_2\rangle\cdots\langle p_n\rangle\varphi$$

$$\Re_{p_0} \supseteq \Re_{p_1} \circ \Re_{p_0} \circ \cdots \circ \Re_{p_m}$$

Inclusion relation of accessibility relations

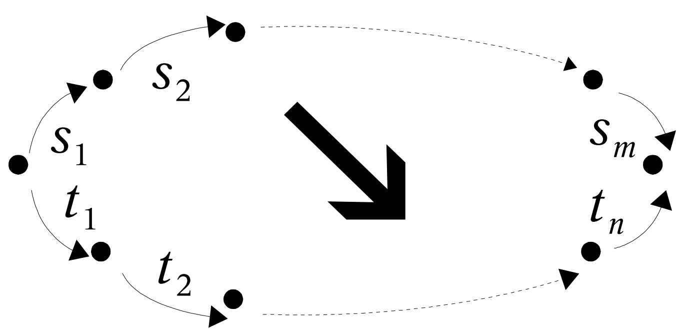


- Protocols are specified by means of inclusion axioms
- Kripke models for logics characterized by inclusion axioms satisfy the corrisponding inclusion properties

Inclusion axioms to represent procedures

$$t_1 t_2 \cdots t_n \rightarrow s_1 s_0 \cdots s_m$$

$$[t_1][t_2]\cdots[t_n]\varphi \supset [s_1][s_0]\cdots[s_m]\varphi$$



$$\Re_{t_1} \circ \Re_{t_2} \circ \cdots \circ \Re_{t_n} \supseteq \Re_{s_1} \circ \Re_{s_0} \circ \cdots \circ \Re_{s_n}$$

$p_0 \rightarrow p_1 p_2$

$p_0 \rightarrow \varepsilon$

$\langle p_0 \rangle \varphi \subset \langle p_1 \rangle \langle p_2 \rangle \varphi$

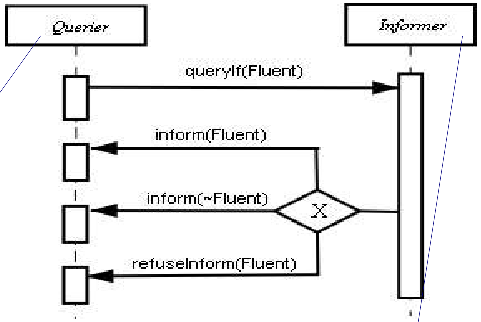
$\langle p_0 \rangle \varphi \subset \varphi$

- Fariñas del Cerro & Penttonen, 1988: Grammar logics
  - Modal logics defined on the basis of production rules of a grammar
  - For simulating the behaviour of grammas
  - Undecidability result
- Baldoni, Giordano, Martelli, 1998; Baldoni, 1998 e 2000
  - Tableaux calculus
  - (Un)Decidability results for subclasses and superclasses (incestual modal logics)

Representing protocols in DyLOG

- Agents have a *subjective perception* of communication with the others, then an agent represents a protocol as one of *its* (conversation) *policies*
- *Policies* are represented by a set of *inclusion axioms* of the form:

$$\langle p_0 \rangle \varphi \subset \langle p_1 \rangle \langle p_2 \rangle \cdots \langle p_n \rangle \varphi$$



$$\begin{aligned}
&\langle \mathsf{yes_no_query}_Q(Self, Other, Fluent)\rangle\varphi \subset \\
&\quad \langle \mathsf{queryIf}(Self, Other, Fluent); \\
&\quad \mathsf{get_answer}(Self, Other, Fluent)\rangle\varphi
\end{aligned}$$

$$\begin{aligned}
&[\mathsf{get_answer}(Self, Other, Fluent)]\varphi \equiv \\
&\quad [\mathsf{inform}(Other, Self, Fluent) \cup \\
&\quad \mathsf{inform}(Other, Self, \neg Fluent) \cup \\
&\quad \mathsf{refuseInform}(Other, Self, Fluent)]\varphi
\end{aligned}$$

$$\begin{aligned}
&\langle \mathsf{yes_no_query}_I(Self, Other, Fluent)\rangle\varphi \subset \\
&\quad \langle \mathsf{get_start}(Self, Other, Fluent); \\
&\quad \mathcal{B}^{Self} Fluent?; \mathsf{inform}(Self, Other, Fluent)\rangle\varphi \\
&\langle \mathsf{yes_no_query}_I(Self, Other, Fluent)\rangle\varphi \subset \\
&\quad \langle \mathsf{get_start}(Self, Other, Fluent); \\
&\quad \mathcal{B}^{Self} \neg Fluent?; \mathsf{inform}(Self, Other, \neg Fluent)\rangle\varphi \\
&\langle \mathsf{yes_no_query}_I(Self, Other, Fluent)\rangle\varphi \subset \\
&\quad \langle \mathsf{get_start}(Self, Other, Fluent); \\
&\quad \mathcal{U}^{Self} Fluent?; \mathsf{refuseInform}(Self, Other, Fluent)\rangle\varphi \\
&[\mathsf{get_start}(Self, Other, Fluent)]\varphi \equiv \\
&\qquad [\mathsf{queryIf}(Other, Self, Fluent)]\varphi
\end{aligned}$$

DyLOG + Ckit: overview

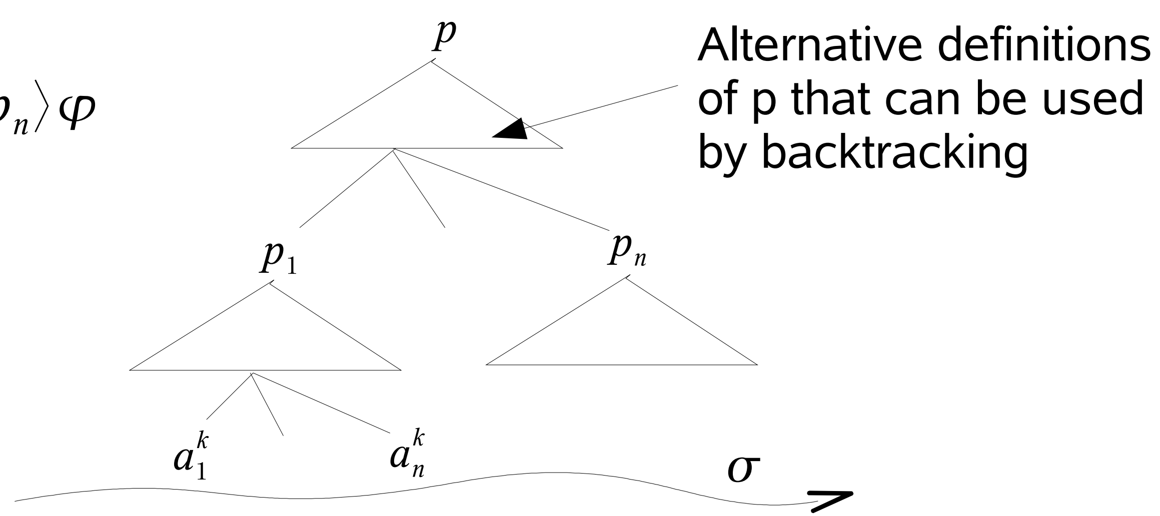
- Given a domain description, we can reason about it by means of *existential queries*:

$$(\Pi, CKit^{ag_i}, S_0) \vdash \langle p \rangle Fs \; w.a. \; \sigma$$

  - *p* is an interaction protocol
  - We look for a conversation, which is an *instance* of the protocol described by *p*, after which the condition *Fs* holds

$$\langle p \rangle \varphi \subset \langle p_1 \rangle \langle p_2 \rangle \cdots \langle p_n \rangle \varphi$$

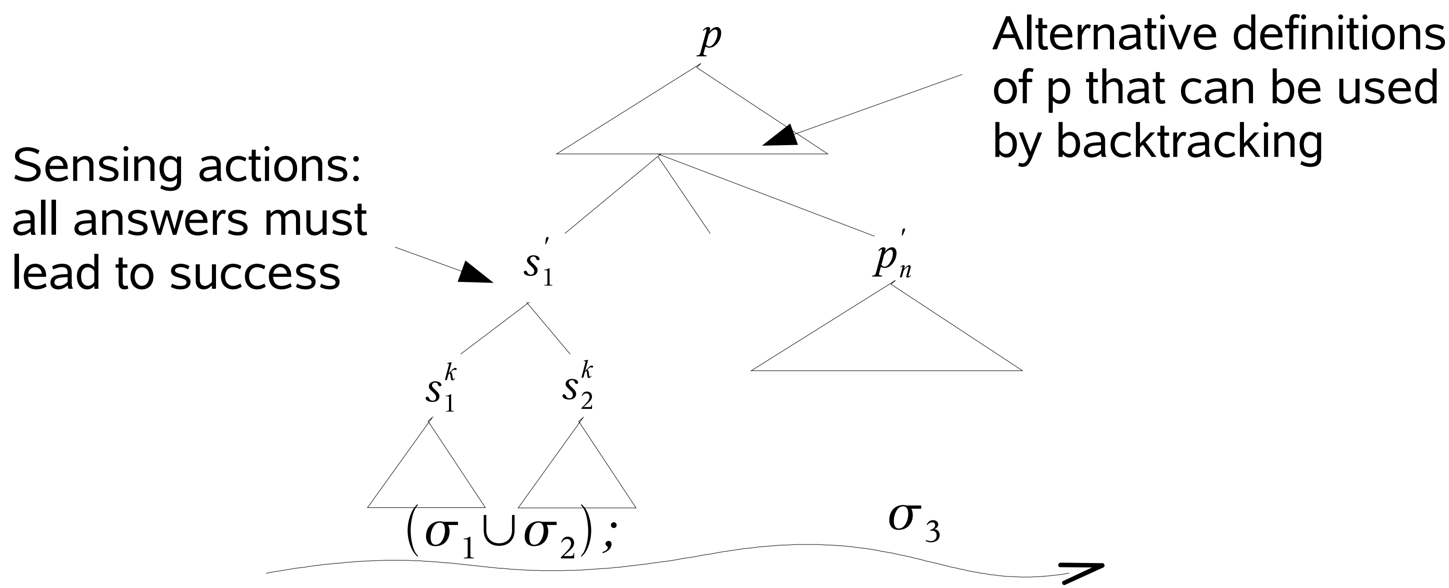
$$p \rightarrow p_1 \, p_2 \cdots p_n$$



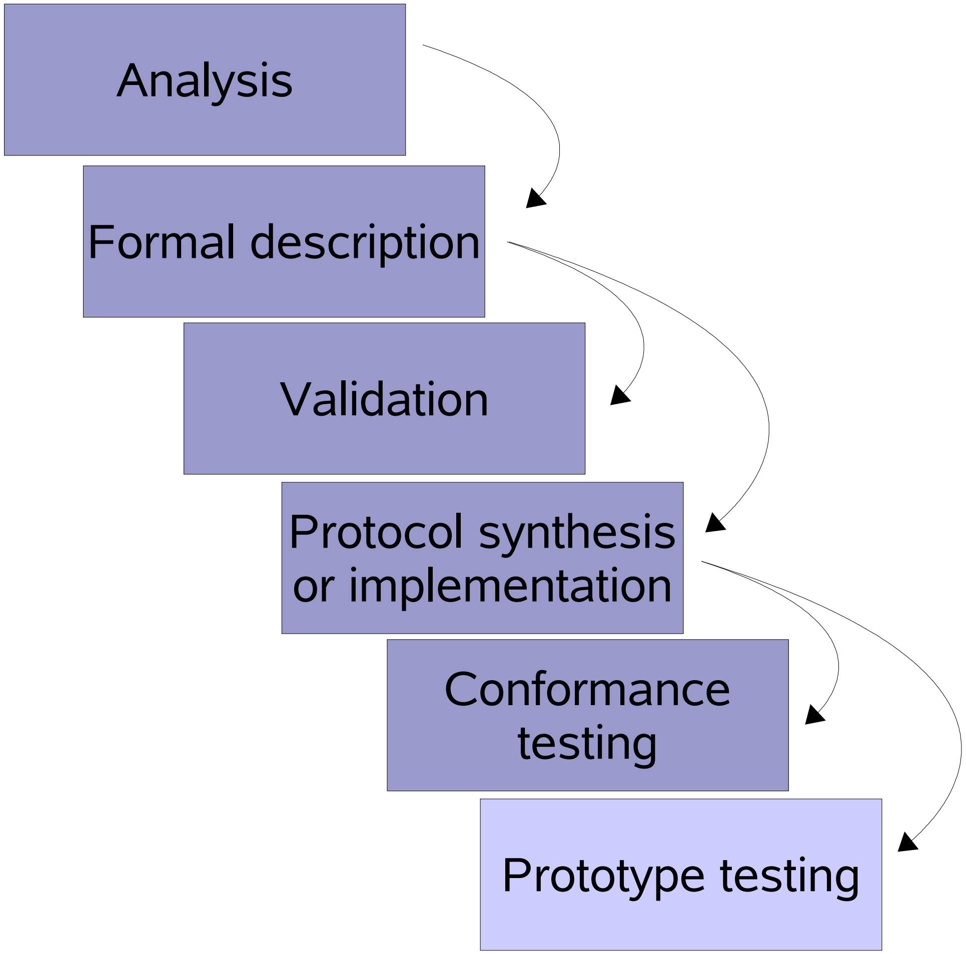
DyLOG + Ckit: overview

- We treat get-message actions as sensing actions, whose outcome cannot be known at planning time (*conditional plans vs linear plans*)
- Goal directed proof procedure, based on negation as failure (deailing with persistency) [ICTCS 2003]

$$(\Pi, CKit^{ag_i}, S_0) \vdash \langle p_m \rangle Fs \; w.a.\; \sigma$$



Prototype testing: testing a priori



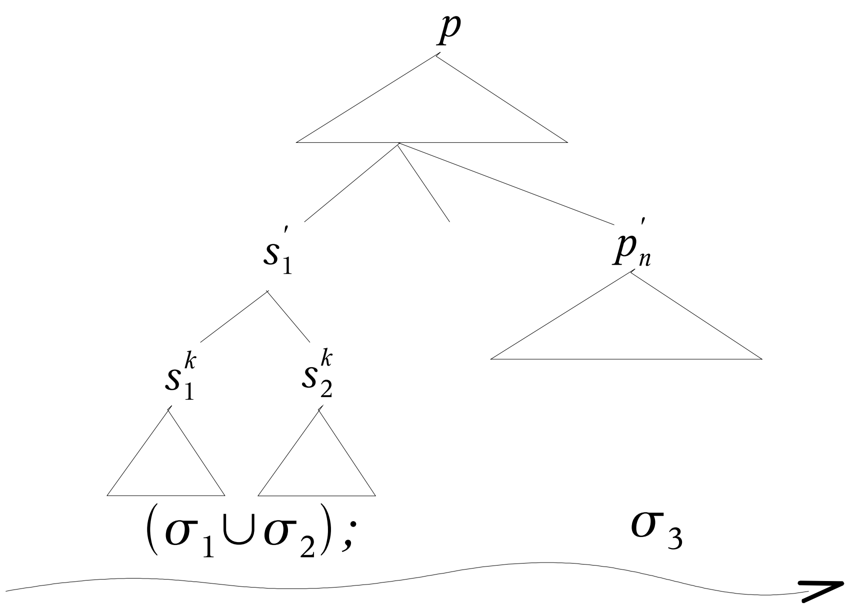
- Verify if a protocol implementation or a composition of protocol implementations satisfies some property

DyLOG + Ckit: testing a priori

- Look for a protocol that has one possible execution, after which the service provider does not know the customer's credit card number, and a reservation has been taken

$$\langle search\_service(restaurant, Protocol)\ ;\ Protocol(customer, service, time)\rangle$$
$$(\mathcal{B}^{customer} \neg \mathcal{B}^{service} cc\_number \wedge \mathcal{B}^{customer} reservation(time))$$

Existential query!!

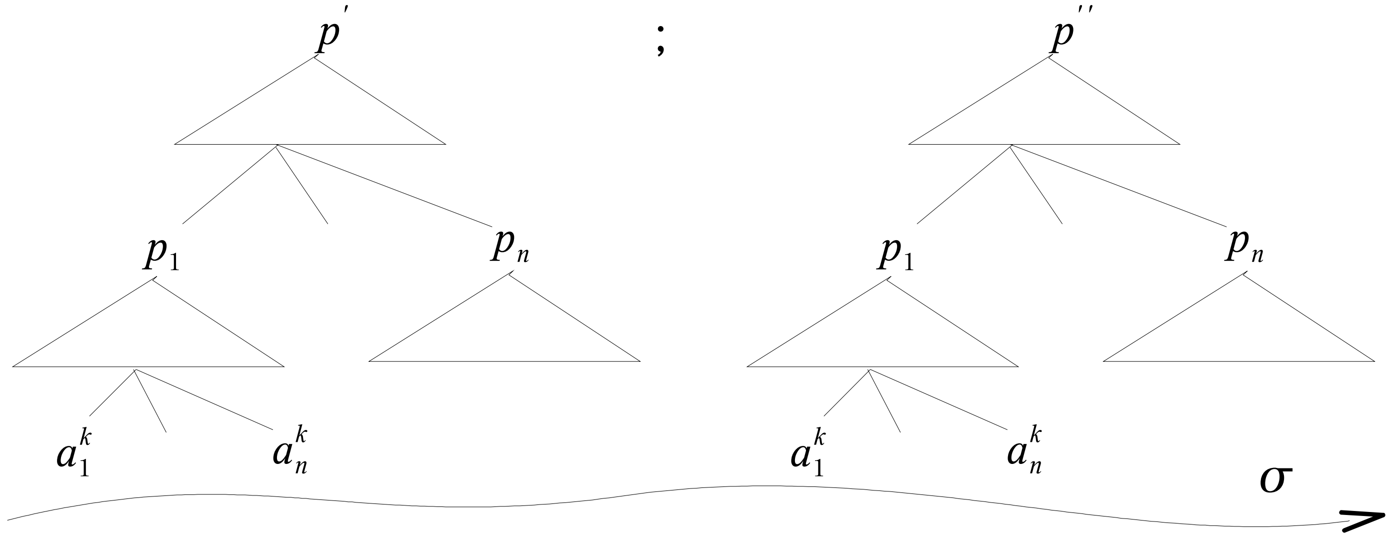


DyLOG + Ckit: testing a priori

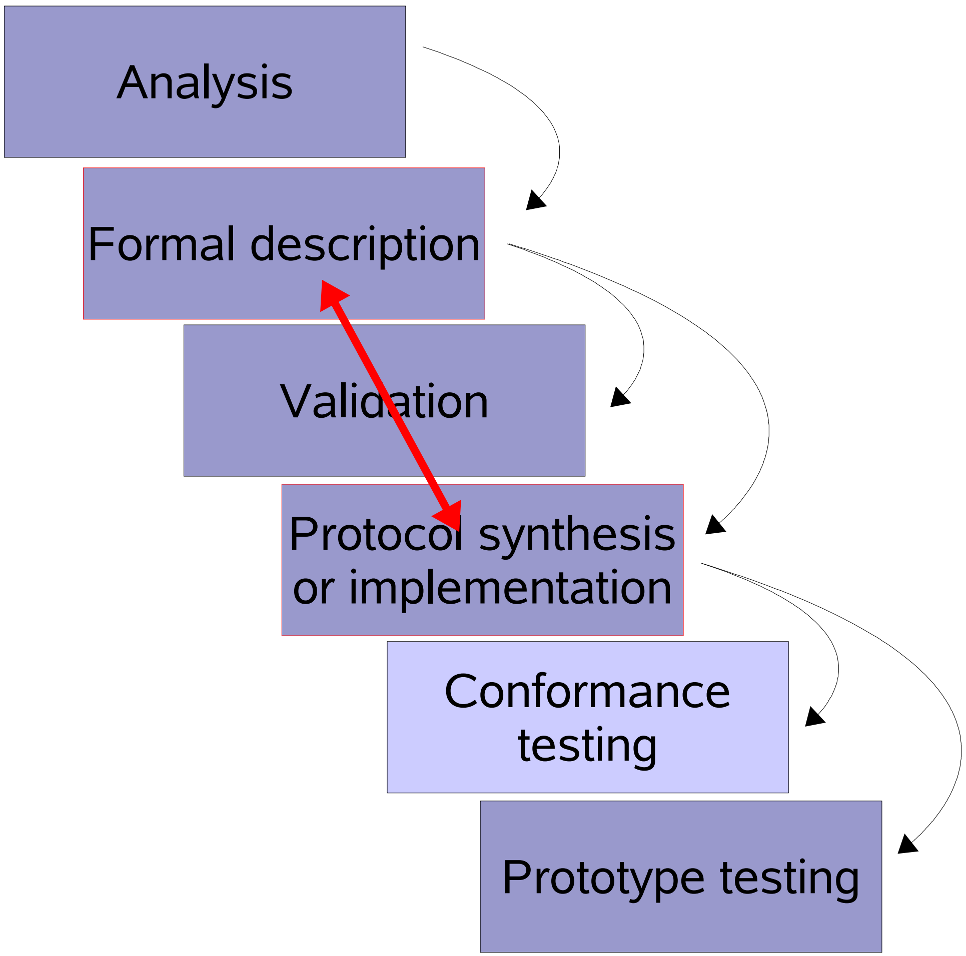
- Is it possible to compose the interaction so to reserve a table for dinner and to book a ticket for a movie, exploiting a promotion?

$$\langle \mathsf{reserv\_rest\_1}_C(customer, restaurant, dinner) \ ; \ \mathsf{reserv\_cinema\_1}_C(customer, cinema, movie)\rangle$$
$$(\mathcal{B}^{customer} cinema\_promo \wedge \mathcal{B}^{customer} reservation(dinner) \wedge \mathcal{B}^{customer} reservation(movie) \wedge \mathcal{B}^{customer} \mathcal{B}^{C} ft\_number)$$

Existential query!!

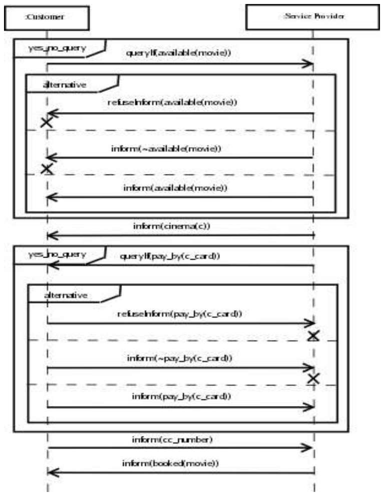


The conformance testing



- To check if the operational version of the protocol still verifies the AUML specification
- Checking the properties of the operational version vs the properties of the formal description
- [Endriss, Maudet, Sadri, Toni, 2003, 2004]

Verifying the conformance



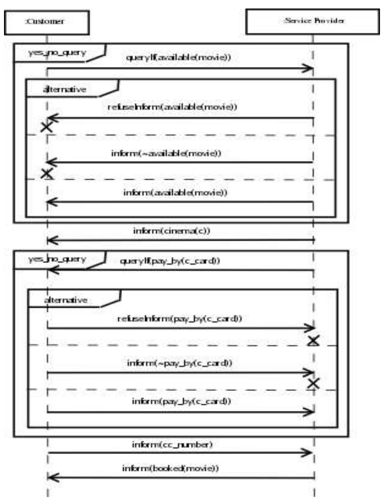
AUML interaction diagram

To check that an agent never performs a dialogue move that is not foreseen by the AUML specification

DyLOG implementation

$$
\begin{aligned}
&\langle \mathsf{reserv\_rest\_1}_C(Self, Service, Time)\rangle\varphi \subset \\
&\quad \langle \mathsf{yes\_no\_query}_Q(Self, Service, available(Time)) \ ; \\
&\qquad \mathcal{B}^{Self} available(Time)? \ ; \\
&\qquad \mathsf{get\_info}(Self, Service, reservation(Time)) \ ; \\
&\qquad \mathsf{get\_info}(Self, Service, cinema\_promo) \ ; \\
&\qquad \mathsf{get\_info}(Self, Service, ft\_number)\rangle\varphi
\end{aligned}
$$

The conformance testing w.r.t. DyLOG implementation



AUML interaction diagram

extract

Formal Language: it represents all possible sequences of dialogue acts on the basis of the AUML sequence diagram

Different sets of possible dialogues depending on the level of abstraction from the agent mental state

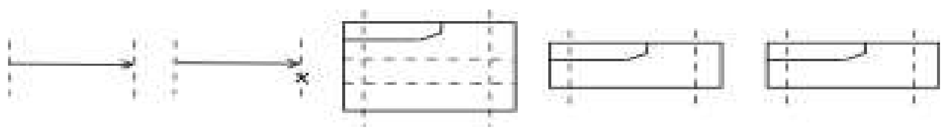
Sequences corresponding to all possible dialogues allowed by the implementation

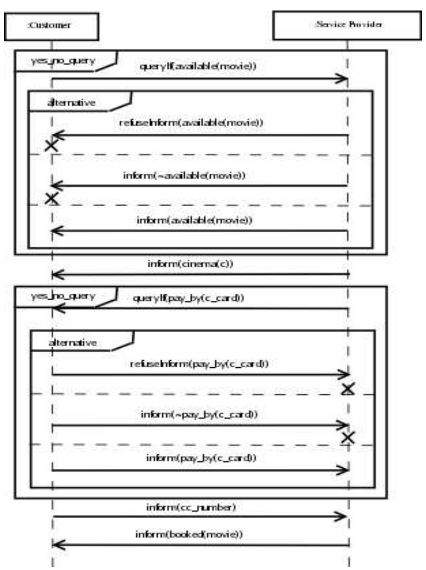
DyLOG implementation

$$
\begin{array}{l}
\langle \text{reserv\_rest\_1}_C(Self, Service, Time)\rangle\varphi \subset \\
\quad \langle \text{yes\_no\_query}_Q(Self, Service, available(Time)) \,; \\
\quad\quad \mathcal{B}^{Self} available(Time)? \,; \\
\quad\quad \text{get\_info}(Self, Service, reservation(Time)) \,; \\
\quad\quad \text{get\_info}(Self, Service, cinema\_promo) \,; \\
\quad\quad \text{get\_info}(Self, Service, ft\_number))\rangle\varphi
\end{array}
$$

estract

Traslating AUML into linear grammars



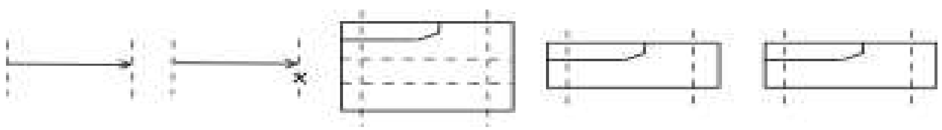


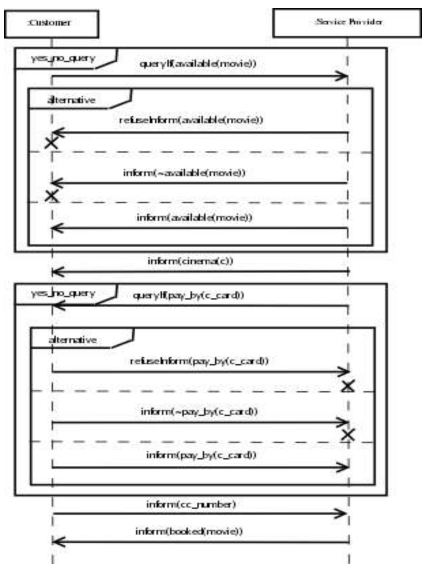
$$
\begin{aligned}
Q_0 &\longrightarrow \mathsf{queryIf}(cus, sp, available(movie))\; Q_1\\
Q_1 &\longrightarrow Q_2 \mid Q_4 \mid Q_6\\
Q_2 &\longrightarrow \mathsf{refuseInform}(cus, sp, available(movie)) Q_3\\
Q_3 &\longrightarrow \varepsilon\\
Q_4 &\longrightarrow \mathsf{inform}(sp, cus, \neg available(movie)) Q_5\\
Q_5 &\longrightarrow \varepsilon\\
Q_6 &\longrightarrow \mathsf{inform}(sp, cus, available(movie))\; Q_7\\
Q_7 &\longrightarrow Q_8\\
Q_8 &\longrightarrow \mathsf{inform}(sp, cus, cinema(c))\; Q_9\\
Q_9 &\longrightarrow \mathsf{queryIf}(sp, cus, pay\_by(c\_card))\; Q_{10}\\
Q_{10} &\longrightarrow Q_{11} \mid Q_{13} \mid Q_{15}\\
Q_{11} &\longrightarrow \mathsf{refuseInform}(cus, sp, available(movie)) Q_{12}\\
Q_{12} &\longrightarrow \varepsilon\\
Q_{13} &\longrightarrow \mathsf{inform}(sp, cus, \neg available(movie)) Q_{14}\\
Q_{14} &\longrightarrow \varepsilon\\
Q_{15} &\longrightarrow \mathsf{inform}(sp, cus, available(movie))\; Q_{16}\\
Q_{16} &\longrightarrow Q_{17}\\
Q_{17} &\longrightarrow \mathsf{inform}(cus, sp, cc\_number)\; Q_{18}\\
Q_{18} &\longrightarrow \mathsf{inform}(sp, cus, booked(movie)) Q_{19}\\
Q_{19} &\longrightarrow \varepsilon
\end{aligned}
$$

- In [BBMPS05, submitted] we present an algorithm to traslate AUML 2.0 *operators message, alternative*, *loop*, and *sub-protocol* into a formal linear grammar
- The language generated by the grammar represents *all allowed interactions* between agents

$$L(G_{p_{AUML}})$$

Traslating AUML into linear grammars





$$
\begin{aligned}
Q_0 &\longrightarrow \text{queryIf}(cus, sp, available(movie))\ Q_1\\
Q_1 &\longrightarrow Q_2 \mid Q_4 \mid Q_6\\
Q_2 &\longrightarrow \text{refuseInform}(cus, sp, available(movie)) Q_3\\
Q_3 &\longrightarrow \varepsilon\\
Q_4 &\longrightarrow \text{inform}(sp, cus, \neg available(movie)) Q_5\\
Q_5 &\longrightarrow \varepsilon\\
Q_6 &\longrightarrow \text{inform}(sp, cus, available(movie))\ Q_7\\
Q_7 &\longrightarrow Q_8\\
Q_8 &\longrightarrow \text{inform}(sp, cus, cinema(c))\ Q_9\\
Q_9 &\longrightarrow \text{queryIf}(sp, cus, pay\_by(c\_card))\ Q_{10}\\
Q_{10} &\longrightarrow Q_{11} \mid Q_{13} \mid Q_{15}\\
Q_{11} &\longrightarrow \text{refuseInform}(cus, sp, available(movie)) Q_{12}\\
Q_{12} &\longrightarrow \varepsilon\\
Q_{13} &\longrightarrow \text{inform}(sp, cus, \neg available(movie)) Q_{14}\\
Q_{14} &\longrightarrow \varepsilon\\
Q_{15} &\longrightarrow \text{inform}(sp, cus, available(movie))\ Q_{16}\\
Q_{16} &\longrightarrow Q_{17}\\
Q_{17} &\longrightarrow \text{inform}(cus, sp, cc\_number)\ Q_{18}\\
Q_{18} &\longrightarrow \text{inform}(sp, cus, booked(movie)) Q_{19}\\
Q_{19} &\longrightarrow \varepsilon
\end{aligned}
$$

- Proposition 1:
  The set of conversation allowed by an AUML sequence diagram is a regular language
- Proof: The algorithm produces a right linear grammar.

$$L(G_{p_{AUML}})$$

Regular language

Different degree of testing conformance

Agent Conformance

$$\Sigma(S_0)=\{\sigma:(\Pi, CKit^{ag_i}, S_0)\vdash\langle p_m\rangle Fs\ w.a.\ \sigma\}$$

different levels
of abstraction from
the agent mental state

Protocol Conformance

$$p_0 \rightarrow p_1\, p_2$$

$$p_0 \rightarrow \varepsilon$$

structural conformance

$$\langle p_0\rangle\varphi\subset\langle p_1\rangle\langle p_2\rangle\varphi$$

$$\langle p_0\rangle\varphi\subset\varphi$$

Agent Strong Conformance

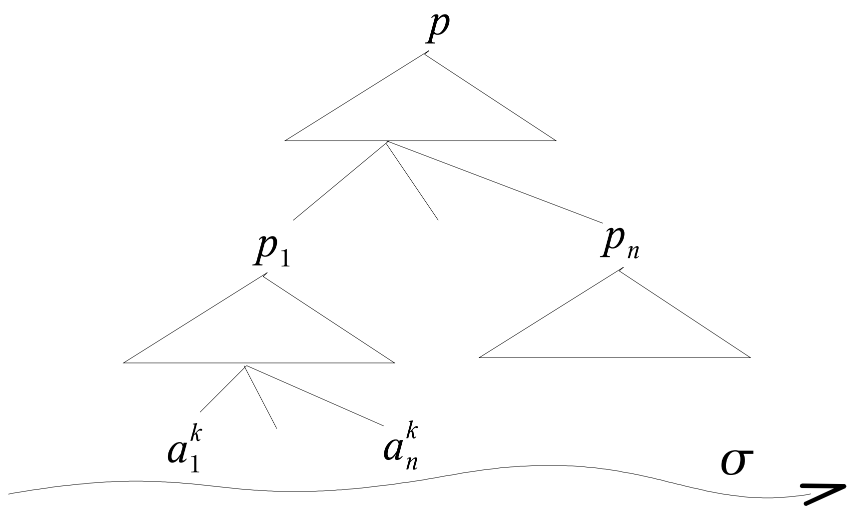
$$\cup\,\Sigma(S)=\{\sigma:(\Pi, CKit^{ag_i}, S)\vdash\langle p_m\rangle Fs\ w.a.\ \sigma\}$$

Agent (strong) conformance

- *Agent conformance*: every conversation σ, *instance* of the protocol implementation is also generated by the linear grammar that represents the AUML diagram

$$\Sigma(S_0) \subseteq L(G_{p_{AUML}})$$

  - *where* $\Sigma(S_0) = \{\sigma : (\Pi, CKit^{ag_i}, S_0) \vdash \langle p_m \rangle Fs \; w.a. \; \sigma\}$
  - *It depends on the agent initial state*!

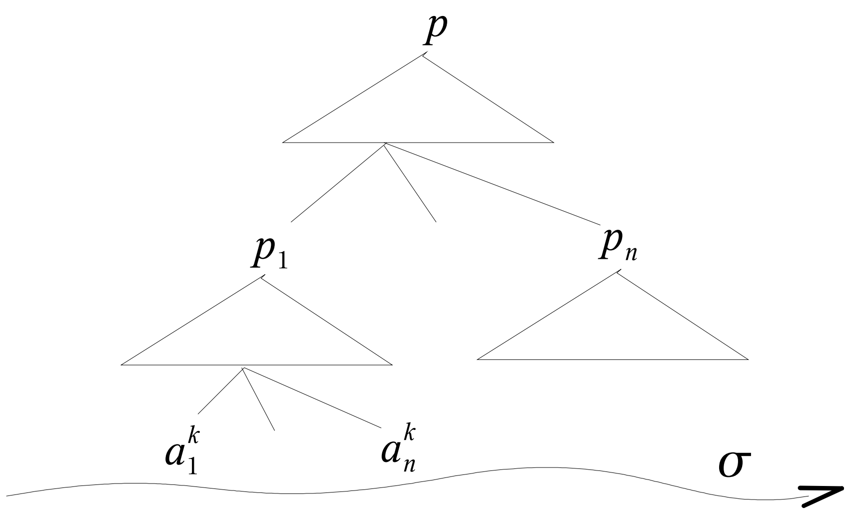


$$\sigma \in L(G_{p_{AUML}})$$

Agent strong conformance

- *Agent strong conformance*: for every initial state S, the above definition holds

$$\cup_S \Sigma(S) \subseteq L(G_{p_{AUML}})$$



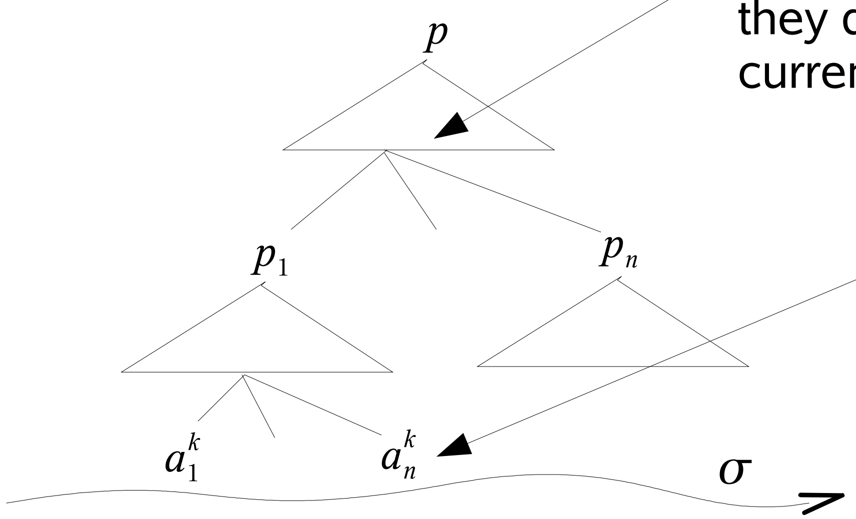
For every possible initial state!

$$\sigma \in L(G_{p_{AUML}})$$

Protocol conformance

- However, a better notion of conformance should require that a DyLOG implementation is conformant w.r.t. an AUML sequence diagram *independently* of the semantics of speech acts!

The derivation depends on the tests inside the protocol implementation, that, in turn, they depends on the current agent's mental state



The derivation depends on the preconditions of speech acts that are tested on the current agent's mental state

The derivation depends on the semantics of speech acts

Protocol conformance

- It is necessary to provide a sort of “structural” notion of conformance
- The idea is to define a *context-free grammar* from the DyLOG implementation, exploiting the natural interpretation of inclusion axioms as *rewriting rules*

$$\langle p_0 \rangle \varphi \subset \langle p_1 \rangle \langle p_2 \rangle \cdots \langle p_n \rangle \varphi$$

$$p_0 \rightarrow p_1 \, p_2 \cdots p_n$$

$$\begin{array}{l}
\langle \mathsf{get_cinema_ticket}_C(cus, sp, movie)\rangle \varphi \subset \\
\quad \langle \mathsf{yes_no_query}_Q(cus, sp, available(movie)); \\
\quad \mathcal{B}^{cus} available(movie)?; \mathsf{get_info}(cus, sp, cinema(c)); \\
\quad \mathsf{yes_no_query}_I(cus, sp, pay_by(credit_card)); \\
\quad \mathcal{B}^{cus} pay_by(credit_card)?; \mathsf{inform}(cus, sp, cc_number); \\
\quad \mathsf{get_info}(cus, sp, booked(movie))\rangle \varphi
\end{array}$$

$$\begin{array}{l}
\mathsf{get_cinema_ticket}_C(cus, sp, movie) \longrightarrow \\
\quad \mathsf{yes_no_query}_Q(cus, sp, available(movie)) \\
\quad \mathsf{get_info}(cus, sp, cinema(c)) \\
\quad \mathsf{yes_no_query}_I(cus, sp, pay_by(credit_card)) \\
\quad inform(cus, sp, cc_number) \\
\quad \mathsf{get_info}(cus, sp, booked(movie))
\end{array}$$

Protocol conformance

- The language generated by the context-free grammar $G_{p_{DyLOG}}$ so defined represents all possible sequences of speech acts allowed by the DyLOG implementation independently of the evolution of the agent mental state

- *Protocol conformance*: all possible sequences of speech acts allowed by the DyLOG implementation is also generated by the grammar that represents the AUML diagram

$$L(G_{p_{DyLOG}}) \subseteq L(G_{p_{AUML}})$$

Different degree of testing conformance

Agent Conformance

$$\Sigma(S_0)=\{\sigma:(\Pi, CKit^{ag_i}, S_0) \vdash \langle p_m\rangle Fs\ w.a.\ \sigma\}$$

IMPLIES

different levels
of abstraction from
the agent mental state

Protocol Conformance

IMPLIES

Agent Strong Conformance

$$\cup\ \Sigma(S)=\{\sigma:(\Pi, CKit^{ag_i}, S) \vdash \langle p_m\rangle Fs\ w.a.\ \sigma\}$$

$p_0 \to p_1\, p_2$

$p_0 \to \varepsilon$

$\langle p_0\rangle\varphi \subset \langle p_1\rangle\langle p_2\rangle\varphi$

$\langle p_0\rangle\varphi \subset \varphi$

Verifying the conformance

- ***Proposition 2***: Protocol conformance is decidable (it can be reduced to the decideble problem of emptiness of context-free languages)

- ***Proposition 3***: The complexity for testing the protocol conformance is $O(n^4)$ time and $O(n^3)$ space

Conclusions and future works

- Methodology for producing skeletons that respect the protocol conformance.
- The work is in progress, future steps:
  - Turning the whole AUML 2.0 in linear grammars or finite automata
  - Integrating DyLOG in DCaseLP
    - Implementation of DyLOG+CKit (now only DyLOG in Sicstus Prolog)
  - Implementation of a graphical tool for programming in DyLOG and producing the DyLOG skeleton directly from an AUML interaction diagram
  - DyLOG represented by means of OWL