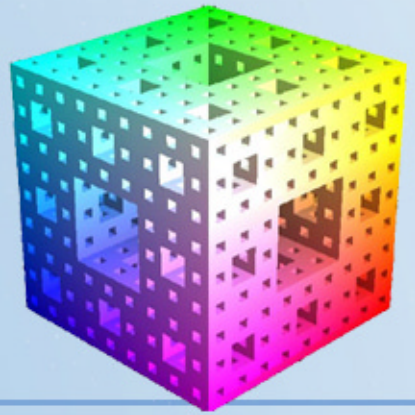
WOA 2004 - Sistemi Complessi e Agenti Razionali

A Game-Theoretic Operational Semantics for the DALI Communication Architecture

Stefania Costantini Arianna Tocchio Alessia Verticchio



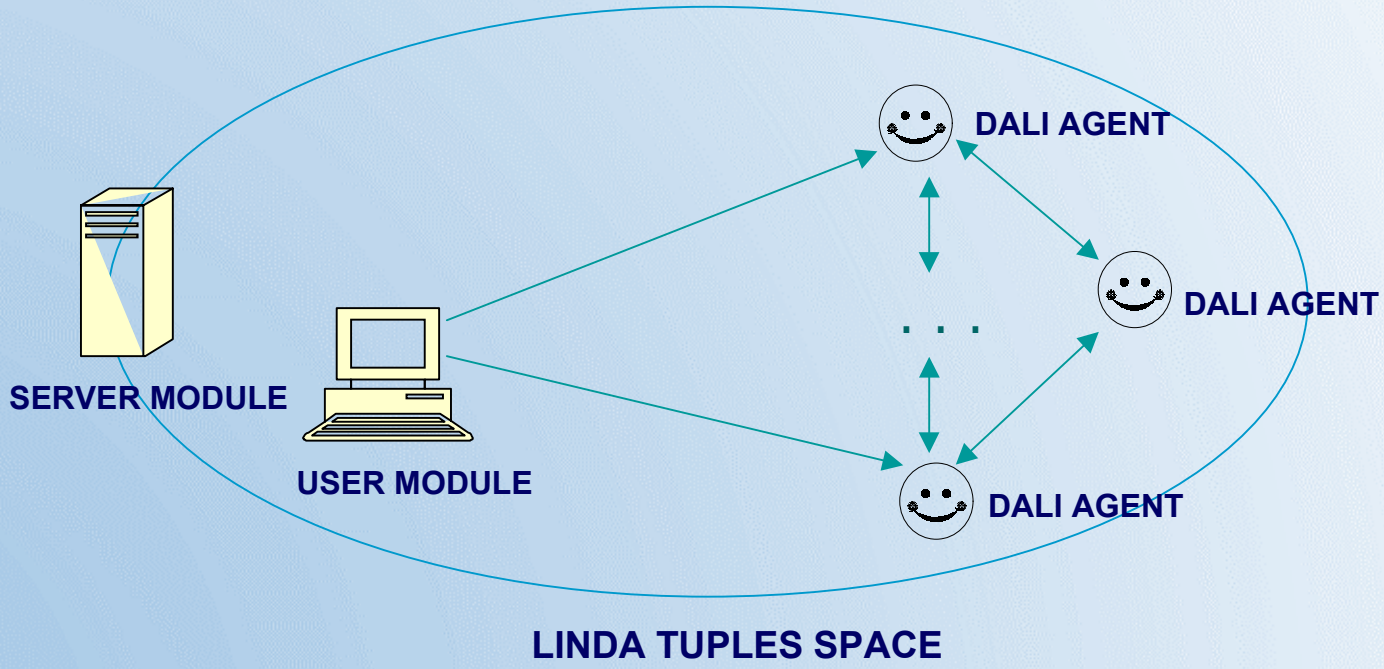
Università degli Studi di L'Aquila

SUMMARY

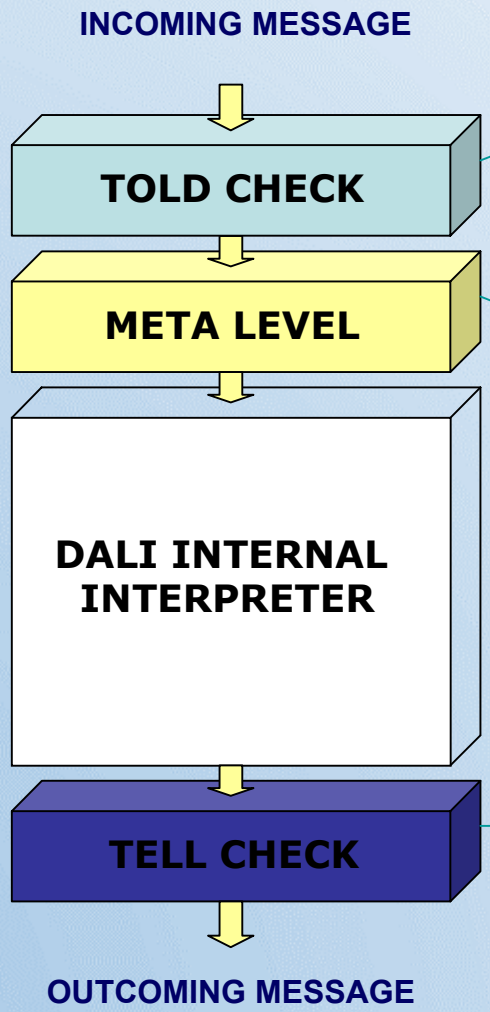
- ✓ DALI language
- ✓ Communication Architecture
- ✓ Game theory
- ✓ Operational Semantics
- ✓ Conclusions

WHAT IS THE DALI LANGUAGE?

DALI is an Agent-Oriented Logic Programming language designed for executable specification of logical agents , that allows one to define one or more agents interacting among themselves and with an external environment.



DALI COMMUNICATION ARCHITECTURE

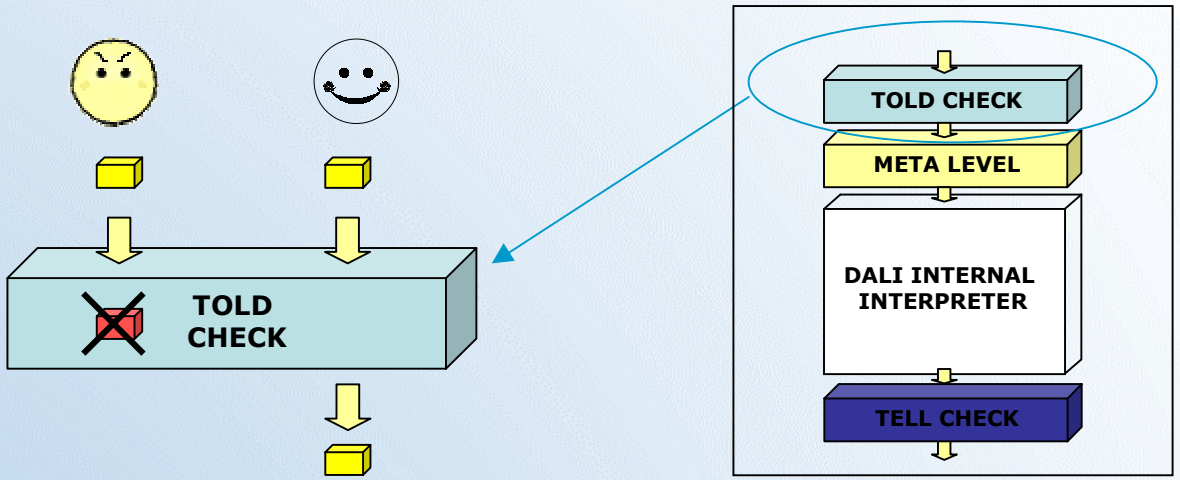


The message goes through this level only if the corresponding told rule is true.

If the agent doesn't know the content of the message, she calls the meta-level and uses the ontology and/or other properties in order to understand the communication act.

The message, submitted to the tell check, is sent only if the corresponding tell rule is true.

THE TOLD CHECK LEVEL



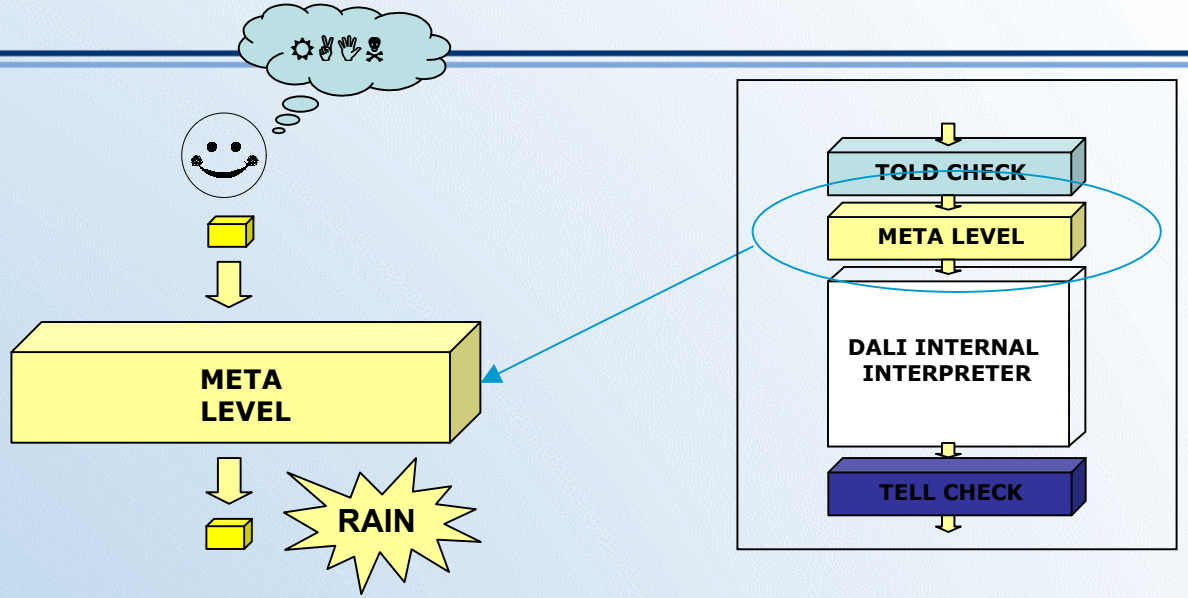
The structure of a told rule is:

told(Sender,Content):-constraint$_1$,…,constraint$_n$.

A message that does not go through the told level is eliminated.

Ex. ***told(Sender agent, propose(Action, Preconditions)) : ?***
not(unreliableP (Sender agent)), specialized for(Action).

THE META LEVEL



Each DALI agent uses a meta procedure to interpreter an unknown message.
This procedure can be modified by the user.

In the initialization file the user can specify the ontology .txt file that the agent must use. This file contains some strings with the following prefix and parameters.

ontology(Agent_name, Term1,Term2).

where Agent_name is the name of the agent that can use the association of terms Term1 and Term2.
In this file, the user can write some properties of terms useful to meta-procedure. For example.

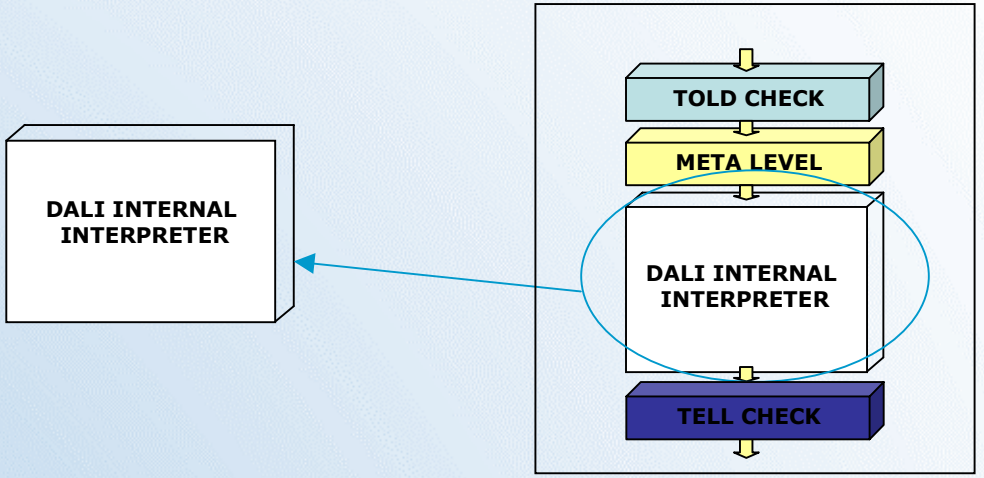
symmetric(love(julie,tom)).

specifies that the term love is symmetric.

DALI INTERNAL INTERPRETER

The internal interpreter generates the 'behaviour' of a DALI agent by managing:

- ✓ queue of external/present events
- ✓ queue of internal events
- ✓ queue of actions/messages
- ✓ queue of goals
- ✓ set of past events
- ✓ rules of logic program (reactive, action,…)
- ✓ internal directives for the interpreter
- ✓ generation of child-agent
- ✓ interface with external modules
- ✓…

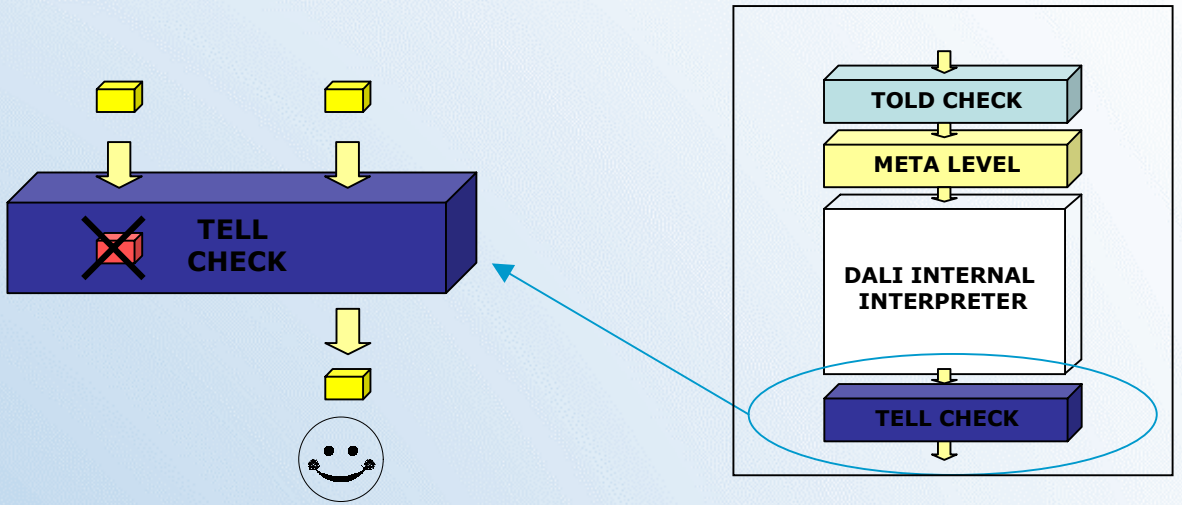



```
dangerE:>once(ask_for_help).
ask_for_help:-call_policeA.
call_police:<have_a_phoneP.
ask_for_help:-screamA.

remain_at_home:-dangerP,call_policeP.
remain_at_homeI:>go_to_bathroomA,
                 close_the_doorA.

go_out:-dangerP,screamP.
go_outI:>go_to_neighbourA.
```

THE TELL CHECK LEVEL



The structure of a tell rule is:

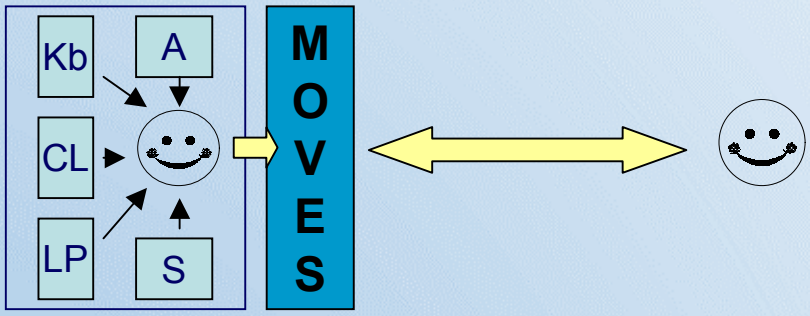
tell(Receiver,Sender,Content):-constraint$_1$,…,constraint$_n$.

A message that does not go through the told level is eliminated.

Ex. tell(_,_,refuse(X,_)):-functor(X,F,_),(F=send_message,F=query_ref).

WHY GAME THEORY?

Game theory is a mathematical framework designed for analyzing the interaction between several agents whose decisions affect each other. In a game-theoretic analysis, an interactive situation is described as a *game*: an abstract description of the players (agents), the courses of actions available to them, and their preferences over the possible outcomes. (Daphne Koller's article for the MIT Encyclopedia of Cognitive Science)



Larson, K. and Sandholm, T. (2001). Bargaining in computationally complex problems:Deliberation equilibrium.
Parkes, D. C. (2000). Optimal auction design for agents with hard valuation problems.
Sandholm, T. (1997). Unenforced ecommerce transactions.
Sandholm, T., Suri, S., Gilpin, A., and Levine, D. (2001). Cabob: A fast optimal algorithm for combinatorial auctions.

P. Mcburney, R. M. Van Eijk, S. Parsons, L. Amgoud (2003). A Dialogue Game Protocol for Agent Purchase Negotiations.

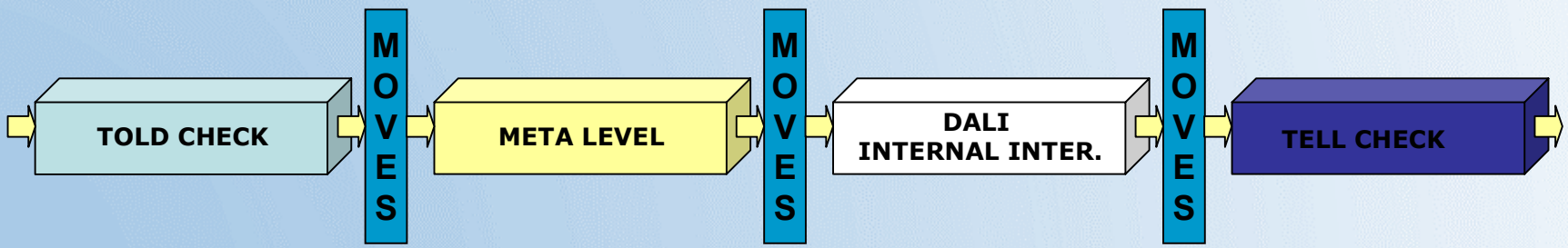
OPERATIONAL SEMANTICS

SEMANTICS is the meaning of a string in some language, as opposed to syntax which describes how symbols may be combined independent of their meaning.

OPERATIONAL SEMANTICS defines the behaviour of a program on a precisely defined mathematical "machine". The machine is defined as a transition system.

We will show how solutions from game theory together with computing theories can be used to publicly specify rules and prove desirable properties for agent systems.

We describe the **operational semantics of DALI** agents by using a **game-theoretic approach**.



FORMALIZATION OF DALI OPERATIONAL SEMANTICS

In order to formalize DALI operational semantics as steps of a dialogue game, we defined the behaviour of interpreter as a set of **states, transition rules and…laws**.

Agx to identify the **name** of the agent involved by the transition

S_{Agx} or **NS_{Agx}** to identify the **state** before and after the application of laws

L_x to identify the applied **law**

R_i to identify the **transition rules**

DEFINITION OF A STATE OF A DALI AGENT

$< Ag_x, S_{Agx} >$

indicates a link between the name of an agent and her state.

$S_{Agx} \equiv < P_{Ag} , IS_{Ag} , Mode_{Ag} >$

where P_{Ag} is the **logic program**, IS_{Ag} is the **internal state** and **Mode** is a particular attribute describing what the interpreter is doing.

The **internal state** IS_{Ag} of an agent is the tuple:

$< E, N, I, A, G, T, P >$

composed by the sets of, respectively, **external events, present events, internal events, actions,goals, test goals and past events.**

STRUCTURE OF A LAW

L_k: name of the law

Locution*:* arguments of the law

Preconditions*:* preconditions to application of the law

Meaning*:* meaning of the law

Response*:* how the law change the state of the agent

L0: the **receive message(.)** law

Locution: *receive message*(*Agx, Agy,Ontology, Language, Primitive*)

Preconditions: this law is applied when the agent *Agx* finds in the Tuple Space a message with her name.

Meaning: the agent *Agx* receives a message from *Agy*(environment, other agents,...).
For the sake of simplicity we consider the environment as an agent.

Response: the interpreter takes the information about the language and the ontology and extracts the name of sender agent and the primitive contained in the initial message.

STRUCTURE OF A TRANSITION RULE

$$R_i: \quad <Ag_x,<P,IS,Mode>> \xrightarrow{L_i,not(L_k)} <Ag_y,<NP,NIS,NMode>>$$

P_{Ag} is the **logic program**

NP_{Ag} is the **logic program modified by the application** of one or more laws

IS_{Ag} is the **internal state**

NIS_{Ag} is the **internal state modified**

Mode/NMode is a particular attribute describing **what the interpreter is doing**

L_i/not(L_j) are the **laws applied/not applied**

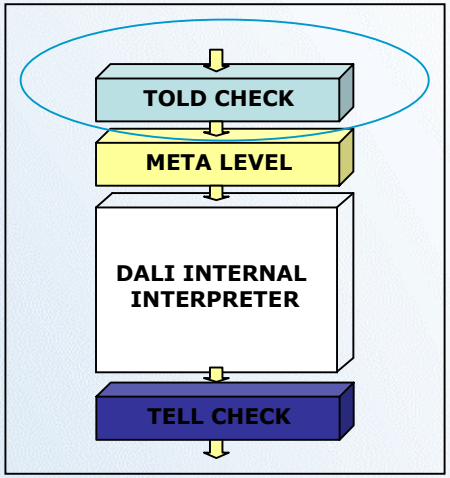
R0 : < Ag1,< P, IS, wait >> $\xrightarrow{L_0}$ < Ag1,< P, IS, received messagex >>

OPERATIONAL SEMANTICS: TOLD CHECK LEVEL

R0 : < Ag1,< P, IS, wait >> $\xrightarrow{L_0}$ < Ag1,< P, IS, received_messagex >>

R1 : < Ag1,< P, IS, received_messagex >> $\xrightarrow{L_1}$ < Ag1,< P, IS, toldx >>

R2 : < Ag1,< P, IS, received_messagex >> $\xrightarrow{not(L_1)}$ < Ag1,< P, IS, wait >>



L0: the **receive_message(.)** law:
Locution: *receive_message*(*Agx, Agy,Ontology, Language, Primitive*)
Preconditions: this law is applied when the agent *Agx* finds in the Tuple Space a message with her name.
Response: the interpreter takes the information about the language and the ontology and extracts the name of sender agent and the primitive contained in the initial message.

L1: the **L1 told_check_true(.)** law:
Locution:*told_check _true*(*Agy, Primitive*)
Preconditions: the constraints of told rule about the name of the agent sender *Agy* and the primitive must be true for the primitive *told_check_true*.
Response: depends on the constraints of told level. If the constraints are true the primitive can be processed by the next step.

OPERATIONAL SEMANTICS: META LEVEL

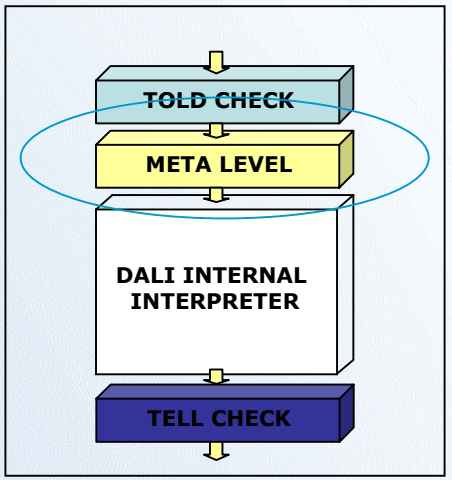
R3 : $< Ag1,< P, IS, toldx >> \xrightarrow{L_2} < Ag1,< P, IS, understoodx >>$

R4 : $< Ag1,< P, IS, toldx >> \xrightarrow{not(L_2),L_3} < Ag1,< P, IS, apply\_ontologyx >>$

R5 : $< Ag1,< P, IS, toldx >> \xrightarrow{not(L_2),not(L_3)} < Ag1,< P, IS, wait >>$

R6 : $< Ag1,< P, IS, apply\_ontologyx >> \xrightarrow{L_2} < Ag1,< P, IS, understoodx >>$

R7 : $< Ag1,< P, IS, apply\_ontologyx >> \xrightarrow{not(L_2)} < Ag1,< P, IS, wait >>$

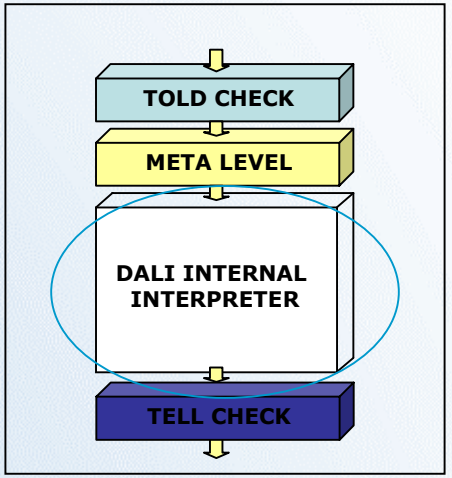


L2 : the **L2 understood(.)** law:
Locution: *understood*(*Primitive*)
Preconditions: in order to process the primitive the agent must understand the content of the message. If the primitive is *send message*, the interpreter will check if the external event belongs to a set of external events of the agent. If the primitive is *propose*, the interpreter will verify if the requested action is contained in the logic program.
Response:the message enters processing phase in order to trigger a reaction, communicate a fact or propose an action.

L3 : the **L3 apply_ontology(.)** law:
Locution: *apply ontology*(*Primitive*)
Preconditions: in order to apply the ontology the primitive must belong to set of locutions that invoke the metalevel(send message,propose,execute proc,query ref,is a fact).
Meaning: this law applies, when it's necessary, the ontologies to the incoming primitive in order to understand its content.
Response:the message is understood by using the ontology of the agent and properties of the terms.

OPERATIONAL SEMANTICS: DALI INTERNAL INTERPRETER

R8 : $< Ag1,< P, IS, understoodx >> \xrightarrow{L_6,L_7,L_8,L_9} < Ag1,< NP, NIS, processx >>$



Lk: the **add X(.)** law:
Locution: *add X*(:)
where $X \in$ *{internal event, external event, action, message, past event}*
Preconditions: the agent is processing X.
Response: the agent will reach a new state.
The state *SAg* of the agent will change in the following way.
k=6 and **X=internal event**:
$S_{Ag} =< P_{Ag},< E,N, I, A, G, T, P >,Mode >$
$NS_{Ag} =< P_{Ag},< E,N, I_1, A, G, T, P >,Mode >$ where
$I_1 = I \cup$ *Internal event*.
k=7and **X=external event**:
$S_{Ag} =< P_{Ag},< E,N, I, A, G, T, P >,Mode >$
$NS_{Ag} =< P_{Ag},< E_1,N, I, A, G, T, P >,Mode >$ where
$E1 = E \cup$ *external event*.
k=8 and **X=action**:
SAg =< *PAg*,< *E,N, I, A, G, T, P* >,*Mode* >
NSAg =< *PAg*,< *E,N, I,A*1, *G, T, P* >,*Mode* > where
$A1 = A \cup$ *Action* or $A1 = A$ ¥ *Action* if the communication primitive is *cancel*.
k=9 and **X=message**:
SAg =< *PAg*,< *E,N, I, A, G, T, P* >,*Mode* >
NSAg =< *PAg*,< *E,N, I,A*1, *G, T, P* >,*Mode* > where
A1=A ∪ Message. In fact, a message is an action.
k=10 and **X=past event**:
SAg =< *PAg*,< *E,N, I, A, G, T, P* >,*Mode* >
NSAg =< *PAg*,< *E,N, I, A, G, T, P*1 >,*Mode* > where
$P1 = P \cup$ *Past event*.

OPERATIONAL SEMANTICS: TELL CHECK LEVEL

R9 : $< Ag1,< P, IS, sendx >> \xrightarrow{L_4} < Ag1,< P, IS, tellx >>$

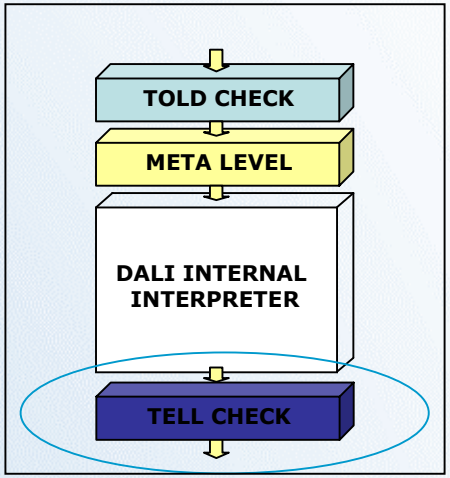
R10 : $< Ag1,< P, IS, sendx >> \xrightarrow{not(L_4),L_9} < Ag1,< P, NIS, sentx >>$

R11 : $< Ag1,< P, IS, tellx >> \xrightarrow{L_5,L_9} < Ag1,< P, NIS, sentx >>$

R12 : $< Ag1,< P, IS, tellx >> \xrightarrow{not(L_5)} < Ag1,< P, NIS, wait >>$

R13 : $< Ag1,< P, IS, sentx >> \xrightarrow{L_{10}} < Ag1,< P, IS, wait >>$

$< Ag1,< P, IS, sentx >> \longrightarrow \ldots < Ag2,< P, IS, received\_messagex >>$



L4: the **L4 send_message_with_tell(.)** law:
Locution:*send_msg_with_tell*(*Agx, Agy,Primitive*)
Preconditions: the precondition for L4 is that the primitive belongs to set of locutions submitted to tell check.
Response: the message will be sent to the tell level.

L5: the **L5 tell_check(.)** law :
Locution: *tell_check*(*Agx, Agy, Primitive*)
Preconditions: the constraints of tell rule about the name of the agent receiver *Agx*, the agent sender *Agy* and the primitive are true for L5.
Response: the message will either be sent to addressee agent(L5).

CONCLUSIONS

- ✓ is a formal base for verifying the correctness of the language
- ✓ allows one to check the properties of DALI system
- ✓ is the first step of the work useful to generate a model of the interpreter for a model checker

THANK YOU FOR YOUR ATTENTION