Empowering Emotional Behavior Trees with Neural Computation for Digital Forensic

Stefania Costantini [a] Pierangelo Dell'Acqua[b] Giovanni De Gasperis[a]
Andrea Rafanelli[a]

[a]University of Aquila, Italy
[b]University of Linkoping, Sweden

ESCIM 2024, DigForASP Workshop, May 14, 2024, Krakow, Poland

Introduction

A recent focus in Artificial Intelligence (AI) is the development of intelligent systems in which humans and AI Systems work together as teams (human-AI teaming, HAI).

When working together, humans and AI can produce results that exceed what they can achieve alone, whereas they can control and improve each other. A particularly important application of human-AI teaming pertains critical tasks which are burdensome and error-prone for humans.

AI Agents can thus be endowed with emotion recognition and be capable of empathy and modeling aspects of the Theory of Mind (ToM), in the sense of being able to reconstruct what the human is thinking or feeling.

Introduction (cont'd)

Affective computing refers to the domain of computing focused on leveraging emotions, feelings, moods, and other aspects of human psychology to accomplish specific tasks

Empathy brings benefits to social interactions, fostering better relationships. In fact, endowing artificial agents with empathetic capabilities enhances human-agent interactions. So, computational modelling of empathy has become an active research area.

This work focuses on modelling empathetic behaviour in virtual agents to be employed for user support in critical fields such as Digital Forensics as a follow-up to our work in the COST Action DigForASP.

An Envisaged Application

In DigForASP, we explored methods for aiding investigators by using AI, particularly computational logic and computational intelligence applied to digital forensics.

Despite the work done, digital forensics experts still face high pressure in high-stakes cases involving serious crimes. Long hours poring through volumes of data on multiple devices, though with the help of AI tools, can take a physical and mental toll.

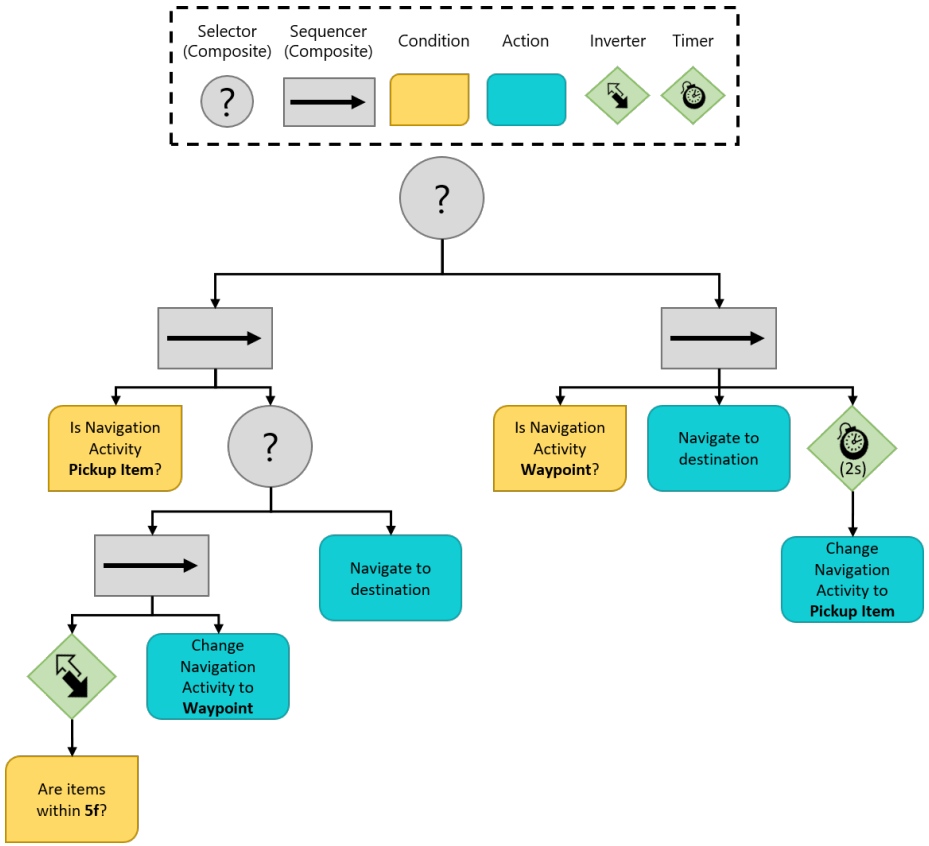
An Envisaged Application

The framework proposed in this work integrates emotional intelligence and empathy in a rigorous formal setting, aiming to provide effective support to investigators or in general of human users in high-pressure, high-stakes tasks.

By monitoring and analyzing the affective and physiological states of the human investigator, intelligent personal agents can adapt their behaviour to provide empathetic support, thus maintaining the productivity and accuracy of the analysis without compromising the well-being of the investigators.

A Formal Tool: Behaviour Trees (BTs)

A behaviour tree is a mathematical model of plan execution, widely adopted, for example, in videogames to model the behaviour of non-player characters (NPCs)



Our Proposal for HAI Agents: Affective States in BTs

Implemented in Prolog

The envisaged HAI agent is endowed with an "affective state", composed of a set of variables representing the agent's "emotions", i.e., how the agent "feels" in a human-relatable sense.

The agent elaborates the affective state during repeated interactions with the user and can thus tune its reaction accordingly.

Our Proposal for HAI Agents: Neural Empathy-Aware Behavior Trees (NEABT)

We enhance the basic BT definition with various kinds of nodes:

Neural Node
A neural node draws conclusions about a user's emotional state through a deep learning model. It takes sensor input from the environment (e.g., wearable devices, videocameras, etc.) catching sensory data like facial expressions, vocal tones, and body language, and the agent's previous emotional state. The node's outcome contributes to updating the agent's affective state variables.

By continually updating the agent's internal emotional state, the neural node allows dynamic adaptation of the EABT to the emotional context.

Our Proposal for HAI Agents: Neural Empathy-Aware Behavior Trees (NEABT)

We enhance the basic BT definition with various kinds of nodes:

Emotional Selector
The emotional selector is a node that orders its child sub-nodes based on a set of relevant decision factors and the agent's current "affective state".

Once the ordering has been established, the emotional selector behaves as a priority selector.

Our Proposal for HAI Agents: Neural Empathy-Aware Behavior Trees (NEABT)

We enhance the basic BT definition with various kinds of nodes:

Empathy Node An empathy node provides an emotional characterization of its single child node. An empathy node can only be a child of an emotional selector.

Its child can be a leaf or an inner node; where empathy nodes cannot be nested. An empathy node is represented by a dashed line circle with the name of the empathy emotion in it.

NEABT of the personal assistant agent assisting digital forensics experts

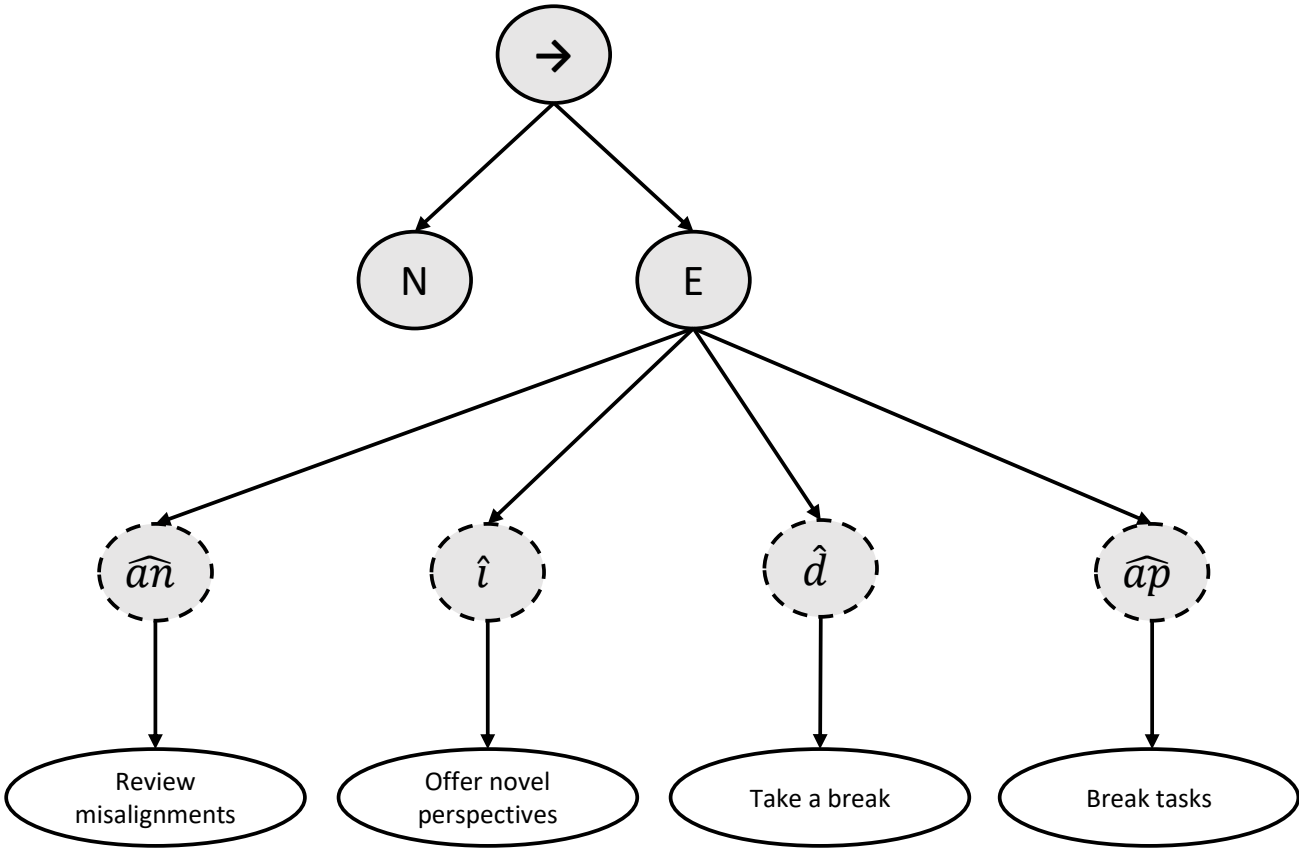
The objective is to monitor human investigators' affective states and modulate the agent's behaviour accordingly. We adopt a model characterising the human affect system across six primary emotive states– *anger*, *fear*, *surprise*, *sadness*, *disgust*, and *joy*.

By formalizing the agent's behaviour via a NEABT guided by affective factors, the agent can match its collaboration to the human's needs. For Digital Forensics Experts, this empathetic support keeps analysis productive without compromising human well-being.

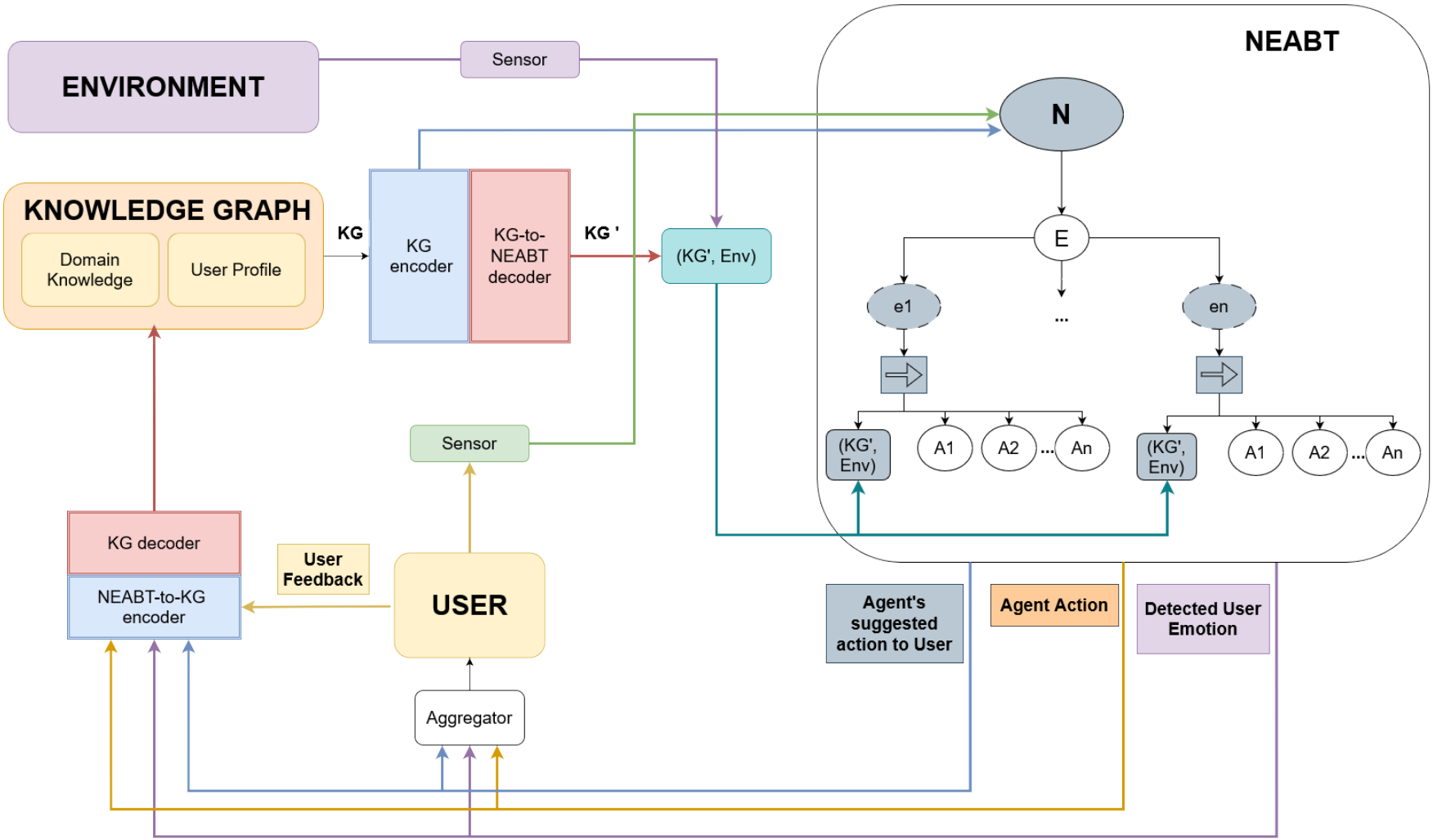
NEABT of the personal assistant agent assisting digital forensics experts

The empathy state variables are:

$\widehat{E} = \{\widehat{\text{annoyance}}, \widehat{\text{interest}}, \widehat{\text{distraction}}, \widehat{\text{apprehension}}\}$



Future Directions



Conclusion

The research that we just illustrated is a follow-up of DigForASP, and falls within the activities of the Italian PRIN Project:

TrustPACTX - Design of the Hybrid Society Humans-Autonomous Systems: Architecture, Trustworthiness, Trust, EthiCs

Participants are the Universities of L'Aquila, Messina, Naples "Federico II", and the National Research Council.

Thank you for your Attention!

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
FAIR Future Artificial Intelligence Research

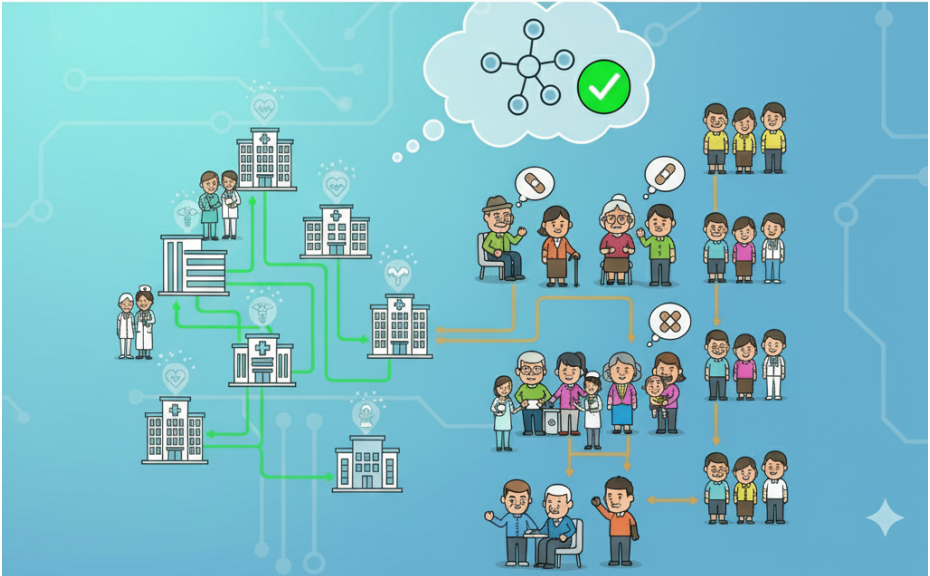
Epistemic Logic Meets Logic Programming for Environmentally Aware Agents in Dynamic Settings

Stefania Costantini
Università degli Studi dell'Aquila

ENABLE Project

December 10, 2025

A Multi-Agent System (MAS)



Agent Groups in MAS

- Many organizations can, in fact, be seen as a MAS, and within a MAS, agents can congregate in groups, aiming to reach common objectives cooperatively.
- It is important to model the dynamics in groups of cooperative agents.
- How to model if and how a group can reach its objectives? Explainability is crucial, allowing one to understand the system's conclusions.

International Association for
Safe & Ethical AI

Global (MAS') and Single Agent's Objectives

Appointment scheduling (by Costantini, Monaldini, Pado, Pitoni, Vozna, ICLP 2025 TC)

Medical appointment scheduling is complex:

- Our solution: **Answer Set Programming + Blueprint Personas**
  - **Structured patient archetypes** used in healthcare systems
  - Represent **socio-clinical profiles**, including:
    - Medical conditions (e.g., chronic diseases)
    - Accessibility needs (e.g., mobility or sensory limitations)
    - Preferences (e.g., clinic, doctor, time slots, distance, cost)
    - Social context (e.g., caregiver support)
    - Digital literacy
  - Encoded in ASP as logic facts and constraints
  - Enable personalized scheduling while maintaining scalability
- Adaptability and explainability
- Adapting to patients' preferences and needs
- Balancing urgency, resources, preferences, **green constraints**

Example and Limitations

A **disabled patient** may, e.g., be represented as follows:

```
patient(p1, "Maria", "Rossi", "L'Aquila").
disabled(p1).
appointment_preference(p1, c3, 1850, 2000).
sensory_preference(p1, "no_noise").
...
```

bf Priorities and urgencies are expressed through constraints of the type:

```
:- needs(P1, Visit, Urg1),
needs(P2, Visit, Urg2) , Urg1 > Urg2,
appointment(P1, Clinic, Doctor, Visit, Time1),
appointment(P2, Clinic, Doctor, Visit, Time2),
Time1 > Time2.
```

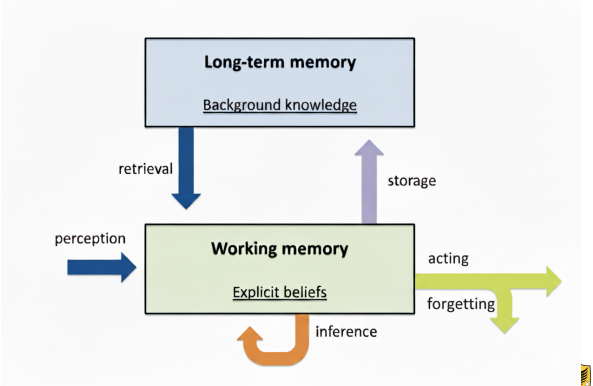
Captures needs, and offers more flexibility than existing systems, but:

- **Static**: no runtime changes, not even in case of **disruption**, without global re-scheduling that can bring more disruption.
- **No belief revision**: fixed preferences.
- **No collaboration**: patients seen as data, not as agents.

Paso Double: L-DINF

Epistemic logic framework for intelligent agents:

- Models cognitive features such as:
  - **Beliefs** (B_i), **Knowledge** (K_i)
  - **Intentions**, **Belief updates**, **Preferences**, **Group roles**
  - **Action feasibility**, **Action cost**, **Budget**, **Cost Sharing in a group**
  - **Time**
- Agents can:
  - Form and revise intentions
  - Reason about alternative actions
  - Coordinate via shared beliefs and group actions
- Supports **Theory of Mind**, dynamic planning, and social interaction



Motivation

- Why Epistemic Logic: it has proven to be a good tool to express the semantics underlying (aspects of) agent-oriented KRR, as it allows properties of the behaviour of an agent or a group of agents to be expressed and proved.

$$\mathbf{K}_{1,2,3}(human(X) \rightarrow mortal(X))$$
$$\mathbf{K}_{1,2,3}(human(me))$$

$$\mathbf{B}_1(ill)$$
$$\mathbf{B}_1(ill \rightarrow need\text{-}cure)$$
$$\mathbf{B}_2(need\text{-}cure \rightarrow consult\text{-}reputable\text{-}doctor)$$
$$\mathbf{B}_2(need\text{-}cure \wedge offer\text{-}bogus\text{-}remedy_{\mathbf{k}} \rightarrow refuse\text{-}bogus\text{-}remedy)$$

$$\mathbf{B}_3(rich(2) \wedge hypochondriac(2) \rightarrow offer\text{-}bogus\text{-}remedy)$$

Agents belonging to the same group (here, 1 and 2) share their conclusions, so when agent 1 has a problem, agent 2 provides advice and useful countermeasures.

Why Integrate ASP + Personas with L-DINF

Advantages

- **Dynamic adaptation** to real-time changes, no need of global re-scheduling
- **Explainability** via beliefs and intentions
- **Personalized decision-making**
- **Social coordination** through group logic
- **ASP optimization + cognitive reasoning**: a bridge between theory and practice.

Disadvantages

- Increased **logical complexity**
- Higher **computational cost** (???)
- More challenging to implement and validate (but good approximations are possible)

Use Case: Mario the Proactive Agent

Initial Setup:

- Mario has chronic conditions and noise sensitivity
- Preferences:
  - prefers_clinic = c1, preferred time = 08:30
  - doctor preference = George Peterson (GP), (Chronic disease specialist)
- Beliefs and preferences encoded as:

```
B_mario(prefers_clinic(c1)) ...
B_mario(accessible(c1) ∧ distance_to_clinic(c1, 12)
∧ doctor_available(GP,c1) → can_do_mario(slot(c1, 0830)))
```

Disruption: Clinic c1 becomes inaccessible ⇒ +(¬accessible(c1)) ⇒ Mario infers ¬can_do_mario(slot(c1, 0830)) ⇒ triggers goal revision

Use Case Mario (cont'd)

Adaptation Strategy: Mario looks for equivalent alternatives

```
B_mario(Cl(slot(c1, 0830), slot(c2, 0930)))
B_mario(pref_do_mario(slot(c2, 0930), 8))
B_mario(fCl_mario(slot(c1, 0830), slot(c2, 0930)))
B_mario(intend_mario(slot(c2, 0930)))
```

Group Change: Mario calls clinic c_2, and Anna, the receptionist, ensures that the consultation can be done. Formally, Mario has to join Anna's group G_2 at clinic c_2, and this leads to updating his beliefs

```
do_mario(joinA(mario, anna))
B_mario((accessible(c2))
B_mario(doctor_available(DocP,c2)) ...
```

The new intention:

```
B_mario(can_do_mario(slot(c2, 0930))
B_mario(do_mario(slot(c2, 0930)))
```

Summary: Belief revision → new plan → group coordination → action execution

Use Case: Belief Revision and Cross-Group Delegation

Scenario: two clinics, $Clinic_A$ and $Clinic_B$, which are two different (agent) groups with their own medical personnel.

- A patient, *Alice*, is associated with $Clinic_A$, which means that the Alice agent is in this group. She is scheduled a consultation in a time slot t_1. She is seriously ill and cannot be moved
- The doctor at $Clinic_A$, Dr. Jones, becomes unavailable due to an emergency. However, Dr. Smith from $Clinic_B$ is qualified and available for the same action.
- So, $Clinic_B$, which is willing, temporarily lends Dr. Smith to $Clinic_A$ for the consultation.

This example demonstrates two key L-DINF capabilities:

- Belief revision: Agents dynamically update their internal knowledge in response to changes.
- Delegation across groups: a capable agent from one group is temporarily lent to another to perform an action that no local agent can perform.

Use Case: Initial Beliefs and Agents' Capabilities:

`B_alice(needs_consultation(t1))`
`B_docJ(can_do(consultation(t1)))`

Disruption: Dr. Jones can not do that action anymore
$\Rightarrow +(\neg can\_do\_docJ(consultation(t1)))$
$\Rightarrow B\_alice(\neg can\_do\_docJ(clinic_A, consultation(t1)))$

Delegation and New plan:
$lend\_clinic_A(j, clinic_B, consultation(t_1)) \leftarrow$
$\forall i \in clinic_A \ \neg can\_do_i(consultation(t_1)) \wedge$
$\exists j \in clinic_B \ can\_do_j(consultation(t_1)) \wedge willing\_lend(clinicB, clinicA)$

`B_docS(can_do(consultation(t1)))`
`B_i(lend_clinic`$_A$`(docS, clinic`$_B$`, consultation(t1)))`
`B_docS(join`$_A$`(docS, docJ))`
`B_docS(do_docS(consultation(t1)))`

Conclusions

✔ **Summary:** We have proposed an Hybrid Architecture ASP + L-DINF:

- **ASP + Blueprint Personas**: for global optimization under personal and global (green) constraints
- **L-DINF**: for runtime intelligence
- Combined: responsive and explainable scheduling, no repeated global re-scheduling
- Stronger alignment with real clinical dynamics and patients' needs and preferences

🛈 **Ongoing: implementation and validation in the field**

- L-DINF dynamic features: prototypically implemented in DALI ("our" Multi-Agent Systems language and framework)
- Lending agents: see RCRA 2025 paper (by Costantini, De Gasperis, De Meo, Gullo, and Provetti)
- Other group dynamics: under study

Thank You!

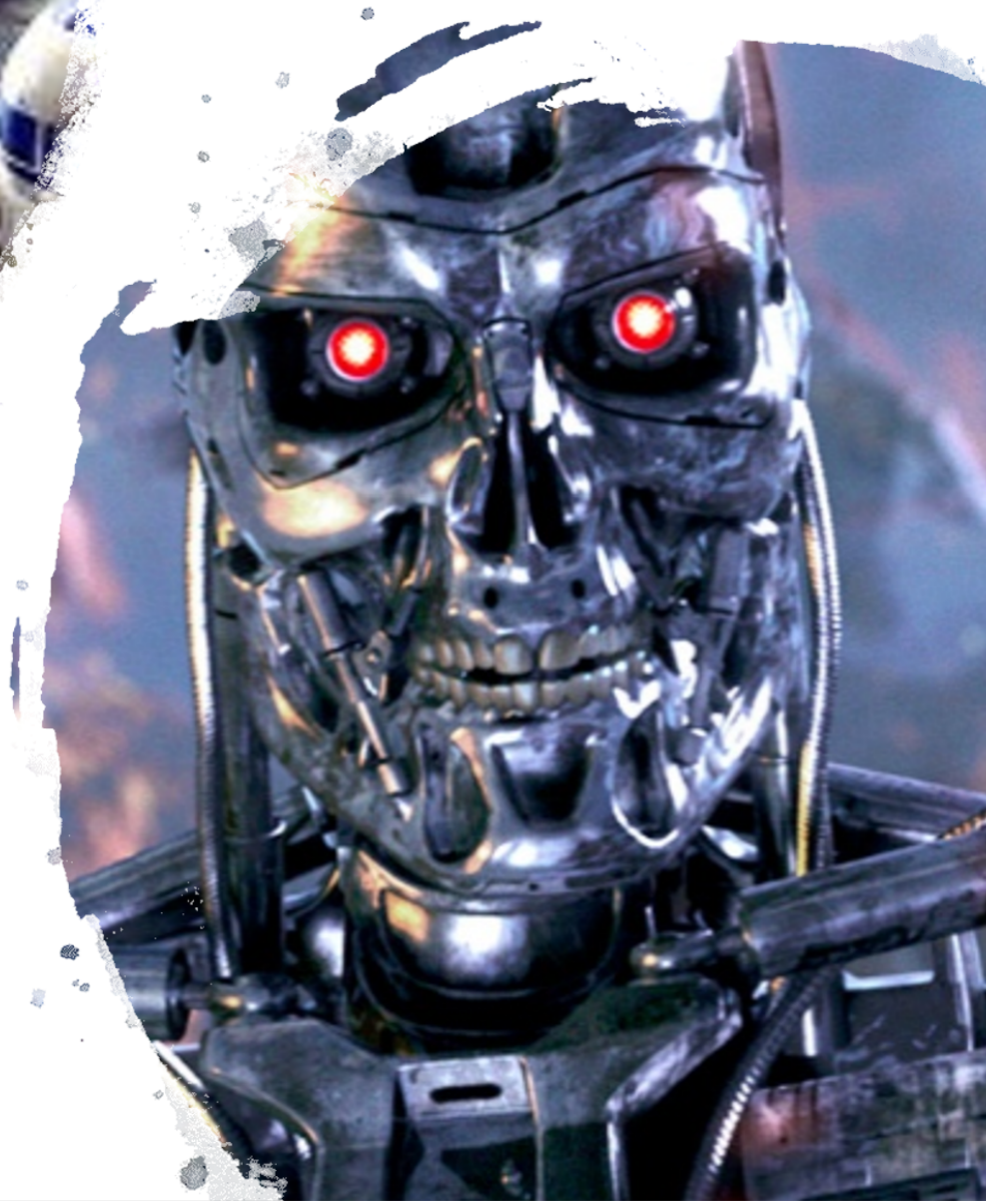
Runtime Self-Checking of Logical Agent Systems

Stefania Costantini

University of L'Aquila, Italy

Genova, January 15, 2024

UniGe | DIBRIS



Since when
there is fear of
‘the bad robot’?

- How to control the behavior of Intelligent Autonomous Agents?
- If agents interact with humans, this also involves the ethical aspects.
- We need certification methods (a priori) but also "assurance" (at run-time):
  - *in fact, via learning and interaction with the environment, agents can develop behaviors and objectives not in line with user's intentions.*

Intelligent Agents but… controllable, and adherent to human values

Self-aware Reflective computing (or human in the loop)

Tørresen, Plessl, and Yao 2015:

Self-aware and self-expressive computing describes an emerging paradigm for systems and applications … transforming our relationship with and use of computers.

Amir, Andreson, and Chaudri 2007:

A self-aware system must have sensors, effectors, memory, … reasoning, learning, goal setting, and an explicit awareness... The system should be reactive, deliberative, and ***reflective****.*

Costantini, Lanzarone et al. 2000

The aim of metalevel [and reflection] has been viewed as a guide for the object level inference or computation, i.e., for expressing 'properties of control' in the same way as 'properties of the domain'

"Reflection" mechanisms

- Agents should "reflect" on themselves to block inappropriate behavior and lead to self-modification and self-improvement.
- Reflection mechanisms can be local, operating on single actions, or global, at the level of objectives and planning.
- Such global mechanisms, similar to integrity constraints, can be based on reflexive "meta-reasoning".

'Local' Reflection mechanisms

$solve(execute\_action'(Act')) :-$
$present\_context(C), ethical(C, Act').$

$solve\_not(execute\_action'(Act')) :-$
$present\_context(C), ethical\_exception(C, Act').$

% *We might have e.g., one of the following :*
$context(reality).$
$context(videogame).$
$context(storytelling).$
. . . or others

- A is the name f atom A, computed by any so-called *naming mechanism*; it is said that atom A is *reified*
- We restrict any semantics SEM to determine only *acceptable* sets of atoms.
- I is an *acceptable* set of atoms iff I satisfies the axiom schemata:

$$A \leftarrow solve(A) \qquad \neg A \leftarrow solve\_not(A)$$

- Current subgoal A is *reified*, i.e., its name is computed.
- Applicable *solve* and *solve_not* metarules are searched; if such metarules are found, control, in fact, shifts from the object to the metalevel (consider that *solve* and *solve_not* can rely upon any set of auxiliary metalevel rules).
- If no rule is found or whenever *solve* and *solve_not* metarules complete their execution, *downward reflection* returns control to the object level, to execution of A if confirmed or to subsequent goals/actions if A has been cancelled by either failure of an applicable *solve* metarule or success of an applicable *solve_not* metarule.

Verification of Properties in Agents

- How to ensure that a property φ holds in any future state of an agent?

  - Verify φ prior to agent's activation: this is related to *certification*, aimed at producing evidence indicating that deploying a given system in a given context involves the lowest possible level of risk of adverse consequences (techniques such as Model Checking and Theorem Proving).
  - Perform a run-time verification of φ, so that suitable counter-measures can be undertaken in case of violation: this is related to *assurance*, or *dependability*, i.e., to ensuring (or at least obtaining a reasonable confidence) that system users can rely upon the system.

Motivation for Run-time Verification

Given the evolving nature of learning agents, their behaviour can hardly be fully verifiable "a priori".

- A priori verification should be reiterated whenever an agent learns new information.
- A priori complete validation of agents' behaviour would have to consider all possible scenarios that may not be known in advance.
- Sometimes, requirements can be either missing or incomplete.

Our Approach

- Based on an interval temporal logic tailored to the agent's realm.
- Not fully implemented as a logic, but rather implemented in the form of special meta-axioms to be periodically checked (at low complexity).
- Repair/improvement actions provided.

- Several interesting interval temporal logics exist, mainly devised for applications involving real-time and hybrid systems, with an underlying 'dense' (continuous) model of time.
- General results about expressiveness, decidability and complexity are often lacking.
- A-ILTL: Defined as a simple extension of LTL, then still relying upon an underlying discrete linear model of time. Suitable for checking agent's properties over discrete time.
- For limiting complexity: no nesting of interval connectives, so as to avoid having to explicitly cope with the interaction among time intervals.

A-ILTL (Agent-Oriented LTL) Logic

A-ILTL expressions are interpreted (like LTL ones) in a discrete, linear model of time represented by a suitable structure $\mathcal{M}$ including a sequence of states and an interpretation function for (sets of) atomic propositions over these states, and a satisfaction relation $\models$.

Set $\mathcal{F}$ of A-ILTL formulas is built out of classical connectives and of LTL operators and of A-ILTL operators (the latter cannot be nested).

Given formula $\varphi \in \mathcal{F}$ and state (time instant) i, we write $\mathcal{M}, i \models \varphi$ if, in the satisfaction relation, φ is true w.r.t. $\mathcal{M}, i$. We can also say that φ *holds* at i, or equivalently in state s_i, or that state s_i satisfies φ.

A-ILTL Interval operators

Let m, n (with $m \leq n$) be positive integer numbers, and φ and ψ be LTL formulas.

$C(i)$ (*current state*). $C(i)$ is true if s_i is the current state. I.e., $\mathcal{M}, i \models C(i')$ iff $i = i'$. From this operator we obtain the shorthand expression now, where $now = t : C(t)$.

p_i (*p at i*). Proposition p holds at time i. This notation transposes a propositional letter into a "timed" version. I.e., $\mathcal{M}, i \models p(i)$ if $\mathcal{M}, i \models p$.

$p_{\langle i \rangle}$ (*p since i*). Proposition p holds since time i. This notation transposes a propositional letter into a "timed" version, where due to the interaction with the environment, the agent modifies its knowledge base.

A-ILTL Interval operators

Let m, n (with $m \leq n$) be positive integer numbers, and φ and ψ be LTL formulas.

X_m (*future m-state*). $X_m\varphi$ holds if φ is true in the $(m+1)$-th state after the current state. I.e., $\mathcal{M}, i \models X_m\varphi$ if $\mathcal{M}, i' \models \varphi$, $i' = i + m$. This operator is relative to current state $C(i)$, in fact it can hold or not hold depending on state i that is considered. Therefore a more suitable form is $X_m(i)$, where the reference state is explicitly stated.

A-ILTL Interval operators

Let m, n (with $m \leq n$) be positive integer numbers, and φ and ψ be LTL formulas.

F_m (*bounded eventually (or "finally")*). $F_m\varphi$ holds if φ is true somewhere on the path from the current state to the (m)-th state after the current one. I.e., $\mathcal{M}, i \models F_m\varphi$ if there exists j such that $j \geq i$ and $j \leq i + m$ and $\mathcal{M}, j \models \varphi$. This operator is relative to current state $C(i)$, in fact it can hold or not hold depending on state i that is considered. Therefore a more suitable form is $F_m(i)$, where the reference state is explicitly stated.

$F_{m,n}$ (*eventually (or "finally") in time interval*). $F_{m,n}\varphi$ states that φ has to hold somewhere on the path from state s_m to state s_n. I.e., $\mathcal{M}, i \models F_{m,n}\varphi$ if there exists j such that $j \geq m$ and $j \leq n$ and $\mathcal{M}, j \models \varphi$.

A-ILTL Interval operators

Let m, n (with $m \leq n$) be positive integer numbers, and φ and ψ be LTL formulas.

G_m (*bounded always*). $G_m\varphi$ states that φ should become true at most at state s_m. It differs from LTL G in that the state where the property should start to hold is explicitly indicated. I.e., $\mathcal{M}, i \models G_m\varphi$ if for all j such that $j \geq m$ $\mathcal{M}, j \models \varphi$.

$G_{\langle m \rangle}$ (*bounded strong always*). $G_{m,n}\varphi$ states that φ should become true just at state s_m, while it was not true at previous states.

$G_{m,n}$ (*always in time interval*). $G_{m,n}\varphi$ states that φ should become true at most at state s_m and then hold at least until state s_n. I.e., $\mathcal{M}, i \models G_{m,n}\varphi$ if for all j such that $j \geq m$ and $j \leq n$ $\mathcal{M}, j \models \varphi$.

$G_{\langle m,n \rangle}$ (*strong always in time interval*). $G_{\langle m,n \rangle}\varphi$ states that φ should become true just in s_m and then hold until state s_n, and not in s_{n+1}.

A-ILTL Interval operators

Let m, n (with $m \leq n$) be positive integer numbers, and φ and ψ be LTL formulas.

N^b_m (*never before*). $N^b_m\varphi$ states that φ should not be true in any state prior than s_m, i.e., $\mathcal{M}, i \models N^b_m\varphi$ if there not exists j such that $j < m$ and $\mathcal{M}, j \models \varphi$.

N^a_m (*never after*). $N^a_m\varphi$ states that φ should not be true in any state after s_m, i.e., $\mathcal{M}, i \models N^a_m\varphi$ if there not exists j such that $j > m$ and $\mathcal{M}, j \models \varphi$.

$N_{m,n}$ (*never in time interval*). $N_{m,n}\varphi$ states that φ should not be true in any state between s_m and s_n, i.e., $\mathcal{M}, i \models N_{m,n}\varphi$ if there not exists j such that $j \geq m$ and $j \leq n$ and $\mathcal{M}, j \models \varphi$.

A-ILTL Interval operators

Let m, n (with $m \leq n$) be positive integer numbers, and φ and ψ be LTL formulas.

$E_{m,f}$ (*bounded sometimes*). $E_{m,n}\varphi$ states that φ has to be true one or more times starting from state s_m, with frequency f. I.e., $\mathcal{M}, i \models E_{m,f}\varphi$ if $\mathcal{M}, m \models \varphi$ and $\mathcal{M}, i \models E_{m',f}\varphi$, $m' = m + f$.

$E_{m,n,f}$ (*sometimes in time interval*). $E_{m,n}\varphi$ states that φ has to be true one or more times between s_m and s_n, with frequency f. I.e., $\mathcal{M}, i \models E_{m,n,f}\varphi$ if $\mathcal{M}, i \models \varphi$ whevener $m + f \geq n$, or otherwise if $\mathcal{M}, m \models \varphi$ and $\mathcal{M}, i \models E_{m',n}\varphi$ with $m' = m + f$.

A-ILTL Response Rules

We can employ A-ILTL in *response rules* (Manna 2010), of the form $p \Rightarrow q$ (meaning that any state which satisfies p must be followed by a later state which satisfies q).

Example: upon an order issued at date d, the corresponding product should be always received within k days.

$$G(sent\_order_{\langle d \rangle} \Rightarrow F_{d+k_{days}}\ product\_received)$$

$$G(sent\_order(prod)_{\langle t \rangle} \Rightarrow F_{t+k_{days}}\ product\_received(prod))$$

In a logic agent-oriented programming language, like, e.g., DALI

Example

$$FINALLY(T, T+k; freq)\ product\_received(P) ::$$
$$sent\_order(P) : T, product(P) \qquad \% \ context$$
$$\div\ complain \qquad \% \ repair\ action$$
$$\div\ give\_good\_rating \qquad \% \ improvement\ action$$

Expressing General Properties

Example

$$
\begin{array}{l}
NEVER \\
\quad goal(G), \\
\quad eval\_context(G, M), P(G, M), \\
\quad not\ timed\_out(G) \\
\quad not\ achieved(G) \div \\
\qquad retry_A(G)
\end{array}
$$

The reactive part might be

reconsider_context(G,M,M'),P(G,M'),retry$_A(G)$

Here, the module for evaluating possibilities could be updated, and this might lead to either continuing or stopping retrying the goal.

A-ILTL Response Rules in practice

A-ILTL Contextual Rules

- Checked dynamically at (possibly default) frequency *freq* within a *critical interval* that can be easily determined.
- Enhancement: *Evolutionary A-ILTL Expressions*, triggered by the occurrence of a (partially specified) sequence of events, regular-expressions-like notation. Possible specification of what may be expected to happen during the critical interval: *normal* subsequent event occurrences (should not disrupt a property) or *breaking* event occurrences (undermine a property).
- A-ILTL rules are integrated into the *Evolutionary Semantics* for agent-oriented logical languages, which is a general framework encompassing, e.g., AgentSpeak, DALI, GOAL.

States in our setting are not simply intended as time instants: Rather, they correspond to stages of the agent evolution. We extend the definition by Manna, 1991:

Definition

Let σ be a (finite or infinite) sequence of states, where the ith state e_i, $e_i \geq 0$, is the *semantic snapshots at stage* i ε_i^{Ag} of given agent Ag. Let T be a corresponding sequence of time instants t_i, $t_i \geq 0$. A *timed state sequence for agent* Ag is the couple $\rho_{Ag} = (\sigma, T)$. Let ρ_i be the i-th state, $i \geq 0$, where $\rho_i = \langle e_i, t_i \rangle = \langle \varepsilon_i^{Ag}, t_i \rangle$.

We consider timed state sequences which are *monotonic*, i.e., if $e_{i+1} \neq e_i$ then $t_{i+1} > t_i$.

A-ILTL and Agent Evolution (cont'd)

Given A-ILTL rules $\{\tau_1, \ldots, \tau_l\}$ associated to an agent, the agent's evolution can be considered to be "satisfactory" only if it obeys all these properties.

Definition

Given agent Ag and given a set of A-ILTL expressions $\mathcal{A} = \{\tau_1, \ldots, \tau_l\}$, timed state sequence ρ_{Ag} is *coherent* w.r.t. $\mathcal{A}$ if A-ILTL formula $G\zeta$ with $\zeta = \tau_1 \wedge \ldots \wedge \tau_n$ holds.

The expression $G\zeta$ is an *invariance property* in the sense of Manna, 1984.

The formulation $G_{m,n}\zeta$ allows *temporally limited coherence* to be expressed, concerning for instance "critical" parts of an agent's operation. Or also, one might express forms of *partial* coherence concerning only some properties.

Reactive A-ILTL Rules

Definition

A reactive A-ILTL rule is of the form (where M, N, K can be either variables or constants)

$$OP(M, N; K)\varphi :: \chi \div \rho$$

where:(i) $OP(M, N; K)\varphi :: \chi$ is a contextual A-ILTL formula, called the *monitoring condition*, that should involve the observation of either external or internal events; (ii) ρ is called the *recovery component* of the rule, and it consists of a complex reactive pattern.

$$ALWAYS(8:00\ a.m.,\ 5:00\ p.m.; 10m)\ 19 \leq temperature \leq 21$$
$$modify\_temperature_G(S),$$
$$S\ IN\ \{external\_electricity, gas, solar\_panel\_electricity\ :\ most\_effective\}$$

Evolutionary A-ILTL Expressions

Definition

Let $\mathcal{S}^{\mathcal{E}vp}$ be a sequence of past events, and $\mathcal{S}^{\mathcal{F}}$ and $\mathcal{J}^{\mathcal{J}}$ be sequences of events. Let τ be a contextual A-ILTL formula $Op\,\varphi :: \chi$. An *Evolutionary LTL Expression* ϖ is of the form $\mathcal{S}^{\mathcal{E}vp} : \tau ::: \mathcal{S}^{\mathcal{F}} :::: \mathcal{J}^{\mathcal{J}}$ Where: (i) $\mathcal{S}^{\mathcal{E}vp}$ denotes the sequence of relevant events which are supposed to have happened and in which order, for the rule to be checked, i.e., these events act as preconditions: whenever one or more of them happen in the given order, τ will be checked; (ii) $\mathcal{S}^{\mathcal{F}}$ denotes the events that are expected to happen in the future without affecting τ; (iii) $\mathcal{J}^{\mathcal{J}}$ denotes the events that are expected *not* to happen in the future; i.e., whenever any of them should happen, φ is not required to hold any longer, as these are "breaking events".

Evolutionary A-ILTL Expressions: Example 1

Keeping the available quantity of a product under control: given an unknown number of incoming events and an unknown number of actions, the *invariant* must hold.

Example

$$supply_P{}^{+}(r, \_s) \ : \\ N(quantity(r, V), V < th) \ ::: \\ consume_A{}^{+}(\_r, Q)$$

But what if the invariant does not hold?

Evolutionary A-ILTL Expressions: Example 2

Controlling a robot working on batteries

Whenever an Evolutionary A-ILTL expression is either violated or broken, a reaction can be attempted aiming at recovering a desirable or at least acceptable agent's state.

Example

$$
\begin{aligned}
&recharge\_battery_P : T \ : \\
&\quad ALWAYS(T, T + 6_{hour}) \ \ charge\_level(L), L > low \\
&\quad ::: normal\_usage\_act(Act)* \ \ :::: extensive\_usage\_act(Act)* \\
&\qquad \mid alert\_user\_possible\_fault_A \ || \ recharge\_battery_G
\end{aligned}
$$

We have used an A-ILTL constraint as a programming construct, which, however, has a role in assurance since it forces the agent to respect timing, which is essential for the system's good functioning.

Complexity of Check

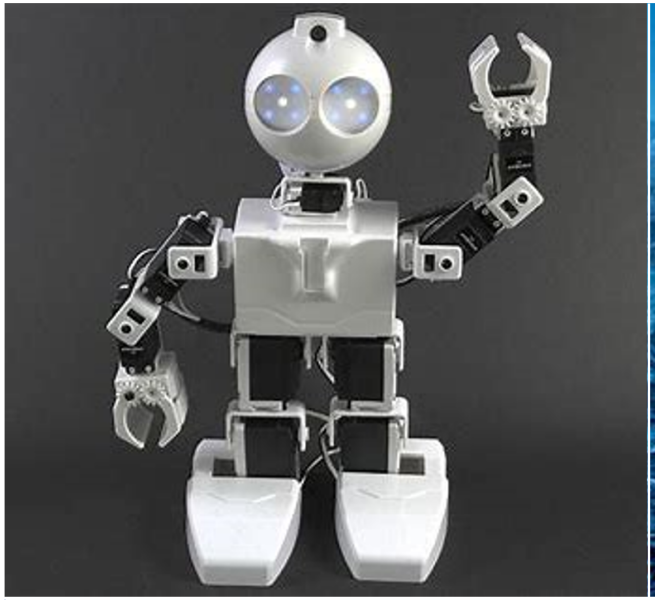
Assumptions: (1) same frequency for all f expressions; (2) k different A-ILTL operators occurring in the f expressions; (3) max time m for retrieving each expression; (4) max time max_eval for context evaluation; (5) max time if_eval for deciding whether each expression needs to be evaluated at the present state: in case of Evolutionary A-ILTL Expressions, includes checking the triggering event sequence w.r.t. current agent's history; (6) max time $if\_viol\_or\_broken$ for deciding whether an expression is violated.

Total time for checking:

$$\mathcal{O}((f \star m) + (f \star (if\_eval + max\_eval + if\_viol\_or\_broken)))$$

Concluding Remarks

- (Part of) the power of temporal logic without related computational complexity.
- Run-time periodical checking at low complexity, repair and improvement actions provided.
- Flexible language-independent constructs with wide potential applicability.
- Partly implemented and partly simulated in DALI.
- Experiments show that performance depends on the number of meta-axioms, anyway better than if performing such checks in other ways.



Thank you for your attention!

Questions are welcome

Integrating Cognitive Reasoning into Medical Scheduling: a Transition from ASP Personas to L-DINF Agents

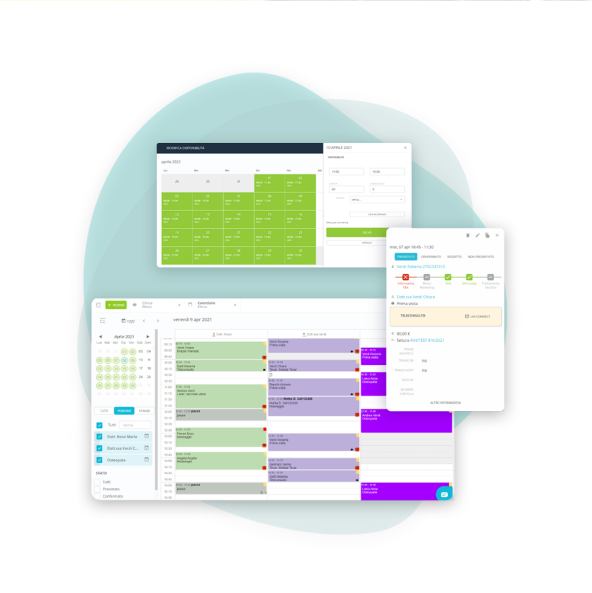
Valentina Pitoni, Stefania Costantini

Department of Information Engineering, Computer Science and Mathematics,
University of L'Aquila, Italy

16 Dicembre 2025

PRIMA 2025
Modena, Italy

Motivation



Medical appointment scheduling is complex:

- Balancing urgency, resources, preferences
- Traditional systems are static
- Need for adaptability and explainability

Background: ASP + Blueprint Personas

Blueprint Personas are:

- **Structured patient archetypes** used in healthcare systems
- Represent **socio-clinical profiles**, including:
  - Medical conditions (e.g., chronic diseases)
  - Accessibility needs (e.g., mobility or sensory limitations)
  - Preferences (e.g., clinic, doctor, time slots)
  - Social context (e.g., caregiver support)
  - Digital literacy
- Encoded in ASP as logic facts and constraints
- Enable personalized scheduling while maintaining scalability

Example and Limitations

A disabled patient may be represented as follows:

```
patient(p1, "Mario", "Rossi", "L'Aquila").
disabled(p1).
appointment_preference(p1, c3, 1850, 2000).
```

Priorities are expressed through constraints of the type:

```
:- needs ( P1, Visit, Urg1 ),
needs ( P2, Visit, Urg2 ) , Urg1 > Urg2,
appointment ( P1, Clinic, Doctor, Visit, Time1 ),
appointment ( P2, Clinic, Doctor, Visit, Time2 ),
Time1 > Time2.
```

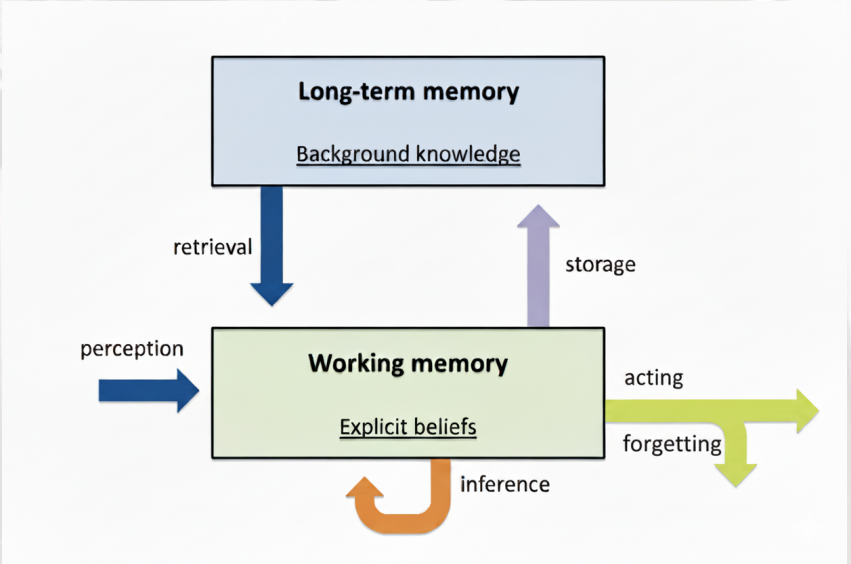
Captures needs, but:

- **Static**: no runtime changes
- **No belief revision**: fixed preferences
- **No collaboration**: agents are isolated

Background: L-DINF

Epistemic logic framework for intelligent agents:

- Models cognitive features such as:
  - **Beliefs** (B_i), **Knowledge** (K_i)
  - **Intentions**, **Belief updates**, **Preferences**, **Group roles**
  - **Action feasibility**, **Action cost**, **Budget**, **Cost Sharing in a group**
  - **Time**
- Agents can:
  - Form and revise intentions
  - Reason about alternative actions
  - Coordinate via shared beliefs and group actions
- Supports **Theory of Mind**, dynamic planning, and social interaction



L-DINF: Syntax

The language of *L-DINF*, denoted by $\mathcal{L}_{\text{L-DINF}}$, is defined by the following grammar in Backus-Naur form:

$$
\begin{array}{lll}
\varphi, \psi & ::= & p \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathbf{B}_i\,\varphi \mid \mathbf{K}_i\,\varphi \mid \\
 & & do_G(\phi_A) \mid can\_do_G(\phi_A) \mid do^P_G(\phi_A) \mid pref\_do_G(i, \phi_A) \mid pref\_do_i(\phi_A, d) \\
 & & intend_i(\phi_A) \mid intend_G(\phi_A) \mid exec_i(\alpha) \mid exec_G(\alpha) \mid [G : \alpha]\,\varphi \\
 & & CI(\phi_A, \phi'_A) \mid fCI_i(\phi_A, \phi'_A) \mid lend_G(i, H, \phi_A) \\
\alpha & ::= & \vdash(\varphi,\psi) \mid \cap(\varphi,\psi) \mid \downarrow(\varphi, \psi) \mid \dashv(\varphi, \psi)
\end{array}
$$

where p ranges over Atm and $i \in Agt$.

- $\top$, $\bot$, $\rightarrow$, $\leftrightarrow$ are defined from $\neg$ and $\wedge$ in the standard way, and
- B_i, K_i, denote modal operators;
- Those in α are the "mental actions" that an agent can perform.

Why Integrate Personas with L-DINF

Advantages

- **Dynamic adaptation** to real-time changes
- **Explainability** via beliefs and intentions
- **Personalized decision-making**
- **Social coordination** through group logic
- Enhances ASP optimization with cognitive reasoning

Disadvantages

- Increased **logical complexity**
- Higher **computational cost** (especially at scale)
- More challenging to implement and validate

Translation: ASP to L-DINF

Blueprint Persona (ASP)	L-DINF Representation
`patient(P, ...)`	agent identity
`disabled(P)`	$B_i(disabled(P))$ i is an agent who manages the reservations
`preference(P, C)`	$B_i(prefers\_clinic(C))$
`appointment_preference(P, C, S, E)`	$B_i(appointment\_time\_preference(C, S, E))$
`sensory_preference(P, ''noise'')`	$B_i(sensory\_noise\_sensitive)$
`doctor_preference(P, T, S, Y)`	$B_i(doctor\_preference(T, S, Y))$
`distance(P, C, D)`	$B_i(distance\_to\_clinic(C, D))$
`availability(C, T)`	$B_i(can\_do_i(slot(C, T)))$
`alternative(C1, T1, C2, T2)`	$B_i(Cl(slot(C1, T1), slot(C2, T2)))$

Use Case: Mario the Proactive Agent

Initial Setup:

- Mario has chronic conditions and noise sensitivity
- Preferences:
  - prefers_clinic = c1, preferred time = 08:30
  - doctor preference = George Peterson (GP), (Chronic disease specialist)
- Beliefs and preferences encoded as:

```
B_mario(prefers_clinic(c1)) ...
B_mario(accessible(c1) ∧ distance_to_clinic(c1, 12)
∧ doctor_available(GP,c1) → can_do_mario(slot(c1, 0830)))
```

Disruption: Clinic c1 becomes inaccessible ⇒ +(¬accessible(c1)) ⇒ Mario infers ¬can_do_mario(slot(c1, 0830)) ⇒ triggers goal revision

Adaptation Strategy: Mario looks for equivalent alternatives

```
B_mario(Cl(slot(c1, 0830), slot(c2, 0930)))
B_mario(pref_do_mario(slot(c2, 0930), 8))
B_mario(fCl_mario(slot(c1, 0830), slot(c2, 0930)))
B_mario(intend_mario(slot(c2, 0930)))
```

Group Change: Mario calls clinic c_2, and Anna, the receptionist, ensures that the consultation can be done. Formally, Mario has to join Anna's group G_2 at clinic c_2, and this leads to updating his beliefs

```
do_mario(joinA(mario, anna))
B_mario((accessible(c2))
B_mario(doctor_available(DocP,c2)) ...
```

The new intention:

```
B_mario(can_do_mario(slot(c2, 0930))
B_mario(do_mario(slot(c2, 0930)))
```

Summary: Belief revision $\rightarrow$ new plan $\rightarrow$ group coordination $\rightarrow$ action execution

Use Case: Belief Revision and Cross-Group Delegation

Scenario: two clinics, $Clinic_A$ and $Clinic_B$, which are two different (agent) groups with their own medical personnel.

- A patient, *Alice*, is associated with $Clinic_A$, which means that the Alice agent is in this group. She is scheduled a consultation in a time slot t_1. She is seriously ill and cannot be moved
- The doctor at $Clinic_A$, Dr. Jones, becomes unavailable due to an emergency. However, Dr. Smith from $Clinic_B$ is qualified and available for the same action.
- So, $Clinic_B$ momentarily lends Dr. Smith to $Clinic_A$ to do the consultation.

This example demonstrates two key L-DINF capabilities:

- Belief revision: agents dynamically update their internal knowledge when changes occur.
- Delegation across groups: a capable agent from one group is temporarily lent to another to perform an action that no local agent can perform.

Initial Beliefs and Agents Capabilities:

`B_alice(needs_consultation(t1))`
`B_docJ(can_do(consultation(t1)))`

Disruption: Dr. Jones can not do that action anymore
$\Rightarrow +(\neg can\_do\_docJ(consultation(t1)))$
$\Rightarrow B\_alice(\neg can\_do\_docJ(clinic_A, consultation(t1)))$

Delegation and New plan:
$lend\_clinic_A(j, clinic_B, consultation(t_1)) \leftarrow$
$\forall i \in clinic_A \ \neg can\_do_i(consultation(t_1)) \wedge$
$\exists j \in clinic_B \ can\_do_j(consultation(t_1))$

`B_docS(can_do(consultation(t1)))`
$\texttt{B\_i(lend\_clinic}_A(docS, clinic_B, consultation(t1)))$
$\texttt{B\_docS(join}_A(docS, docJ))$
`B_docS(do_docS(consultation(t1)))`
$\texttt{B\_alice(can\_do\_alice(clinic}_A, consultation(t1)))$

Background: DALI

Overview

- DALI is a logic programming language for multi-agent systems that extends Horn-clause logic with event-driven constructs.
- It enables agents to be both **reactive** (responding to environment) and **proactive** (pursuing goals).

The Four Event Types DALI agents classify events to manage reasoning and memory:

1. **External (E)**: Environmental changes or messages from other agents (e.g., `alarmE`).
2. **Internal (I)**: Triggers derived from the agent's own reasoning, representing intentions (e.g., `needVisitI`).
3. **Present (N)**: Events currently being deliberated/processed.
4. **Past (P)**: A record of actions performed, enabling reflection.

Operationalizing Cognitive Agents (L-DINF)

In the context of Medical Scheduling, DALI provides the **executable semantics** for the L-DINF logic model.

- **Reactivity**: Uses rules like *eventE* :> *actionA* to handle immediate triggers.
- **Belief Revision**: Beliefs are stored as dynamic facts that can be updated (matching L-DINF mental actions).
- **Communication**: Supports FIPA ACL for inter-agent delegation and lending.

Mapping L-DINF to DALI

L-DINF Concept	**DALI Implementation**
Belief (B_i)	Dynamic facts in Knowledge Base
Intention ($intend_i$)	Internal events triggering goals
Feasibility (can_do_i)	Guarded rules with preconditions
Delegation ($lend_G$)	Reactive delegation event patterns

From L-DINF to DALI

1. Reactive Trigger (Event → Goal)

- *Logic*: Perceiving a need creates an intention.
- *DALI*: External event (E) triggers an internal Goal (G).

$needs\_consultationE(T) :> goalG(get\_consultation(T))$.

2. Belief Revision (The Cognitive Aspect)

- *Logic*: Update belief $B_{Alice}(\neg can\_do(doc_J))$.
- *DALI*: React to `unavailable` event by retracting a fact.

$unavailableE(D, T) :> retract(can\_do\_docJ\_A(T))$.

3. Cross-Group Delegation

- *Logic*: $lend\_Clinica_A(doc_S, Clinic_B, consultation(t_1))$.
- *DALI*: Request help from an external agent.

$start\_delegation(T) :> request\_help(docS, clinicB, T)$.

Conclusions

✔ **Summary:** We propose an Hybrid Architecture ASP + L-DINF:

- **ASP**: for global optimization
- **L-DINF**: for runtime intelligence
- Combined: responsive and explainable scheduling
- Stronger alignment with real clinical dynamics
- **DALI** prototype

ℹ Future: quantitative evaluation, trust-aware delegation and human-in-the-loop experimentation

Thank You!

valentina.pitoni@univaq.it

Resilient, Context-Aware Nesy Logical Agents, and their Application in Virtual Environments

Stefania Costantini Giovanni De Gasperis Alina Vozna
Università de L'Aquila *Università di Pisa*

XRIA 2025 @ 28th European Conference on Artificial Intelligence
Bologna, October 25, 2025

Logical Agents: Pitfalls

- Logic-based agent-oriented programming languages: formal precision, **verifiability**, **explainability**, and robust **planning** capabilities.
- Dynamic or unpredictable environments: minor deviations or unforeseen inputs can **lead to failure**.
- **Brittleness**: stems primarily from a reliance on static knowledge bases and fixed inference rules.

Generative Agents

Several recent works propose leveraging LLMs to enhance agent capabilities, by using LLMs as black-box planners, environment simulators, or direct controllers

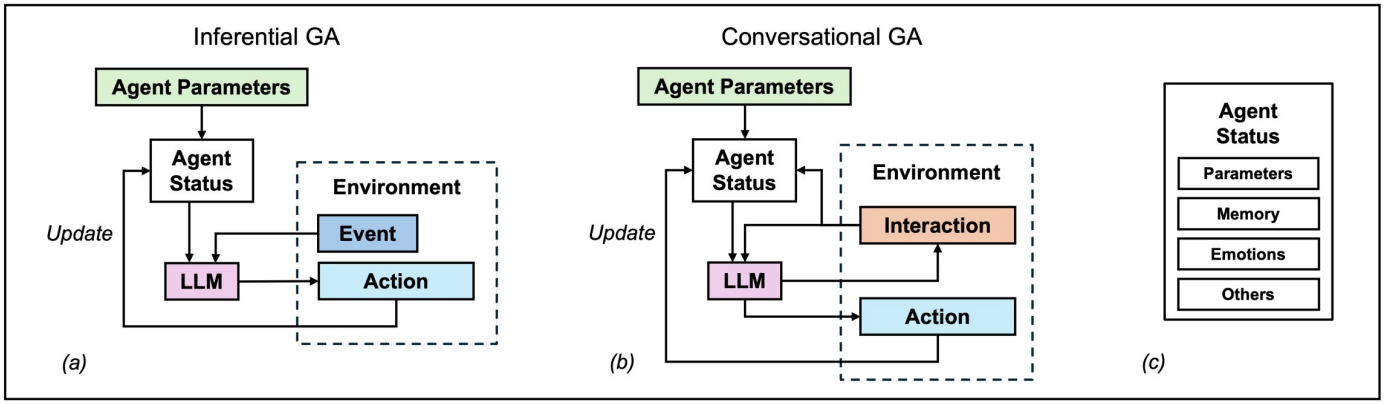


Fig. 2. LLM-Based Feedback loop and Status of GAs: *(a)* Inferential GA feedback loop where the agent's LLM receives the status of the agent and an event as inputs; LLM produces an action; such action updates the agent's current status. *(b)* Conversational GA feedback loop where the LLM decides based on the agent's status and natural language interactions. Such interactions can directly affect the status. *(c)* Example of an Agent Status content.

In these architectures, agents delegate the entire perception-reasoning-action loop to the LLM, reducing the symbolic agent to a thin prompting-and-local-memory interface.

Our proposal: Key features

- **Preservation of autonomy:** Agents retain control over reasoning and decision-making.
- **Dynamic knowledge acquisition:** Agents can incorporate context-dependent information at runtime.
- **Logical adaptability:** New beliefs and reactive behaviors are added consistently to the existing rule base.
- **Emergent behavior modeling:** Agents can learn new goals (with caution!) or derive reactive rules from externally acquired knowledge.
- **Fully transparent, interpretable, and efficient decision-making processes.**
- **Significantly enhanced viability of logical agents in real-world, evolving environments.**
- **Prototypical implemantation: DALI language.**

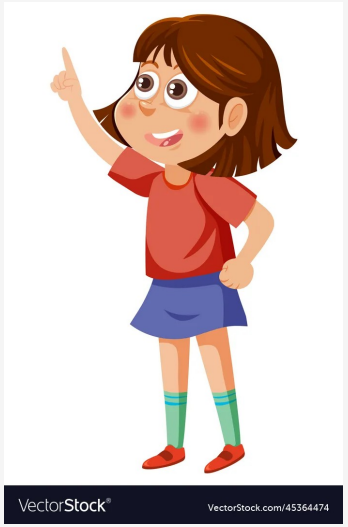
Example 1: A Mixed-Reality Museum Guide

```
object_pointed_atE(Obj)
            :> identify(Obj).
identify(Obj):-
      known_object(Obj),
      explain(Obj).
identify(Obj):-
      unknown_object(Obj),
      manage_unknown_object(Obj).

object_detectedE(sundial42).

known_object(sundial42). % OR

manage_unknown_object(Obj):-
    askLLMA(Obj),
    get_responseA(sundial42),
    provide_responseA(sundial42,ThisUser).
    % Response personalized
    % on the User's profile and
    % preferences!
    % Provided through an LLM
    % in Natural Language
```



Example 2: Adaptive Reasoning in Immersive Environments

Example 2: Adaptive Reasoning in Immersive Environments

Simulation-generated events


```
sunsetE :>
  adjust_lightingA(dimmed),
  suggestA("Would you like
           to explore the
           night attractions?",
           Ans),
  elaborate(Ans).
```

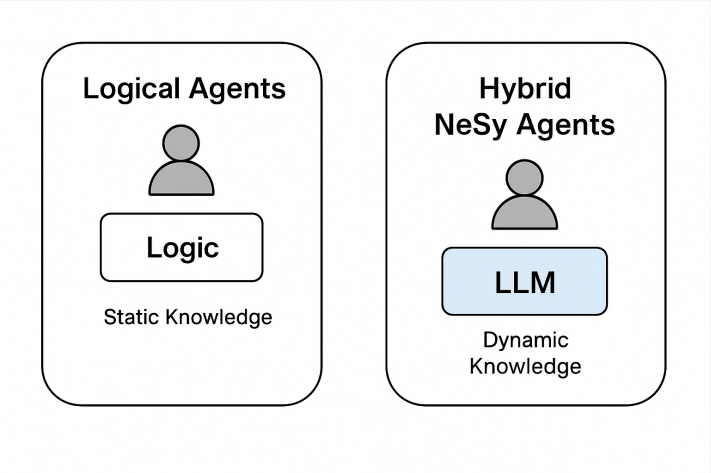
Example 2: Adaptive Reasoning in Immersive Environments (cont'd)

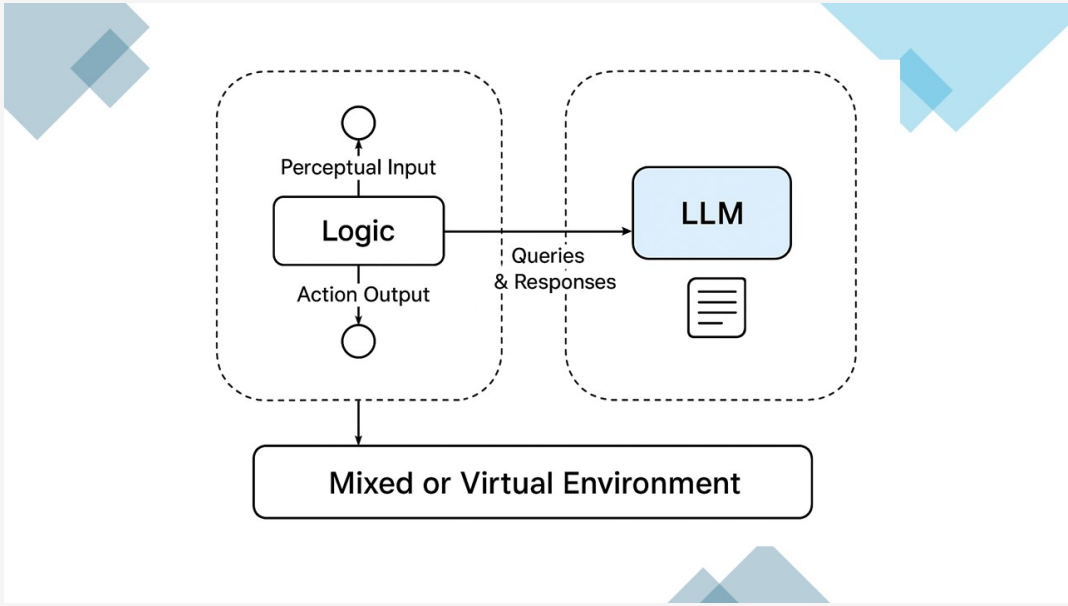
Meeting NPCs


```
npc_arrivalE(Character) :>
    evaluate(Character),
    % according to the User's
        interests
    greet(Character),
    offer_interaction(Character).

npc_utteranceE(Character,Utterance) :>
    evaluate(Utterance,Val),
    worthwhile(Val),
    % according to the User's
        interests
    offer_interaction(Character).
```

The Architecture (visual)





The Architecture (1)

Agent Input

- External systems (e.g., Unity, Unreal, or a sensor simulator) detect some event (e.g., user gesture, object detected, NPC arriving).
- These are turned into DALI external events with the E suffix.
- The events are sent to a DALI agent via the Linda tuple space or directly into its input queue through the PyDALI middleware.
- Thus, the agent can process environmental input as if it came from a physical sensor.

Action Output: Enacting Behavior in the Environment

- To let the agent act in the environment, actions are triggered via DALI logic rules.
- The PyDALI middleware listens for actions like $moveA(\ldots)$, etc.
- These are mapped to commands understood by the virtual world engine (e.g., Unity script or VR API call).
- The external system enacts the action.

The Architecture (2): the role of LLM

- **DALI agents can get factual knowledge or new reactive rules**
- **DALI agents can interact with a user in natural language**

- **Problem 1: Obtaining answers from an LLM in a desired format**
  - It has been shown that LLMs are able to generalize from the examples to generate similar responses for new inputs (example-based prompting).
  - We need highly structured, formatted outputs (Prolog-like facts) aligned to domain-specific or task-specific conventions.
- **Problem 2: Check against hallucinations**
  - Carefully formalize prompts (experiments already performed).
  - Cross-check acquired knowledge w.r.t. existing one (past work).
  - Use more than one LLM and compare results.
  - ...

Our Contributions

We extended symbolic (DALI) agents to autonomously detect context changes via internal inference or perceptual input

- We introduced a modular interaction mechanism that enables selective delegation of perception and knowledge acquisition tasks to external systems.
- We aim to refine our methods for logically incorporating external information into the agent's belief base while preserving consistency and ensuring actual usefulness.

We maintain symbolic verifiability and transparency while enabling dynamic, runtime adaptation

- We demonstrated how our NeSy agents can operate within virtual environments, leveraging LLMs to enhance their flexibility and support more human-like interactions.
- We described an implementation based on *PyDALI*.

Conclusion

Our framework, already prototypically implemented, alleviates brittleness, without compromising the internal reasoning structure of the agents

Next/current work?

- **Large-Scale Evaluation**
- **Multimodal Perception Integration**
- **Support for Other Agent Frameworks**
- **Trust and Reliability Management**
- **Real-World Deployment**
- ...