Runtime Verification of Agent Properties

Stefania Costantini and Pangiota Tsintza
University of L’Aquila, Italy

Pierangelo Dell’Acqua
Linköping University, Sweden

Luís Moniz Pereira
New University of Lisbon, Portugal

INAP09 - Évora, Portugal - November 2009
Verification of Properties in Agents

• As agent systems are more widely used in real-world applications, the issue of verification is becoming increasingly important:
  - A priori verification: Model Checking & Theorem Proving
  - Dynamic verification: properties verified at runtime

• The two approaches are complementary rather than compete
How to verify in agents that a property $\varphi$ holds?

Prove that $\varphi$ holds in any future state of the agent:

- One can verify $\varphi$ by explicitly examining all possible future states
- One can perform a run-time verification of $\varphi$
  and suitable counter-measures can be undertaken in case of violation
Agents’ Abilities

- Agents:
  - software entities
  - interact with each other and with the environment
  - proactive and deliberative capabilities
  - able to learn by observing other agent behavior or by imitation
  - subject to modify themselves and evolve due to external/internal stimuli
Motivation

- Given the evolving nature of learning agents, their behavior has to be checked from time to time and not only a priori.

- Model checking and other a priori approaches are static and need to re-check whenever the agent learns a new piece of information.

- A priori full validation of agents' behavior would have to consider all possible scenarios that are not known in advance.

- These are the reasons why we propose a run-time control on agent behavior and evolution, for checking correctness during agents’ activity, rather than a model checking control.
Agent Model

> Agent Program

- An *agent program* is a tuple $P = \langle BA, MA, C, CI \rangle$ of software components:
  - $BA$ and $MA$ are logic programs (base and meta agent)
  - $C$ is the control component executor specification, e.g. a meta-interpreter
  - $CI$ contains declarative control information directives
Agent Model

> Evolutionary Semantics

- An agent starts from an initial program $P_0$

- Changes in the environment as well as agent’s own self-modifications are modeled as *program transformation steps*:

  \[
  \langle BA_i, MA_i, C_i, CI_i \rangle \xrightarrow{U(C_i, CI_i, w_i)} \langle BA_{i+1}, MA_{i+1}, C_{i+1}, CI_{i+1} \rangle
  \]

  where $U$ is the *underlying control mechanism* and $w_i$ is the *external environment*
Agent Model

Evolutionary Semantics

- We thus have a Program Evolution Sequence $PE = [P_0, \ldots, P_n]$

- We have a corresponding Semantic Evolution Sequence $ME = [M_0, \ldots, M_n]$ where $M_i$ is the semantic account of $P_i$ according to the specific language and the chosen semantics

- The pair $\langle PE; ME \rangle$ is called the *Evolutionary Semantics* of the agent
Linear-time temporal logic

Future-time connectives

Next state
$X\varphi$ states $\varphi$ will be true at next state

Always in future
$G\varphi$ means $\varphi$ will be true in every future state

Sometime in future
$F\varphi$ states there is a future state where $\varphi$ will be true

Weak until
$\varphi W \psi$ is true in $s$ if $\psi$ is true in $t$, in the future of $s$, and $\varphi$ is true in every state in time interval $[s,t)$

Strong until
$\varphi U \psi$ is true in $s$ if $\psi$ is true in $t$, in the future of $s$, and $\varphi$ is true in every state in time interval $[s,t]

Never
$N \varphi$ states $\varphi$ should not become true in any future state
• Past-time connectives

Last state
\( \hat{X}\varphi \) states that, if there is a last state, \( \varphi \) was true then

Some time in the past
\( \hat{F}\varphi \) states \( \varphi \) was true in some past state

Always in the past
\( \hat{G}\varphi \) states \( \varphi \) was true in all past states

Weak since
\( \varphi \hat{Z}\psi \) is true in \( s \) if \( \psi \) was true in \( t \) (in the past of \( s \)), and \( \varphi \) was true in every state of time interval \( [t,s) \)

Since
\( \varphi \hat{Z}\psi \) is true in \( s \) if \( \psi \) was true in \( t \) (in the past of \( s \)), and \( \varphi \) was true at every state in time interval \( [t,s] \)
I-METATEM logic

- Extends METATEM with intervals:

  \[ U_{m,n}: \text{strong until in a time interval} \]

  \[ \hat{S}_{m,n}: \text{since in a time interval} \]

- Formulae of I-METATEM are defined as:

  \[
  \varphi ::= p \mid \text{true} \mid \neg \varphi_1 \mid \varphi_1 \land \varphi_2 \mid \tau(i)
  \]

  \[
  \varphi ::= X\varphi_1 \mid \varphi_1 U\varphi_2 \mid \hat{X}\varphi_1 \mid \varphi_1 \hat{S}\varphi_2
  \]

  \[
  \varphi ::= \varphi_1 U_{m,n} \varphi_2 \mid \varphi_1 \hat{S}_{m,n} \varphi_2
  \]

  \[
  \varphi ::= (\varphi_1)
  \]

  where \( p \in A_P \) is a proposition, and \( \varphi_1 \) and \( \varphi_2 \) are formulae of I-METATEM
A-IMETATEM

> Semantics of Operators

- A **structure** is a pair \( \langle \sigma, i \rangle \in (\mathbb{N} \rightarrow 2^{AP}) \times \mathbb{N} \)

- Given a timestamp \( j \), let \( \sigma(j) \) be the set of propositions in \( AP \) that are true at time \( j \)

**Propositions and propositional connectives**

\[
\langle \sigma, i \rangle \models p \quad \text{iff} \quad p \in \sigma(i)
\]

\[
\langle \sigma, i \rangle \models \text{true}
\]

\[
\langle \sigma, i \rangle \models \neg \varphi \quad \text{iff} \quad \langle \sigma, i \rangle \not\models \varphi
\]

\[
\langle \sigma, i \rangle \models \varphi \land \psi \quad \text{iff} \quad \langle \sigma, i \rangle \models \varphi \text{ and } \langle \sigma, i \rangle \models \psi
\]

\[
\langle \sigma, i \rangle \models \tau(i)
\]
Temporal connectives

\[ \langle \sigma, i \rangle \models X \varphi \iff \langle \sigma, i+1 \rangle \models \varphi \]

\[ \langle \sigma, i \rangle \models \varphi U \psi \iff \exists k \in \mathbb{N} \, \langle \sigma, i+k \rangle \models \psi \text{ and } \forall j \, (0 \leq j < k) \, \langle \sigma, i+j \rangle \models \varphi \]

\[ \langle \sigma, i \rangle \models \neg X \varphi \iff \text{if } i > 0, \text{ then } \langle \sigma, i-1 \rangle \models \varphi \]

\[ \langle \sigma, i \rangle \models \varphi \hat{S} \psi \iff \exists k \, (1 \leq k \leq i) \, \langle \sigma, i-k \rangle \models \psi \text{ and } \forall j \, (1 \leq j < k) \, \langle \sigma, i-j \rangle \models \varphi \]

Temporal connectives in time intervals

\[ \langle \sigma, i \rangle \models \varphi U_{m,n} \psi \iff i \leq m \leq n, \exists k \, (m - i \leq k \leq n - i) \, \langle \sigma, i + k \rangle \models \psi \text{ and } \forall j \, (0 \leq j < k - i) \, \langle \sigma, i + j \rangle \models \varphi \]

\[ \langle \sigma, i \rangle \models \varphi \hat{S}_{m,n} \psi \iff m \leq n < i, \exists k \, (i - m \leq k \leq i - n) \, \langle \sigma, i - k \rangle \models \psi \text{ and } \forall j \, (1 \leq j < k) \, \langle \sigma, i - j \rangle \models \varphi \]
I-METATEM

> Derived Future-time connectives

future $m$-state
$X_m \varphi$ states $\varphi$ will be true in $s_{m+1}$

bounded eventually
$F_m \varphi$ states $\varphi$ eventually holds somewhere on the path from the current state to $s_m$

bounded eventually in time interval
$F_{m,n} \varphi$ states $\varphi$ eventually holds somewhere on the path from $s_m$ to $s_n$

always in time interval
$G_{m,n} \varphi$ states $\varphi$ should become true at most at $s_m$ and then holds at least until $s_n$

strong always in time interval
$G_{\langle m,n \rangle} \varphi$ states $\varphi$ should become true just at $s_m$ and then holds until $s_n$, and not at $s_{n+1}$

bounded never
$N_{m,n} \varphi$ states $\varphi$ should not be true in any state between $s_m$ and $s_n$

sometime in time interval
$E_{m,n} \varphi$ states $\varphi$ has to occur one or more times between $s_m$ and $s_n$
I-METATEM

> Derived Past-time connectives

\textit{last m-state}

\( \hat{X}_m \varphi \) states \( \varphi \) was true in past state \( s_{m-1} \)

\textit{bounded some time in the past}

\( \hat{F}_m \varphi \) states \( \varphi \) was true in some past state up to \( s_m \), included

\textit{always in time interval in the past}

\( \hat{G}_{m,n} \varphi \) states \( \varphi \) was true in past state \( s_m \) and then remained true at least until past state \( s_n \)

\textit{strong always in time interval in the past}

\( \hat{\tilde{G}}_{(m,n)} \varphi \) states \( \varphi \) became true just in past state \( s_m \) and then remained true exactly until past state \( s_n \)
Basic I-METATEM rules

- Any I-METATEM formula \( \varphi \) is a rule

  The rule is verified or succeeds whenever \( \varphi \) holds, otherwise the rule is violated

  Rule verification requires groundness

- Example: a goal \( g \) that is not achieved cannot be dropped

  \[ N \left( \text{not achieved}(g), \text{dropped}(g) \right) \]

  \[ N_{\text{init},\text{end}} \left( \text{not achieved}(g), \text{dropped}(g) \right) \]

  bounded never
Contextual I-METATEM rules

- A contextual I-METATEM rule takes the form $\chi \Rightarrow \varphi$
  - $\varphi$ is a I-METATEM formula
  - $\chi$ is the evaluation context of the rule, and consists of a conjunction of logic programming literals
  - every variable occurring in $\varphi$ must occur in the context $\chi$

- Example

  \[
  [\text{goal}(\text{Goal}), \text{priority}(\text{Goal}, \text{Pr}), \text{timeout}(\text{Pr}, T_{out})] \Rightarrow F(T_{out}) \text{ achieved}(\text{Goal})
  \]

  a goal with timeout $T_{out}$ should be accomplished before the timeout
Contextual I-METATEM rules with improvement/repair

- An I-METATEM rule with improvement takes the form $\chi \Rightarrow \varphi: A$
  - $\chi \Rightarrow \varphi$ is a contextual rule
  - the atom $A$ is the improvement action of the rule
  - when the monitoring condition $\varphi$ holds, the improvement action $A$ is attempted
    $A$ is executed as an ordinary logic programming goal, and it assumes a declarative semantics
Contextual A-IMETATEM rules with improvement

> Example

\[
[\text{goal}(\text{Goal}), \text{deadline}(\text{Goal}, T), \text{now}(T2), T2 \leq T] \Rightarrow \\
E(1, 12)(\text{not achieved}(\text{Goal}) \land \text{dropped}(\text{Goal})) : \text{inc\_comt}(T2)
\]

If a goal not achieved is dropped sometime in the interval (1,12), then increase commitment level
Contextual A-IMETATEM rules with improvement

> Example

- I-METATEM used to check the past behavior and knowledge of the agent to influence its future behavior

\[
[ \text{now}(T), \text{goal}(\text{Goal}), \text{executed}(\text{Goal}), \text{consequence}(\text{Goal}, C) ] \Rightarrow \\
\hat{F}_{0,T} \text{ not desired}(C) : \text{assert}( [ \text{now}(T), \text{threshold}(T1) ] \Rightarrow N(T, T1) \text{ exec}(\text{Goal}) )
\]

If a goal $G$ had some undesired consequences, then the goal cannot be further pursued, at least until a certain time has elapsed.
Related Work

> SCIFF Abductive Proof Procedure

- Proposed by Lamma et al. in the '80s

- Abductive logic programming language for the specification and run-time verification of interaction protocols

- Most of I-METATEM connectives can be expressed by SCIFF rules except for the *until* connective

- I-METATEM connectives can be composed in several ways with both temporal and logical connectives

  This is not possible with SCIFF rules
Related Work

> LTLeC

- Proposed by D’Souza

- A variant of timed linear time temporal logic (TLTL)

- Its syntax includes the two new atomic formulae:

  \[ \triangleleft_a \in I \]

  asserts that the time since the event \( a \) has occurred last time, lies within interval \( I \)

  \[ \triangleright_a \in I \]

  asserts that the time until \( a \) occurs again lies within \( I \)
Related Work

> LTLeC

Example

\[ G(\text{request} \rightarrow \triangledown_{\text{acknowledge}} \in [0, 5]) \]

if a request event arrives, then it must be handled with an acknowledge event within 5 time units

It can be expressed in I-METATEM as:

\[ [\text{now}(T)] \Rightarrow G(\text{request} \Rightarrow F_{0,T+5} \text{acknowledge}) \]

Thus, I-METATEM logic is at least as expressive as the LTLeC logic
Our conjecture is that the two logics have the same expressive power. For example, the I-METATEM connective $U_{m,n}$ can be represented in LTLeC as:

\[
\begin{align*}
\langle \sigma, i \rangle \models \varphi U_{m,n} \psi & \quad (i \leq m \leq n) \\
\equiv & \\
\langle \sigma, i \rangle \models \bigvee_{x=m}^{n} (\triangleright \psi \in [x, x] \land \Gamma)
\end{align*}
\]

where:

\[
\Gamma = \begin{cases} 
true & \text{if } x - 1 < i \\
\land_{y=i}^{x-1} (\triangleright \varphi \in [y, y]) & \text{otherwise}
\end{cases}
\]
Contribution

- Temporal logic with connectives defined over intervals to verify the run-time behavior of agents evolution

- Improvement actions that can modify the agent’s knowledge base underlying agent model based on meta-levels