Self-checking Logical Agents

(Extended Abstract)

Stefania Costantini
Department of Computer Science and Engineering, and Mathematics
University of L'Aquila
Via Vetoio Loc. Coppito, I-67100 L'Aquila, Italy
stefania.costantini@univaq.it

ABSTRACT

This paper presents a comprehensive framework for run-time self-checking of logical agents, by means of temporal axioms to be dynamically checked. These axioms are specified via an agent-oriented interval temporal logic defined to this purpose, with fully defined syntax, semantics and pragmatics.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence, Distributed Artificial Intelligence, Intelligent Agents

General Terms

Languages, Theory

Keywords

Agent development techniques, Tools and environments, Logic-based approaches and methods, Reasoning (single and multi-agent)

1. INTRODUCTION

Agent systems are more and more widely used in real-world applications: therefore, the issue of verification is becoming increasingly important. This paper presents an approach to dynamic (run-time) verification of agent systems (as opposed, or, better, as a complement to "static" verification, i.e., verification performed prior to agent activation). A crucial point of our proposal is that, in case of violation of a wished-for property, agents should try to restore an acceptable or desired state of affairs by means of run-time *self-repair*. Even in case desired properties are fulfilled, by examining relevant parameters of its own activities an agent might apply forms of *self-improvement* so as to perform better in the future. Self-repair and improvement should alleviate the problem of "brittleness", that can be intended as the propensity of an agent to perform poorly or fail in the face of circumstances not fully or explicitly considered by the agent's designer.

We introduce meta-constraints to be added to agent programs and to be checked dynamically, at a certain (customizable) frequency. They are based upon a simple interval temporal logic particularly tailored to the agent realm, A-ILTL (Agent-oriented Interval Temporal Logic). Thus, properties can be defined that should hold according to what has happened and to what is supposed to happen or not to happen in the future, also considering partially specified event sequences.

Our approach is fairly general, and thus could be adopted in several logic agent-oriented languages and formalisms. In particular, one such language is DALI (cf. [2] for a list of references about DALI and the DALI interpreter, which is publicly available) where we have prototypically implemented the approach.

The reader may refer to a preliminary longer version of this paper [1] for the full formalization of the proposed approach, a complete example of application and a discussion and comparison with related work. The author apologizes for missing discussion and references, due to lack of space.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

2. A-ILTL

For defining properties that are supposed to be respected by an evolving system, a well-established approach is that of Temporal Logic, and in particular of Linear-time Temporal Logics (LTL). These logics are called 'linear' because (in contrast to 'branching time' logics) they evaluate each formula with respect to a vertex-labeled infinite path (or "state sequence") $s_0 s_1 \ldots$ where each vertex s_i in the path corresponds to a point in time (or "time instant" or "state"). In what follows, we use the standard notation for the best-known LTL operators.

Based upon our prior work, in [1] we formally introduced an extension to the well-known linear temporal logic LTL based on *intervals*, called A-ILTL for 'Agent-Oriented Interval LTL'. Though, as discussed in [1], several "metric" and interval temporal logic exist, the introduction of A-ILTL is useful in the agent realm because the underlying discrete linear model of time and the complexity of the logic remains unchanged with respect to LTL. This simple formulation can thus be efficiently implemented, and is nevertheless sufficient for expressing and checking a number of interesting properties of agent systems.

3. A-ILTL IN AGENT PROGRAMS

A-ILTL operators can be introduced into logic agent-oriented programming languages. A *pragmatic* form (that namely we have adopted in DALI) can be $OP(m, n; k)\,\varphi$: m, n define the time interval where (or since when, if n is omitted) expression $OP\,\varphi$ is required to hold, and k (optional) is the frequency for checking whether the expression

actually holds. For instance, $EVENTUALLY(m, n; k)\,\varphi$ states that φ should become true at some point between time instants m and n.

In rule-based logic programming languages like DALI, we restrict φ to be a conjunction of literals. In pragmatic A-ILTL formulas, φ must be ground when the formula is checked. However, we allow variables to occur in an A-ILTL formula, to be instantiated via a *context* χ (we then talk about *contextual A-ILTL formulas*). Notice that, for the evaluation of φ and χ, we rely upon the procedural semantics of the 'host' language.

In the following, a contextual A-ILTL formula τ will implicitly stand for the ground A-ILTL formula obtained via evaluating the context. In [1] we have specified how to *operationally* check whether such a formula holds. This by observing that for most A-ILTL operators there is a *crucial state* where it is definitely possible to assess whether a related formula holds or not in given state sequence, by observing the sequence up to that point and ignoring the rest.

The following formulation deals with agent run-time self-modification.

DEFINITION 3.1. *An A-ILTL rule with repair/improvement is of the form (where M, N, K can be either variables or constants)*

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

where:(i) $OP(M, N; K)\varphi :: \chi$ is a contextual A-ILTL formula, called the monitoring condition*; (ii) η is called the* repair action *of the rule, and it consists of an atom η; (iii) ξ (optional) is called the* improvement action *of the rule, and it consists of an atom η.*

Whenever the monitoring condition (automatically checked at frequency K) is violated, the repair action η is attempted. If instead the monitoring condition succeeds, in the sense that the specified interval is expired and the given A-ILTL formula holds or, in case of the operator 'eventually', if φ holds within given interval, then the improvement action, if specified, can be 'executed'.

Take for instance the example of one who wants to lose some weight by a certain date. If (s)he fails, then (s)he should undertake a new diet, with less calories. But if (s)he succeeds before the deadline, then a normocaloric diet should be resumed. In the proposed approach, this can be formalized as follows (where, as there are no variables, context is omitted):

$EVENTUALLY(May{-}15{-}2012, June{-}10{-}2012)$
$lose\_five\_kilograms$
$\div\ new\_stricter\_diet(June{-}10{-}2012, June{-}30{-}2012)$
$\div\ resume\_normal\_diet$

4. EVOLUTIONARY EXPRESSIONS

It can be useful to define properties to be checked upon arrival of event sequences, of which however only relevant events (and their order) should be considered. To this aim we introduce a new kind of A-ILTL rules, that we call *Evolutionary A-ILTL Expressions*. To define partially known sequences of any length, we admit for event sequences a syntax inspired to that of regular expressions so as to specify irrelevant/unknown events, and repetitions (cf. [1]).

DEFINITION 4.1 (EVOLUTIONARY LTL EXPRESSIONS). *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 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".*

An Evolutionary LTL Expression can be evaluated w.r.t. a state s_i which includes among its components the *history* of the agent, i.e., the list of past events perceived by the agent. A history H *satisfies* an event sequence S whenever all events in S occur in H, in the order specified by S itself.

DEFINITION 4.2. *An Evolutionary A-ILTL Expression ϖ, of the form specified in Definition 4.1: (1)* holds *in state s_i whenever (i) history H_i satisfies $\mathcal{S}^{\mathcal{E}vp}$ and $\mathcal{S}^{\mathcal{F}}$ and does not include any event in $\mathcal{J}^{\mathcal{J}}$, and τ holds or (ii) H_i includes any event occurring in $\mathcal{J}^{\mathcal{J}}$ (the expression is* broken*); (2) is* violated *in state s_i whenever H_i satisfies $\mathcal{S}^{\mathcal{E}vp}$ and $\mathcal{S}^{\mathcal{F}}$ and does not include any event in $\mathcal{J}^{\mathcal{J}}$, and τ does not hold.*

Operationally, an Evolutionary A-ILTL Expression can be finally deemed to hold if either the critical state has been reached and τ holds, or an unwanted event has occurred. Instead, an expression can be deemed *not* to hold (or, as we say, to be *violated* as far as it expresses a wished-for property) whenever τ is false at some point without the occurrence of breaking events.

The following is an example of Evolutionary A-ILTL Expression stating that, after submitting a car to a checkup, it is guaranteed to work properly for six months, even in case of (repeated) long trips, unless an accident occurs.

$checkup_P(Car) : T :$
$ALWAYS(T, T + 6_{months})\ \ work\_ok(Car)$
$::: long\_trip(Car)*$
$:::: accident(Car)$

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

DEFINITION 4.3. *An evolutionary LTL expression with repair ϖ^r is of the form $\varpi|\eta_1||\eta_2$ where ϖ is an Evolutionary LTL Expression adopted in language $\mathcal{L}$, and η_1, η_2 are atoms of $\mathcal{L}$. η_1 will be executed (according to $\mathcal{L}$'s procedural semantics) whenever ϖ is violated, and η_2 will be executed whenever ϖ is broken.*

In previous example, for restoring $work\_ok(Car)$, η_1 might imply asking for a free car repair (under guarantee) and η_2 might imply resorting to the insurance company.

5. REFERENCES

[1] S. Costantini. Self-checking logical agents. In *Proc. of LA-NMR 2012*, Volume 911. CEUR Workshop Proceedings (CEUR-WS.org), 2012. Invited paper.

[2] S. Costantini. The DALI agent-oriented logic programming language: References, 2013. at URL http://www.di.univaq.it/stefcost/info.htm.