Event-Action Modules for Complex Reactivity in Logical Agents

(Extended Abstract)

Stefania Costantini, Univ. of L'Aquila, Italy and Régis Riveret, Imperial College London, UK

Categories and Subject Descriptors

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

Keywords

Agent development techniques, tools and environments, Logic-based approaches and methods, Reasoning (single and multi-agent), Machine learning, Complex-event processing.

1. INTRODUCTION

Event processing (also called CEP, for "Complex Event Processing") has emerged as a relevant new field of software engineering and computer science [2]. In fact, a lot of practical applications have the need to actively monitor vast quantities of event data to make automated decisions and take time-critical actions (cf. the Proceedings of the RuleML Workshop Series).

Complex Event Processing is particularly important in software agents. Naturally, most agent-oriented languages, architectures and frameworks are to some extent event-oriented and are able to perform event-processing. In particular our approach is concerned with logical agents, i.e., agents whose syntax and semantics is rooted in Computational Logic (for a recent survey cf., e.g., [7] and for a recent open source CEP system see e.g. ETALIS [1]).

We propose a novel conceptual view of complex events and a possible formalization of the new concepts. We observe that a complex event cannot always result from deterministic incremental aggregation of simple events. Rather, an agent should be able to possibly interpret a set of simple events in different ways, and to assign/learn a plausibility and reliability of each interpretation. To this aim we propose special modules, illustrated below.

1.1 Event-Action Modules

The following illustrates an *Event-Action Module* evaluating symptoms of either pneumonia, or just flu, or both (clearly, we do not aim at medical precision). We adopt syntax and keywords only for illustration purposes. In the sample syntax, postfix 'P' indicates *past events*, i.e., events which have occurred (after the ':' there is the time-stamp or the interval of occurrence). Postfix 'A' indicates *actions*. Connective :> indicated Event-Condition-Action rules, while :- is the usual prolog *if*. An Event-Action Module will be activated whenever the *triggering* events occur within a certain time interval, and according to specific conditions: in the example, whenever in the last two days both high temperature and intense cough have been recorded.

The module is re-evaluated every time that a new occurrence of the triggering events should arrive.

EVENT-ACTION-MODULE

$TRIGGER$
$(high\_temperatureP\ AND\ intense\_coughP) : [2days]$
$COMPLEX\_EVENTS$
$suspect\_flu\ OR\ suspect\_preumonia$
$suspect\_flu$:- $high\_temperatureP.$
$suspect\_pneumonia$:- $high\_temperatureP : [4days],$
$intense\_coughP.$
$PREFERENCES$
$suspect\_flu$:- $patient\_is\_healty.$
$suspect\_pneumonia$:- $patient\_is\_at\_risk.$
$ACTIONS$
$suspect\_fluI$:> $stay\_in\_bedA.$
$suspect\_fluI,$
$high\_temperatureP : [4days],$
$not\ suspect\_pneumonia$:> $assume\_antibioticA.$
$suspect\_preumoniaI$:> $assume\_antibioticA,$
$consult\_lung\_doctorA.$
$MANDATORY$
$suspect\_preumonia$:- $high\_temperatureP : [4days],$
$suspect\_fluP,$
$assume\_antibioticP : [2days].$

From given symptoms, either a suspect flu or a suspect pneumonia or both can be inferred, though for suspecting pneumonia high temperature should have lasted for at least four days, accompanied by intense cough. This is stated in the *COMPLEX_EVENTS* section, where each of the listed complex events (in the example, *suspect_flu* and *suspect_pneumonia*) can be inferred, though according to the specified conditions. Notice that the whole agent's belief base (including the history, and namely all past events that have been recorded) is implicitly included in the definition of an Event-Action Module. Explicit preferences are expressed, stating that hypothesizing a flu should be preferred in case the patient is healthy, while pneumonia is more plausible for risky patients. If either none or both options hold, then they are equally preferred. More involved forms of preferences might be specified, that for lack of space we do not discuss here (cf., e.g., [4] and the references therein). Actions to undertake in the two cases are specified. As mentioned, the module is re-evaluated at subsequent new occurrences of high temperature and intense cough. Re-evaluation is performed on the (possibly) updated belief base, i.e., on the belief base which is actual when new event occurrences are perceived.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

Actions will be performed depending upon which of the different possible scenarios the agent prefers to adopt. In particular, if a flu is suspected then the patient should stay in bed, and if the high temperature persists then an antibiotic should also be assumed. In case of suspect pneumonia, an antibiotic is mandatory, plus a consult with a lung doctor.

The *MANDATORY* section of the module specifies constraints: in the example, some complex events must necessarily be inferred in module (re)evaluation if certain conditions occur. Namely, it is stated that pneumonia must to be hypothesized whenever flu has been previously assumed, but high temperature persists despite at least two days of antibiotic therapy.

There are other features of Event-Action Modules that for lack of space we cannot discuss here. In particular, it is possible to reason about (approximations of) possibility and necessity [3]. In the example, it may be for instance that clinical history and conditions of a patient force to assume a pneumonia. Anomaly management can also be expressed.

Each Event-Action Module can be translated in a fully automated way into an Answer Set Programming (ASP) module [8, 3], where ASP is a well-established logic programming paradigm where a program have a number (none, one or several) answer sets expressing possible solutions. This except sections *ACTIONS*, *ANOMALY_MANAGEMENT_ACTIONS* and *PRECONDITIONS*, which are sets of simple reactive rules and action preconditions, to be added to the main agent program. Translation into ASP has two aspects: (i) providing a formal semantics to Event-Action Modules, as ASP is fully logical; (ii) making Event-Action Modules executable, as ASP is fully implemented, and as many efficient implementations are publicly available.

2. EVALUATING OUTCOMES VIA REINFORCEMENT LEARNING

Outcomes of an Event-Action Module are not always univocal: thus, at each stage the agent has to choose among the answer sets of the corresponding ASP module. Though, as we have seen in the example, preferences may help the agent in choosing an outcome rather than another one, knowledge can be incomplete or partial about reliability of such preferences, and in general about plausibility of the choice.

To address this issue, we consider a self-improving process so that an agent will learn over time to make the 'best' choice over the answer sets of each module Π (i.e., over the outcomes of each Event-Action Module). For this purpose, an agent is endowed with a simple reinforcement learning mechanism: at each evolution step, occurring at a time t, an agent: (i) senses its environment; (ii) evaluates Π obtaining the answer sets $M_1, \ldots, M_k$; (iii) adopts one of them, say M, and re-evaluates its "quality" $Q(M)$. In order to evaluate the quality of an answer set, we assume that at stage t, a numerical value denoted $V^t(M)$ can associated to it: this value can be either epistemic or practical, but in any case its expression shall be dependent on the application so we leave it unspecified in the description of the present general framework. At time t, the quality of the selected model M will be computed as a discounted moving average of its value over time:

$$Q^{t+1}(M) = Q^t(M) + \alpha.(V^t(M) - Q^t(M))$$

Then, an agent will draw an answer set M with probability $P^t(M)$ amongst all the alternative sets $M_1, \ldots, M_k$ using a Gibbs-Boltzmann distribution:

$$P^t(M) = e^{Q^t(M)/\tau} / \sum_i e^{Q^t(M_i)/\tau}$$

where τ is a 'learning temperature' to balance the exploitation and the exploration of possible models. To merge this reinforcement learning mechanism with preferences, a possibility is to modify the choice mechanism: instead of 'sharply' adopting the answer set with highest probability, the most preferred among the best rated ones can be selected.

3. CONCLUDING REMARKS

In future work, other event aggregation and recognition patterns might be introduced. The approach might be extended to the definition of complex actions, and to the choice among possible action patterns. Learning mechanisms might be refined based on experimentations. We also intend to introduce 'deeper' forms of learning: Event-Action Modules might be defined in a tentative or embryonic form, then module elements might be learnt via a suitable training phase, and refined according to agent's subsequent significant experiences. At present there is no full implementation of the proposed approach, though its most important features have been implemented or simulated using the DALI language [5, 6]. Implementation and experiments are an important future objective.

4. ACKNOWLEDGMENTS

Part of this work is supported by the Marie Curie Intra-European Fellowships PIEF-GA-2012-331472.

5. REFERENCES

[1] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic. Real-time complex event recognition and reasoning-a logic programming approach. *Applied Artificial Intelligence*, 26(1-2):6–57, 2012.

[2] M. K. Chandy, O. Etzion, and R. von Ammon. 10201 Executive Summary and Manifesto – Event Processing. In K. M. Chandy, O. Etzion, and R. von Ammon, editors, *Event Processing*, number 10201 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2011. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[3] S. Costantini. Answer set modules for logical agents. In O. de Moor, G. Gottlob, T. Furche, and A. Sellers, editors, *Datalog Reloaded: First International Workshop, Datalog 2010*, volume 6702 of *LNCS*. Springer, 2011. Revised selected papers.

[4] S. Costantini and G. D. Gasperis. Complex reactivity with preferences in rule-based agents. In A. Bikakis and A. Giurca, editors, *Research and Applications - 6th Intl. Symp., RuleML 2012, Proceedings*, volume 7438 of *Lecture Notes in Computer Science*, pages 167–181. Springer, 2012.

[5] S. Costantini and A. Tocchio. A logic programming language for multi-agent systems. In *Logics in Artificial Intelligence, Proc. of the 8th Europ. Conf.,JELIA 2002*, LNAI 2424. Springer-Verlag, Berlin, 2002.

[6] S. Costantini and A. Tocchio. The DALI logic programming agent-oriented language. In *Logics in Artificial Intelligence, Proc. of the 9th European Conference, Jelia 2004*, LNAI 3229. Springer-Verlag, Berlin, 2004.

[7] M. Fisher, R. H. Bordini, B. Hirsch, and P. Torroni. Computational logics and agents: a road map of current technologies and future trends. *Computational Intelligence Journal*, 23(1):61–91, 2007.

[8] M. Gelfond. Answer sets. In *Handbook of Knowledge Representation*, chapter 7. Elsevier, 2007.