Parametric Runtime Verification of Multiagent Systems (Extended Abstract)

Davide Ancona* Angelo Ferrando Viviana Mascardi
DIBRIS, University of Genova, Italy
angelo.ferrando@dibris.unige.it, davide.ancona@unige.it, viviana.mascardi@unige.it

1. INTRODUCTION

Parametricity [14] is an important feature of a monitoring system for making runtime verification (RV) more effective, since, typically, correctness of traces depends on the specific data values that are carried by the monitored events of the trace, and that, in general, cannot be predicted statically. Typically, the correctness of an interaction protocol may depend on the values exchanged by agents; protocols may also be parametric in the involved agents, and resources, and this parametricity is naturally reflected on the data carried by values.

In this work we propose *parametric trace expressions*, an extension to *trace expressions* [7, 2], expressly designed for parametric RV of multiagent systems. Such an extension is achieved by introducing variables in trace expressions that are substituted with data values at runtime, when events are matched during monitoring.

2. PARAMETRIC TRACE EXPRESSIONS

Trace expressions [7, 2] are a specification formalism expressly designed for RV and inspired by initial work on monitoring of agent interactions in multiagent systems [6, 11].

They have been successfully adopted in practice for both the JADE and Jason [8] platforms, as discussed in [3, 9, 10, 4, 5, 12], and applications for other interesting domains have emerged recently [1]. In particular, several experiments based on real case studies have demonstrated that in most practical examples monitoring through trace expressions exhibits a time complexity which is linear in the length of the analyzed event trace.

2.1 Syntax and semantics

Parametric trace expressions are built on top of event types, each specifying a set of events. If ϑ is an event type, possibly containing free variables, then $match(e, \vartheta) = \sigma$ means that event e matches event type ϑ with computed substitution σ which must be grounding for the event type ϑ, that is, the domain on which σ is defined coincides with the set of variables in ϑ.

A substitution σ is a finite partial map with domain denoted by $dom(\sigma)$. The substitution with the empty domain is denoted by $\emptyset$. The equality $\sigma = \sigma_1 \cup \sigma_2$ holds iff $dom(\sigma) = dom(\sigma_1) \cup dom(\sigma_2)$, and for all $X \in dom(\sigma)$, $\sigma(X) = \sigma_1(X)$ if $X \in dom(\sigma_1)$, and $\sigma(X) = \sigma_2(X)$ if $X \in dom(\sigma_2)$ (hence, σ_1 and σ_2 must coincide on $dom(\sigma_1) \cap dom(\sigma_2)$). The notation $\sigma_{|X}$ denotes the substitution where X is removed from its domain: $\sigma_{|X} = \sigma'$ iff $dom(\sigma') = dom(\sigma) \setminus \{X\}$ and for all $X \in dom(\sigma')$ $\sigma'(X) = \sigma(X)$. The notation $\sigma\vartheta$ denotes the event type obtained from ϑ by substituting all occurrences of $X \in dom(\sigma)$ in ϑ with $\sigma(X)$.

As trace expressions, parametric trace expressions can be recursive through cyclic terms expressed by finite sets of recursive syntactic equations, as supported by modern Prolog systems.

A parametric trace expression τ is built on top of the operators defined below together with their meaning:

- ϵ (empty trace): the singleton set $\{\epsilon\}$ containing the empty event trace ϵ.
- $\vartheta{:}\tau$ (*prefix*): the set of all traces whose first event e matches the event type ϑ, and the remaining part is a trace of τ.
- $\tau_1 \cdot \tau_2$ (*concatenation*): the set of all traces obtained by concatenating the traces of τ_1 with those of τ_2.
- $\tau_1 \wedge \tau_2$ (*intersection*): the intersection of the traces of τ_1 and τ_2.
- $\tau_1 \vee \tau_2$ (*union*): the union of the traces of τ_1 and τ_2.
- $\tau_1 | \tau_2$ (*shuffle*): the set obtained by shuffling the traces of τ_1 with the traces of τ_2.
- $<X;\tau>$ (*binder*): it binds the free occurrences of X in τ; accordingly, the trace expression $\sigma\tau$ obtained from τ by substituting all free occurrences of $X \in dom(\sigma)$ in τ with $\sigma(X)$, is coinductively defined as follows:

$$\sigma(\vartheta{:}\tau) = (\sigma\vartheta){:}(\sigma\tau)$$
$$\sigma(\tau_1 \, op \, \tau_2) = (\sigma\tau_1) \, op \, (\sigma\tau_2) \text{ for } op \in \{\vee, \wedge, |, \cdot\}$$
$$\sigma(<X;\tau>) = <X; \sigma_{|X}\tau>$$

The transition system for parametric trace expressions is defined in Figure 1. The main transition relation $\tau \xrightarrow{e} \tau'$ is defined in terms of the auxiliary relation $\tau \overset{e}{\rightarrowtail} \tau'; \sigma$, where σ is the substitution generated during the transition step. This is required because it is not possible to predict from which variable occurrence a certain substitution is generated; consider for instance the trace expressions $\vartheta_1(X){:}\tau_1 | \vartheta_2(X){:}\tau_2$, where $\vartheta_1(X)$ and $\vartheta_2(X)$ are two distinct event types containing occurrences of variable X; if event e fires a transition on the lhs operand of the shuffle operator, then the computed substitution σ is s.t. $\sigma = match(e, \vartheta_1)$, and the trace expression is rewritten into $\sigma(\tau_1 | \vartheta_2(X){:}\tau_2)$, otherwise, if the transition is fired on the rhs, then $\sigma = match(e, \vartheta_2)$, and the trace expression is rewritten into $\sigma(\vartheta_1(X){:}\tau_1 | \tau_2)$.

Rule (main) defines the main transition relation in terms

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

$$\text{(main)}\ \frac{\tau \overset{e}{\rightarrowtail} \tau';\emptyset}{\tau \overset{e}{\rightarrow} \tau'} \qquad \text{(prefix)}\ \frac{}{\vartheta{:}\tau \overset{e}{\rightarrowtail} \tau;\sigma}\ \sigma{=}match(e,\vartheta) \qquad \text{(or-l)}\ \frac{\tau_1 \overset{e}{\rightarrowtail} \tau_1';\sigma}{\tau_1 \vee \tau_2 \overset{e}{\rightarrowtail} \tau_1';\sigma} \qquad \text{(or-r)}\ \frac{\tau_2 \overset{e}{\rightarrowtail} \tau_2';\sigma}{\tau_1 \vee \tau_2 \overset{e}{\rightarrowtail} \tau_2';\sigma}$$

$$\text{(and)}\ \frac{\tau_1 \overset{e}{\rightarrowtail} \tau_1';\sigma_1 \quad \tau_2 \overset{e}{\rightarrowtail} \tau_2';\sigma_2}{\tau_1 \wedge \tau_2 \overset{e}{\rightarrowtail} \tau_1' \wedge \tau_2';\sigma}\ \sigma{=}\sigma_1 \cup \sigma_2 \qquad \text{(shuffle-l)}\ \frac{\tau_1 \overset{e}{\rightarrowtail} \tau_1';\sigma}{\tau_1 | \tau_2 \overset{e}{\rightarrowtail} \tau_1' | \tau_2;\sigma} \qquad \text{(shuffle-r)}\ \frac{\tau_2 \overset{e}{\rightarrowtail} \tau_2';\sigma}{\tau_1 | \tau_2 \overset{e}{\rightarrowtail} \tau_1 | \tau_2';\sigma}$$

$$\text{(cat-l)}\ \frac{\tau_1 \overset{e}{\rightarrowtail} \tau_1';\sigma}{\tau_1 \cdot \tau_2 \overset{e}{\rightarrowtail} \tau_1' \cdot \tau_2;\sigma} \qquad \text{(cat-r)}\ \frac{\tau_2 \overset{e}{\rightarrowtail} \tau_2';\sigma}{\tau_1 \cdot \tau_2 \overset{e}{\rightarrowtail} \tau_2';\sigma}\ \epsilon(\tau_1) \qquad \text{(var-t)}\ \frac{\tau \overset{e}{\rightarrowtail} \tau';\sigma}{<X;\tau> \overset{e}{\rightarrowtail} \sigma\tau';\sigma_{|X}}\ X{\in}dom(\sigma)$$

$$\text{(var-f)}\ \frac{\tau \overset{e}{\rightarrowtail} \tau';\sigma}{<X;\tau> \overset{e}{\rightarrowtail} <X;\tau'>;\sigma}\ X{\notin}dom(\sigma) \qquad \text{(ϵ-var)}\ \frac{\epsilon(\tau)}{\epsilon(<X;\tau>)} \qquad \text{(ϵ-empty)}\ \frac{}{\epsilon(\epsilon)} \qquad \text{(ϵ-or-l)}\ \frac{\epsilon(\tau_1)}{\epsilon(\tau_1 \vee \tau_2)}$$

$$\text{(ϵ-or-r)}\ \frac{\epsilon(\tau_2)}{\epsilon(\tau_1 \vee \tau_2)} \qquad \text{(ϵ-others)}\ \frac{\epsilon(\tau_1) \quad \epsilon(\tau_2)}{\epsilon(\tau_1\, op\, \tau_2)}\ op{\in}\{|,\cdot,\wedge\}$$

Figure 1: Transition system for parametric trace expressions

of the auxiliary transition relation which computes the substitution; at top level a correct trace expression cannot contain free variables (that is, undeclared variables), hence the main transition is fired only if the computed substitution is empty.

In rule (prefix) a substitution is generated by applying *match* to the current event e and the event type ϑ.

Rules for union, shuffle, and concatenation are straightforward.

In rule (and) the side condition requires that the substitutions σ_1 and σ_2 computed for the two operands must coincide on the intersection of their domains; the final substitution σ is obtained by merging σ_1 and σ_2.

Rules (var-t) and (var-f) deal with the construct $<X;\tau>$ for variable scoping. The former rule is applied when variable X is contained in the domain of the computed substitution σ for the transition starting from τ; in such a case σ is applied to the trace expression τ' in which τ rewrites to, and the scoping construct is removed; furthermore, the computed substitution is $\sigma_{|X}$. The latter rule is applied when variable X is not contained in the domain of the computed substitution σ for the transition starting from τ; in this case the scoping construct is not removed, and the computed substitution coincides with σ.

The auxiliary predicate $\epsilon(\_)$ checking whether a trace expression is allowed to contain the empty trace.

3. CASE STUDY

We were able to formalize and monitor through a parametric trace expressions a slightly different version of the English auction FIPA specification [13].

The protocol involves an *auctioner* agent, and two or more *bidder* agents. The protocol starts with a preamble where *auctioner* sends a single message to all bidders (in any order) informing that the auction for selling a certain item is going to start. It is assumed that all contacted bidders will participate to the auction, until *auctioner* will notify bidders about the closing of the auction.

The preamble is followed by an initial proposal round, where *auctioner* sends a single message to all bidders with a proposed item price, which is equal for all bidders; then *auctioner* waits to receive a single reply from all bidders before deciding to either closing the auction, or moving to the next proposal round. Every bidder can either accept or reject the proposal; a bidder b will keep attending the auction, even when b decides to reject a proposal at a certain round.

Agent *auctioner* moves to the next proposal round if at least two bidders have accepted the previously proposed price; in such a case, the new proposed price will be at least greater or equal than $p + \Delta p$, where p is the price proposed at the previous round, and Δp is an a priori fixed positive constant.

If a bidder b has accepted the last proposed price, then *auctioner* closes the auction by sending a single message to all bidders; bidder b is notified of the purchase of the item, while all other bidders are informed that the item has been sold.

Finally, if no bidder accepts the current proposal, then *auctioner* closes the auction by sending a single message to all bidders to notify them that the item is unsold.

3.1 Monitoring of the protocol

We have defined a parametric trace expression[1] which allowed us to successful monitor a system implementing the auction protocol described above with an unspecified number of participants that can be fixed only at runtime.

This case study shows that parametric trace expressions are a powerful but also efficient formalism for expressing parametric specifications for RV; experiments with our implemented tool for RV through parametric trace expressions show that runtime verification is linear in $TL \cdot BN$, where TL is the total length of the analyzed trace, and BN is the number of bidders; the tool is able to support runtime verification of a Jade multiagent system with 500 bidders. Beyond such a limit, the tool starts reporting protocol violations simply because the Jade platform is no longer able to guarantee that messages sent from agent a to agent b are received by b in the expected order.

To better assess our tool, we are planning to further experiment with parametric trace expressions to constitute a library of specifications for the most commonly used interaction protocols, and to individuate recurrent patterns that can be usefully exploited to ease the specification of complex protocols.

[1]The parametric trace expression together with the JADE MAS and monitor can be downloaded from `http://www.ParametricTraceExpr.altervista.org`.

REFERENCES

[1] F. Aielli, D. Ancona, P. Caianiello, S. Costantini, G. De Gasperis, A. Di Marco, A. Ferrando, and V. Mascardi. FRIENDLY & KIND with your health: Human-friendly knowledge-intensive dynamic systems for the e-health domain. In *Highlights of Practical Applications of Scalable Multi-Agent Systems. The PAAMS Collection - International Workshops of PAAMS 2016, Sevilla, Spain, June 1-3, 2016. Proceedings*, pages 15–26, 2016.

[2] D. Ancona, V. Bono, M. Bravetti, J. Campos, G. Castagna, P. M. Deniélou, S. J. Gay, N. Gesbert, E. Giachino, R. Hu, E. B. Johnsen, F. Martins, V. Mascardi, F. Montesi, R. Neykova, N. Ng, L. Padovani, V. Vasconcelos, and N. Yoshida. Behavioral types in programming languages. *Foundations and Trends in Programming Languages*, 3(2-3):95–230, 2016.

[3] D. Ancona, D. Briola, A. El Fallah Seghrouchni, V. Mascardi, and P. Taillibert. Efficient verification of mass with projections. In F. Dalpiaz, J. Dix, and M. B. van Riemsdijk, editors, *Engineering Multi-Agent Systems: Second International Workshop, EMAS 2014, Paris, France, May 5-6, 2014, Revised Selected Papers*, pages 246–270, Cham, 2014. Springer International Publishing.

[4] D. Ancona, D. Briola, A. Ferrando, and V. Mascardi. Runtime verification of fail-uncontrolled and ambient intelligence systems: A uniform approach. *Intelligenza Artificiale*, 9(2):131–148, 2015.

[5] D. Ancona, D. Briola, A. Ferrando, and V. Mascardi. Mas-drive: a practical approach to decentralized runtime verification of agent interaction protocols. In *Proceedings of the 17th Workshop "From Objects to Agents" co-located with 18th European Agent Systems Summer School (EASSS 2016), Catania, Italy, July 29-30, 2016.*, pages 35–43, 2016.

[6] D. Ancona, S. Drossopoulou, and V. Mascardi. Automatic generation of self-monitoring MASs from multiparty global session types in Jason. In *DALT 2012*, volume 7784 of *LNAI*, pages 76–95. Springer, 2012.

[7] D. Ancona, A. Ferrando, and V. Mascardi. Comparing trace expressions and linear temporal logic for runtime verification. In *Theory and Practice of Formal Methods - Essays Dedicated to Frank de Boer on the Occasion of His 60th Birthday*, pages 47–64. Springer Verlag, 2016.

[8] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.

[9] D. Briola, V. Mascardi, and D. Ancona. Distributed runtime verification of JADE and jason multiagent systems with prolog. In L. Giordano, V. Gliozzi, and G. L. Pozzato, editors, *Proceedings of the 29th Italian Conference on Computational Logic, Torino, Italy, June 16-18, 2014.*, volume 1195 of *CEUR Workshop Proceedings*, pages 319–323. CEUR-WS.org, 2014.

[10] D. Briola, V. Mascardi, and D. Ancona. Distributed runtime verification of JADE multiagent systems. In *IDC*, Studies in Computational Intelligence. Springer, 2014.

[11] A. D., M. Barbieri, and V. Mascardi. Constrained global types for dynamic checking of protocol conformance in multi-agent systems. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13*, pages 1377–1379, 2013.

[12] A. Ferrando, D. Ancona, and V. Mascardi. Decentralizing mas monitoring with decamon. In S. Das, E. Durfee, K. Larson, and M. Winikoff, editors, *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2017. Proceedings*. IFAAMAS, 2017.

[13] Foundation for Intelligent Physical Agents. FIPA english auction interaction protocol specification. `http://www.fipa.org/specs/fipa00031/XC00031F.pdf`, 2001.

[14] Q. Luo, Y. Zhang, C. Lee, D. Jin, P. O. Meredith, T. Serbanuta, and G. Rosu. Rv-monitor: Efficient parametric runtime verification with simultaneous properties. In *Runtime Verification, RV 2014*, pages 285–300, 2014.