Runtime Verification Based on Register Automata

Radu Grigore[•] Dino Distefano[•]
Rasmus Lerchedahl Petersen[•→◦]
Nikos Tzevelekos[•]

[•]Queen Mary, University of London

[◦]Microsoft Research, Cambridge

TACAS 2013

Register Automata

regular languages [K1951, RS1959]

- specified with finite automata, or regexp
- have finite alphabets

quasi-regular languages [KF1990]

- specified with register automata
- may have infinite alphabets

Runtime Verification for Java

- Java executions mention values

$$\ldots,\ \text{call}\ \mathit{hasNext}(231), \text{return}\ \mathit{hasNext}(1), \ldots$$

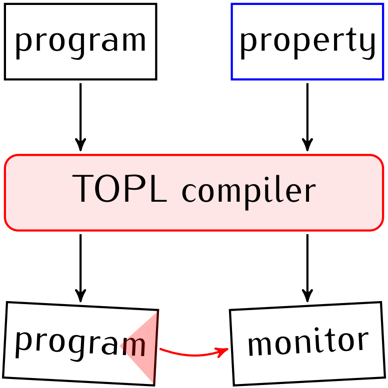
- slicing by values [RC2012]

$$\begin{array}{l} \mathit{HasNext}(i) \\ \quad \text{event}\ hN = \text{call}(*\mathit{hasNext}())\ \&\&\ \text{target}(i) \\ \quad \text{event}\ n = \text{call}(*\mathit{next}())\ \&\&\ \text{target}(i) \\ \quad \text{fsm} : \mathit{start}[hN \to \mathit{safe} \quad n \to \mathit{error}] \\ \qquad\qquad \mathit{safe}[hN \to \mathit{safe} \quad n \to \mathit{start}] \end{array}$$

DEMO

of `http://rgrig.github.com/topl`

Tool Architecture



TOPL Properties and Automata

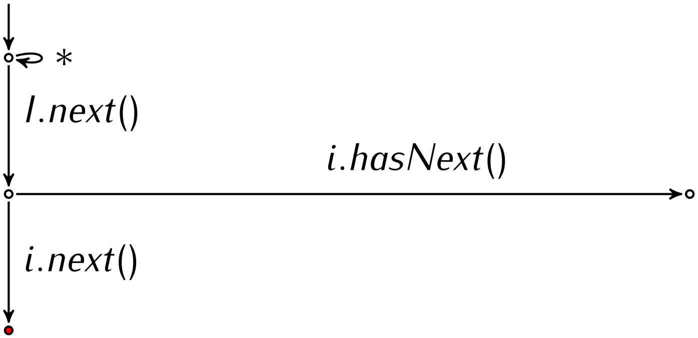
user-friendly

Given a path $start \xrightarrow{l_1} \circ \cdots \circ \xrightarrow{l_k} error$, the sequence $l_1; \ldots; l_k$ of labels looks very much like a small bad Java program.

expressive

- equivalent to register automata
- inherits closure and decidability results

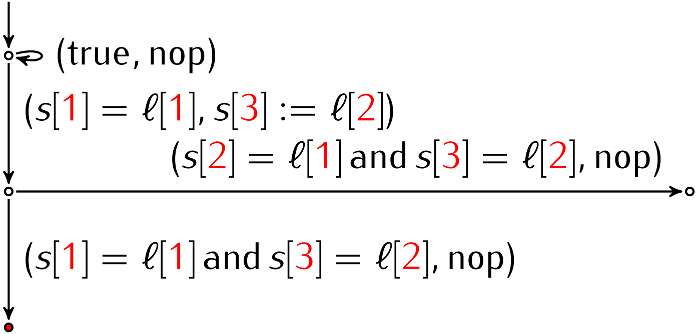
TOPL Automaton Example



$$s_0 = \{\, 1 \mapsto next, 2 \mapsto hasNext, 3 \mapsto \bot \,\} \qquad 3 = i$$

$$\tau = (hasNext, 231), (next, 231), (next, 134), \ldots$$

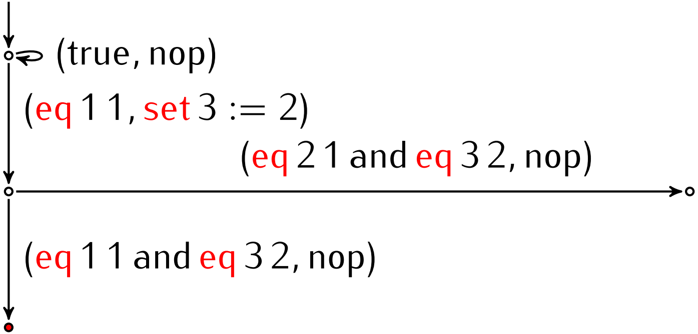
TOPL Automaton Example



$$s_0 = \{\, 1 \mapsto \mathit{next}, 2 \mapsto \mathit{hasNext}, 3 \mapsto \bot \,\} \qquad 3 = i$$

$$\tau = (\mathit{hasNext}, 231), (\mathit{next}, 231), (\mathit{next}, 134), \ldots$$

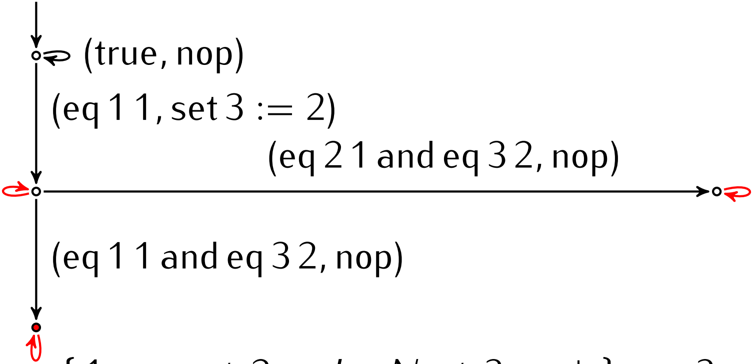
TOPL Automaton Example



$$s_0 = \{\, 1 \mapsto \mathit{next}, 2 \mapsto \mathit{hasNext}, 3 \mapsto \bot \,\} \qquad 3 = i$$

$$\tau = (\mathit{hasNext}, 231), (\mathit{next}, 231), (\mathit{next}, 134), \ldots$$

TOPL Automaton Example



$$s_0 = \{\, 1 \mapsto \mathit{next}, 2 \mapsto \mathit{hasNext}, 3 \mapsto \bot \,\} \qquad 3 = i$$

$$\tau = (\mathit{hasNext}, 231), (\mathit{next}, 231), (\mathit{next}, 134), \ldots$$

Notations

$$
\begin{array}{rll}
u, v \in V = \{0, 1, 2, \ldots\} & \text{values} \\
(s_1, \ldots, s_m) = s \in S = V^m & \text{stores} \\
(\ell_1, \ldots, \ell_n) = \ell \in \Sigma = V^n & \text{letters} \\
g \in G & \text{guards} \\
a \in A \subseteq \Sigma \to S \to S & \text{actions} \\
\lambda \in \Lambda = G \times A & \text{labels} \\
q \in Q & \text{states} \\
(q \xrightarrow{\lambda} q') \in \delta \subseteq Q \times \Lambda \times Q & \text{transitions}
\end{array}
$$

Guards and Actions

guards

$$(s, \ell) \models g \qquad \text{satisfaction relation}$$

actions

$$\big((\mathsf{set}\ i := j)\ \ell\ s\big)_k = (\text{if } k = i \text{ then } \ell_j \text{ else } s_k)$$

$$(a_1 ; a_2)(\ell) = a_1(\ell) \circ a_2(\ell)$$

Automata Definition and Semantics

automaton

$$(Q, q_0, s_0, F, \delta) \qquad \delta \subseteq Q \times \Lambda \times Q$$

configuration graph

$$\begin{array}{llll} & q \xrightarrow{(g,a)} q' & \text{gives} & (q, s) \xrightarrow{\ell} (q', s') \\ \text{when} & (s, \ell) \models g & \text{and} & s' = a\ \ell\ s \end{array}$$

Register Automata Definition

- alphabet: $\Sigma = V^1$
- guards:

$$(s, v) \models \text{fresh} \iff s_i \neq v \text{ for all } i$$
$$(s, v) \models \text{eq } i \iff s_i = v$$

- labels:

$(\text{fresh}, \text{set } i := 1)$	record a (locally) fresh value
$(\text{eq } i, \text{nop})$	test for equality with a register

TOPL Automata Definition

- guards:

$$
\begin{aligned}
(s,\ell) &\models \mathrm{eq}\ i\ j &&\iff && s_i = \ell_j \\
(s,\ell) &\models \mathrm{neq}\ i\ j &&\iff && s_i \neq \ell_j \\
\sigma &\models \mathrm{true} &&\iff && \text{always} \\
\sigma &\models g_1\ \mathrm{and}\ g_2 &&\iff && \sigma \models g_1 \text{ and } \sigma \models g_2
\end{aligned}
$$

where σ is a pair (s,ℓ)

Results

Proposition: TOPL to RA, and back

A TOPL automaton of size N can be transformed into an equivalent RA of size $N^{O(N^2)}$. A RA of size N with $< k$ conjuncts in each guard can be transformed into an equivalent TOPL automaton of size $O(Nk^N)$.

Theorem

TOPL automata inherit all expressivity, closure and decidability results from register automata.

Done

- we noticed: automata over infinite alphabets are useful for runtime verification
- we designed: TOPL automata — taylored for Java, but equivalent to register automata
- we implemented: a tool for runtime verification

ToDo

- static analysis
- implementation of histories (sets) [TG2013]

References

K1951 Kleene, *Representation of Events in Nerve Nets and Finite Automata*

RS1959 Rabin, Scott, *Finite Automata and Their Decision Problems*

KF1990 Kaminski, Francez, *Finite Memory Automata*

A+2005 Allan et al., *Adding Trace Matching with Free Variables to AspectJ*

RC2012 Roșu, Chen, *Semantics and Algorithms for Parametric Monitoring*

B+2012 Barringer et al., *Quantified Event Automata: Towards Expressive and Efficient Runtime Monitors*

TG2013 Tzevelekos, Grigore, *History-Register Automata*