SATURATING AUTOMATA FOR GAME SEMANTICS

Alex Dixon
University of Warwick

Andrzej Murawski
University of Oxford

FICA

$$\frac{}{\Gamma \vdash \mathbf{skip} : \mathbf{com}} \qquad \frac{}{\Gamma \vdash \mathbf{div}_\theta : \theta} \qquad \frac{0 \leqslant i \leqslant max}{\Gamma \vdash i : \mathbf{exp}} \qquad \frac{\Gamma \vdash M : \mathbf{exp}}{\Gamma \vdash \mathbf{op}(M) : \mathbf{exp}}$$

$$\frac{\Gamma \vdash M : \mathbf{com} \qquad \Gamma \vdash N : \beta}{\Gamma \vdash M ; N : \beta} \qquad \frac{\Gamma \vdash M : \mathbf{com} \qquad \Gamma \vdash N : \mathbf{com}}{\Gamma \vdash M || N : \mathbf{com}}$$

$$\frac{\Gamma \vdash M : \mathbf{exp} \qquad \Gamma \vdash N_1, N_2 : \beta}{\Gamma \vdash \mathbf{if}\, M \,\mathbf{then}\, N_1 \,\mathbf{else}\, N_2 : \beta} \qquad \frac{\Gamma \vdash M : \mathbf{exp} \qquad \Gamma \vdash N : \mathbf{com}}{\Gamma \vdash \mathbf{while}\, M \,\mathbf{do}\, N : \mathbf{com}}$$

$$\frac{}{\Gamma, x : \theta \vdash x : \theta} \qquad \frac{\Gamma, x : \theta \vdash M : \theta'}{\Gamma \vdash \lambda x . M : \theta \to \theta'} \qquad \frac{\Gamma \vdash M : \theta \to \theta' \qquad \Gamma \vdash N : \theta}{\Gamma \vdash M N : \theta'}$$

$$\frac{\Gamma \vdash M : \mathbf{var} \qquad \Gamma \vdash N : \mathbf{exp}}{\Gamma \vdash M := N : \mathbf{com}} \qquad \frac{\Gamma \vdash M : \mathbf{var}}{\Gamma \vdash !M : \mathbf{exp}} \qquad \frac{\Gamma, x : \mathbf{var} \vdash M : \mathbf{com}, \mathbf{exp}}{\Gamma \vdash \mathbf{newvar}\, x \,\mathbf{in}\, M : \mathbf{com}, \mathbf{exp}}$$

$$\frac{\Gamma \vdash M : \mathbf{sem}}{\Gamma \vdash \mathbf{release}(M) : \mathbf{com}} \qquad \frac{\Gamma \vdash M : \mathbf{sem}}{\Gamma \vdash \mathbf{grab}(M) : \mathbf{com}} \qquad \frac{\Gamma, s : \mathbf{sem} \vdash M : \mathbf{com}, \mathbf{exp}}{\Gamma \vdash \mathbf{newsem}\, s \,\mathbf{in}\, M : \mathbf{com}, \mathbf{exp}}$$

EXAMPLE

$$f : \mathbf{com} \to \mathbf{com}, c : \mathbf{com} \vdash$$

$$\mathbf{newvar}\, x\, \mathbf{in}\, (f\ (x := 1)\ ||\ \mathbf{if}\ !x\, \mathbf{then}\ c\ \mathbf{else}\ \mathbf{div}_{\mathbf{com}}); !x : \mathbf{exp}$$

GAME MODEL

Available online at www.sciencedirect.com

ScienceDirect

Annals of Pure and Applied Logic 151 (2008) 89–114

ELSEVIER

ANNALS OF PURE AND APPLIED LOGIC

www.elsevier.com/locate/apal

Angelic semantics of fine-grained concurrency[☆]

Dan R. Ghica[a], Andrzej S. Murawski[b,*]

[a] *School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK*

[b] *Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK*

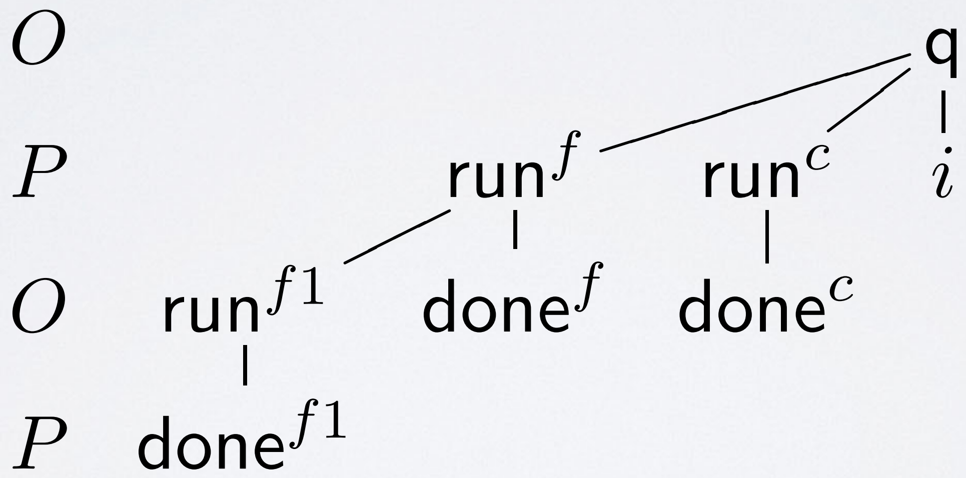
Available online 26 November 2007

Abstract

We introduce a game model for an Algol-like programming language with primitives for parallel composition and synchronization on semaphores. The semantics is based on a simplified version of Hyland–Ong-style games and it emphasizes the intuitive connection between the concurrent nature of games and that of computation. The model is fully abstract for *may*-equivalence.

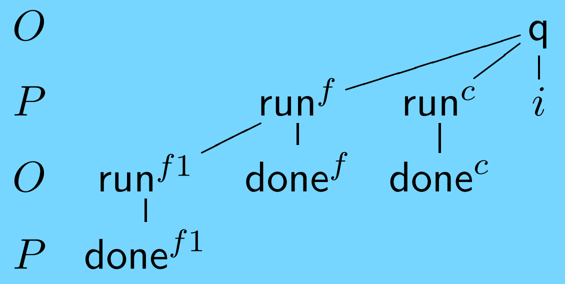
ARENAS

$$A = [\![\mathbf{com} \to \mathbf{com}]\!] \times [\![\mathbf{com}]\!] \Rightarrow [\![\mathbf{exp}]\!]$$



PLAYS

$$A = [\![\mathbf{com} \to \mathbf{com}]\!] \times [\![\mathbf{com}]\!] \Rightarrow [\![\mathbf{exp}]\!]$$



$$s_1 = \begin{array}{ccc} \mathsf{q} & \mathsf{run}^f & \mathsf{done}^f \\ O & P & O \end{array}$$

$$s_2 = \begin{array}{cccccccc} \mathsf{q} & \mathsf{run}^f & \mathsf{run}^{f1} & \mathsf{done}^{f1} & \mathsf{run}^c & \mathsf{done}^c & \mathsf{done}^f & 1 \\ O & P & O & P & P & O & O & P \end{array}$$

FORK AND WAIT (VIA NON-EXAMPLES)

FORK : In any prefix $s' = \cdots q \cdots m$ of s, the question q must be pending when m is played.

q	run^f	done^f	run^{f1}
O	P	O	O

WAIT : In any prefix $s' = \cdots q \cdots a$ of s, all questions justified by q must be answered.

q	run^f	run^{f1}	done^f
O	P	O	P

SATURATED STRATEGIES

$$\frac{sm_1m_2s' \in \sigma \quad sm_2m_1s' \text{ is a play} \quad \neg(m_1 \text{ is an O-move and } m_2 \text{ is an P-move})}{sm_2m_1s' \in \sigma}$$

Saturation is about capturing dependencies of *P-moves* on *O-moves*.

SATURATION EXAMPLE 1

run	run^1	run^2	done^1	done^2	done
O	P	P	O	O	P

run	run^2	run^1	done^1	done^2	done
O	P	P	O	O	P

run	run^2	done^2	run^1	done^1	done
O	P	O	P	O	P

SATURATION EXAMPLE 2

$$\begin{array}{cccccccc}
\mathsf{q} & \mathsf{run}^{f} & \mathsf{run}^{f1} & \mathsf{done}^{f1} & \mathsf{run}^{c} & \mathsf{done}^{c} & \mathsf{done}^{f} & 1 \\
O & P & O & P & P & O & O & P
\end{array}$$

$$\begin{array}{cccccccc}
\mathsf{q} & \mathsf{run}^{f} & \mathsf{run}^{f1} & \mathsf{run}^{c} & \mathsf{done}^{c} & \mathsf{done}^{f1} & \mathsf{done}^{f} & 1 \\
O & P & O & P & O & P & O & P
\end{array}$$

SATURATING AUTOMATA

- Automata over infinite (forest-shaped) alphabets

Accept words over the alphabet $\Sigma \times \mathcal{D}$, where

- Σ is a finite alphabet,
- $\mathcal{D}$ is a countably infinite forest.

$(\mathsf{q}, d_0)\ (\mathsf{run}^f, d_1)\ (\mathsf{run}^{f1}, d_2)\ (\mathsf{done}^{f1}, d_2)\ (\mathsf{run}^c, d_1')\ (\mathsf{done}^c, d_1')\ (\mathsf{done}^f, d_1)\ (1, d_0)$

WORDS AS PLAYS

$$(\mathsf{q}, d_0)\,(\mathsf{run}^f, d_1)\,(\mathsf{run}^{f1}, d_2)\,(\mathsf{done}^{f1}, d_2)\,(\mathsf{run}^c, d'_1)\,(\mathsf{done}^c, d'_1)\,(\mathsf{done}^f, d_1)\,(1, d_0)$$

$$\begin{tikzcd}[column sep=small]
 & d_0 \arrow[dl, dash] \arrow[dr, dash] & \\
d_1 \arrow[d, dash] & & d'_1 \\
d_2 & &
\end{tikzcd}$$

$$\begin{array}{cccccccc}
\mathsf{q} & \mathsf{run}^f & \mathsf{run}^{f1} & \mathsf{done}^{f1} & \mathsf{run}^c & \mathsf{done}^c & \mathsf{done}^f & 1 \\
O & P & O & P & P & O & O & P
\end{array}$$

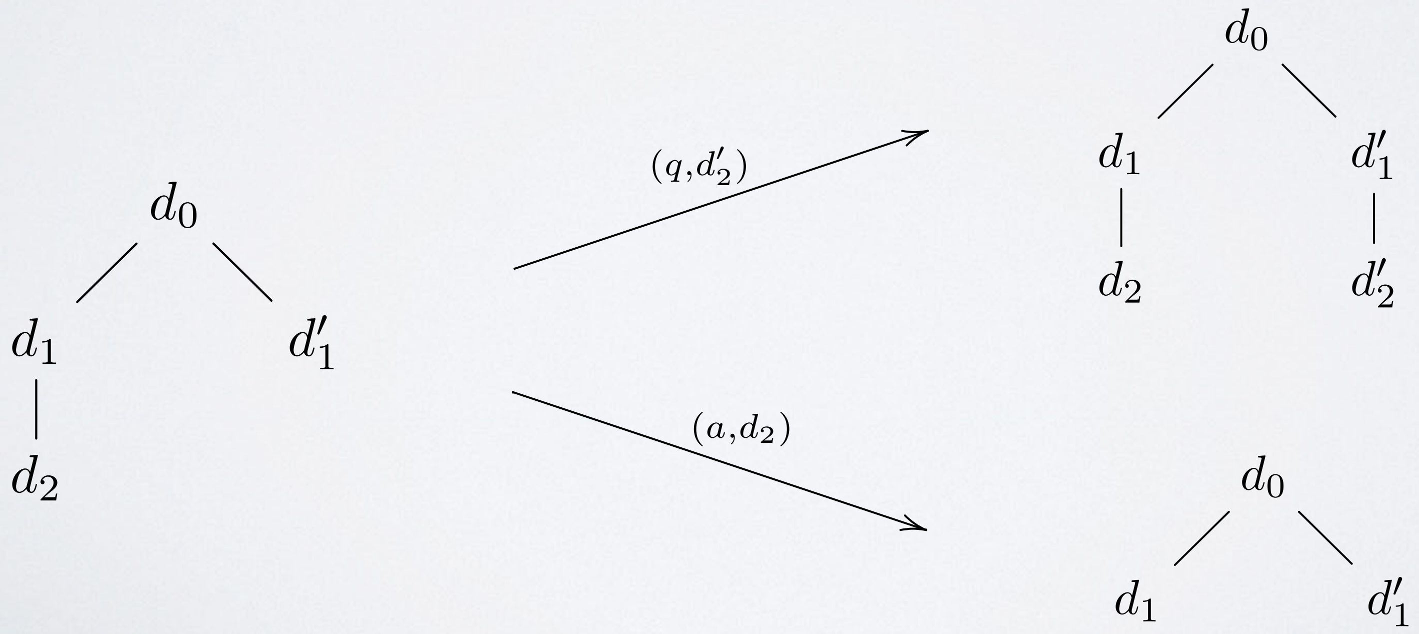
SATURATED LANGUAGES

$L \subseteq (\Sigma \times \mathcal{D})^*$ is *saturated* iff, for any $w \in L$ and independent d_1, d_2, if $w = w_1(t_1, d_1)(t_2, d_2)w_2 \in L$ then $w_1(t_2, d_2)(t_1, d_1)w_2 \in L$ whenever $t_1 \in \Sigma_P$ or $t_2 \in \Sigma_O$.

Languages accepted by saturating automata will be saturated.

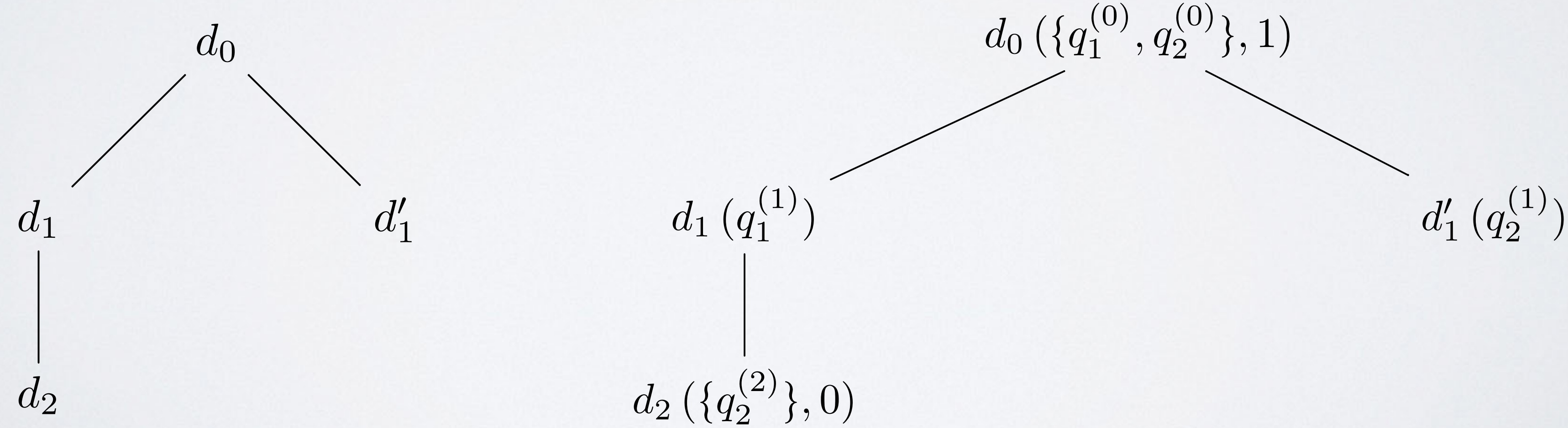
SATURATING AUTOMATA (SATA)

- Σ is partitioned into O/P-questions and O/P-answers.
- Configurations are finite subtrees of $\mathcal{D}$ annotated with extra information.
- The tree evolves: questions add leaves (FORK), answers remove leaves (WAIT).



SATURATING AUTOMATA (SATA)

- Each even-level node is annotated with a multiset of control states, and zero or more memory cells. This information will evolve at run time.
- Odd levels are annotated with single control states, which do not change.



$$\dagger \xrightarrow{q_O} \{l^{(0)}, r^{(0)}\}$$

$$d_0(\{l^{(0)}, r^{(0)}\}, 0)$$

$$l^{(0)} \xrightarrow{q_P} l^{(1)}$$

$$\begin{tikzcd}
d_0(\{r^{(0)}\}, 0) \arrow[d, dash] \\
d_1(l^{(1)})
\end{tikzcd}$$

$$l^{(1)} \xrightarrow{q_O} \{l^{(2)}\}$$

$$\begin{tikzcd}
d_0(\{r^{(0)}\}, 0) \arrow[d, dash] \\
d_1(l^{(1)}) \arrow[d, dash] \\
d_2(\{l^{(2)}\}, 0)
\end{tikzcd}$$

$$
\begin{tikzcd}
d_0(\{r^{(0)}\}, 0) \arrow[d, dash] \\
d_1(l^{(1)}) \arrow[d, dash] \\
d_2(\{l^{(2)}\}, 0)
\end{tikzcd}
$$

$$(l^{(2)}, 1, 0, 0) \xrightarrow{\epsilon} (l_+^{(2)}, 1)$$

$$
\begin{tikzcd}
d_0(\{r^{(0)}\}, 1) \arrow[d, dash] \\
d_1(l^{(1)}) \arrow[d, dash] \\
d_2(\{l_+^{(2)}\}, 0)
\end{tikzcd}
$$

$$(r^{(0)}, 1, 0, 1) \xrightarrow{\epsilon} (r_+^{(0)}, 1)$$

$$
\begin{tikzcd}
d_0(\{r_+^{(0)}\}, 1) \arrow[d, dash] \\
d_1(l^{(1)}) \arrow[d, dash] \\
d_2(\{l_+^{(2)}\}, 0)
\end{tikzcd}
$$

$$\begin{tikzcd}
d_0(\{r_+^{(0)}\},1) \arrow[d, dash] \\
d_1(l^{(1)}) \arrow[d, dash] \\
d_2(\{l_+^{(2)}\},0)
\end{tikzcd}$$

$r_+^{(0)} \xrightarrow{q_P} r^{(1)}$

$$\begin{tikzcd}
& d_0(\varnothing,1) \arrow[dl, dash] \arrow[dr, dash] & \\
d_1(l^{(1)}) \arrow[d, dash] & & d_1'(r^{(1)}) \\
d_2(\{l_+^{(2)}\},0) & &
\end{tikzcd}$$

$\{l_+^{(2)}\} \xrightarrow{a_P} \dagger$

$$\begin{tikzcd}
& d_0(\varnothing,1) \arrow[dl, dash] \arrow[dr, dash] & \\
d_1(l^{(1)}) & & d_1'(r^{(1)})
\end{tikzcd}$$

$$\begin{tikzcd} & d_0(\varnothing, 1) \arrow[dl, no head] \arrow[dr, no head] & \\ d_1(l^{(1)}) & & d'_1(r^{(1)}) \end{tikzcd}$$

$$l^{(1)} \xrightarrow{a_O} l_+^{(0)}$$

$$\begin{tikzcd} d_0(\{l_+^{(0)}\}, 1) \arrow[dr, no head] & \\ & d'_1(r^{(1)}) \end{tikzcd}$$

$$r^{(1)} \xrightarrow{a_O} r_{++}^{(0)}$$

$$d_0(\{l_+^{(0)}, r_{++}^{(0)}\}, 1)$$

$$\{l_+^{(0)}, r_{++}^{(0)}\} \xrightarrow{a_P} \dagger$$

SUMMARY

- SATA grew out of earlier attempts to model the game semantics of FICA: joint work with Ranko Lazić and Igor Walukiewicz [FoSSaCS/LICS 2021]. The decidability results showed therein carry over to SATA.
- The models mentioned above do not satisfy saturation, because they support more permissive communication between levels.
  – FoSSaCS 2021 allowed for unrestricted access to the whole branch.
  – LICS 2021 allowed for state-based communication between children and parents, i.e. indirect communication between children.
- SATA have been designed to be more restrictive in that regard: communication through control states is minimal and restricted to initialization and finalization. All remaining communications are memory-based.
- SATA provide a more intrinsic model of higher-order concurrency, potentially amenable to methods based on partial-order reduction.
- FICA programs in normal form can be translated to SATA in polynomial time, avoiding exponential blow-ups of other translations.