Semantics Properties

A4M36TPJ, 2013/2014

Global Properties of a Programming Language

- **universality**: the language can express all computable programs;
- **determinism**: the set of possible outcomes from executing a program on any particular inputs is a singleton;
- **strong normalization**: all programs are guaranteed to terminate on all inputs (i.e., it is not possible to express an infinite loop);
- **static checkability**: a class of program errors can be found by static analysis without resorting to execution;
- **referential transparency**: different occurrences of an expression within the same context always have the same meaning.

SOS Formal Definition

$$S =< CF, \Rightarrow, FC, IF, OF >$$

- CF is a set of configurations,
- $\Rightarrow$ is a binary relation over configurations ($\Rightarrow \subseteq CF \times CF$),
- FC is a set of final configurations ($FC \subseteq CF$),
- $IF : Program \times Input \to CF$ is an input function and
- $OF : FC \to Output$ is an output function.

Syntax

$$
\begin{aligned}
Expr ::= \; & Num \mid \\
& Bool \mid \\
& \triangle Expr \mid \\
& Expr \odot Expr \mid \\
& \texttt{if } Expr \texttt{ then } Expr \texttt{ else } Expr
\end{aligned}
$$

where Num is a predefined set of integer numbers (a.k.a. Z) and $Bool$ is a predefined set of boolean values.

Convention: $e, e', \cdots \in Expr$, $n, n' \in Num$, $b, b' \in Bool$, $v \in Num \cup Bool$ and $t \in \{Number, Boolean\}$.

Type System

$$\frac{}{\vdash n : Number}$$

$$\frac{}{\vdash b : Boolean}$$

$$\frac{\vdash e : Number}{\vdash \triangle e : Number}$$

$$\frac{\vdash e : Number \quad \vdash e' : Number}{\vdash e \odot e' : Number}$$

$$\frac{\vdash e : Boolean \quad \vdash e' : t \quad \vdash e'' : t}{\vdash \texttt{if}\ e\ \texttt{then}\ e'\ \texttt{else}\ e'' : t}$$

BOS

$$\frac{}{n \mapsto n}$$

$$\frac{}{b \mapsto b}$$

$$\frac{e \mapsto n}{\triangle e \mapsto -n}$$

$$\frac{e \mapsto n \quad e' \mapsto n'}{e \odot e' \mapsto n + n'}$$

$$\frac{e \mapsto true \quad e' \mapsto v}{\texttt{if}\ e\ \texttt{then}\ e'\ \texttt{else}\ e'' \mapsto v}$$

$$\frac{e \mapsto false \quad e'' \mapsto v}{\texttt{if}\ e\ \texttt{then}\ e'\ \texttt{else}\ e'' \mapsto v}$$

SOS

$$\frac{}{\triangle n \rightsquigarrow -n}$$

$$\frac{e \rightsquigarrow e'}{\triangle e \rightsquigarrow \triangle e'}$$

$$\frac{}{n \odot n' \rightsquigarrow n + n'}$$

$$\frac{e \rightsquigarrow e'}{e \odot e'' \rightsquigarrow e' \odot e''}$$

$$\frac{e' \rightsquigarrow e''}{e \odot e' \rightsquigarrow e \odot e''}$$

SOS

$$\frac{}{\texttt{if}\ true\ \texttt{then}\ e'\ \texttt{else}\ e'' \leadsto e'}$$

$$\frac{}{\texttt{if}\ false\ \texttt{then}\ e'\ \texttt{else}\ e'' \leadsto e''}$$

$$\frac{e \leadsto e'''}{\texttt{if}\ e\ \texttt{then}\ e'\ \texttt{else}\ e'' \leadsto \texttt{if}\ e'''\ \texttt{then}\ e'\ \texttt{else}\ e''}$$

$$\frac{e' \leadsto e'''}{\texttt{if}\ e\ \texttt{then}\ e'\ \texttt{else}\ e'' \leadsto \texttt{if}\ e\ \texttt{then}\ e'''\ \texttt{else}\ e''}$$

$$\frac{e'' \leadsto e'''}{\texttt{if}\ e\ \texttt{then}\ e'\ \texttt{else}\ e'' \leadsto \texttt{if}\ e\ \texttt{then}\ e'\ \texttt{else}\ e'''}$$

Normal Form

- Stated formally, if $(A, \rightarrow)$ is an abstract rewriting system, some $x \in A$ is in normal form if no $y \in A$ exists such that $x \rightarrow y$.
- An irreducible configuration is also called a normal form.
- Each normal form is either a final configuration or a stuck state.

Example

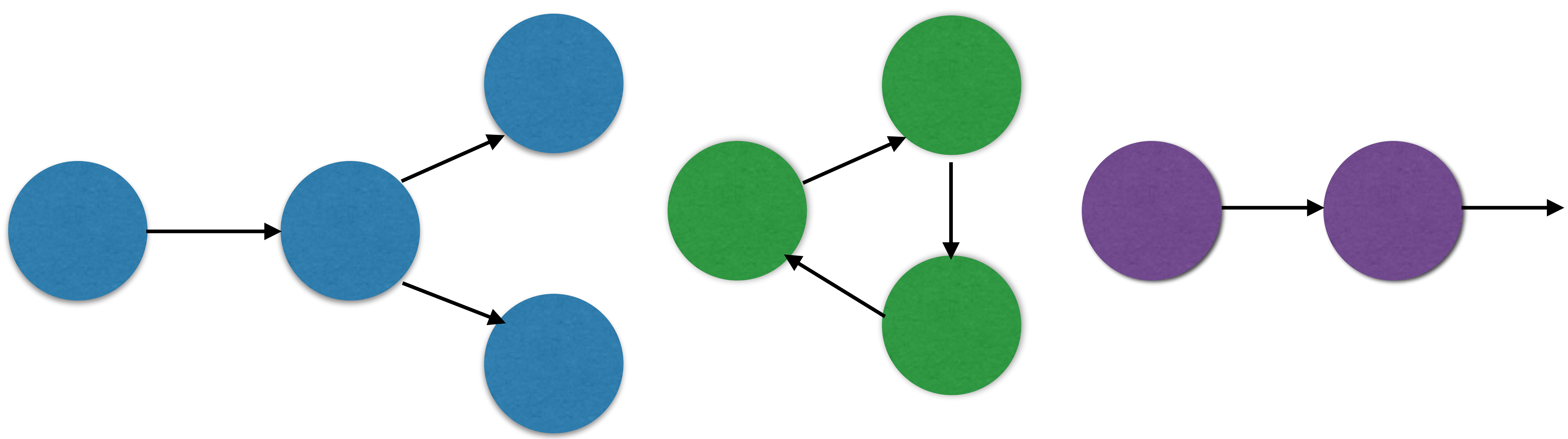
For example, using the term rewriting system with a single rule **f**(x,y)→y, the term f(f(4,2),f(3,1)) can be rewritten as follows:

f(f(4,2),f(3,1)) → **f**(3,1) → 1

Since no rule applies to the last term, **1**, it cannot be rewritten any further, and hence is a normal form of the term f(f(4,2),f(3,1)) with respect to this term rewriting system.

Stuck States

- Stuck States have no output according to output function.
- Stuck states include Cycling, Infinite Semantics, more than one outcome.



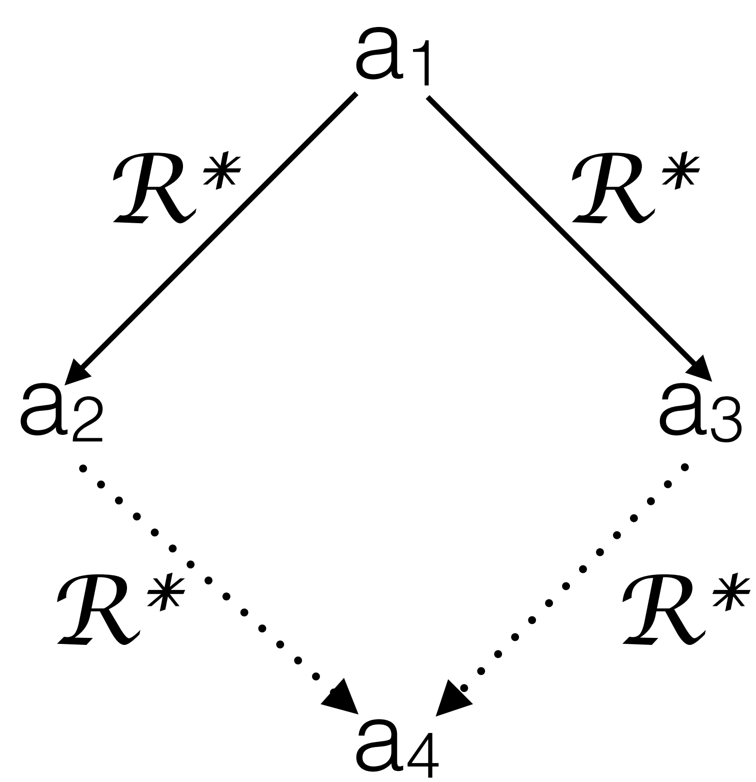
Example

One rule system:

$$g(x,y) \rightarrow g(y,x)$$

Confluence

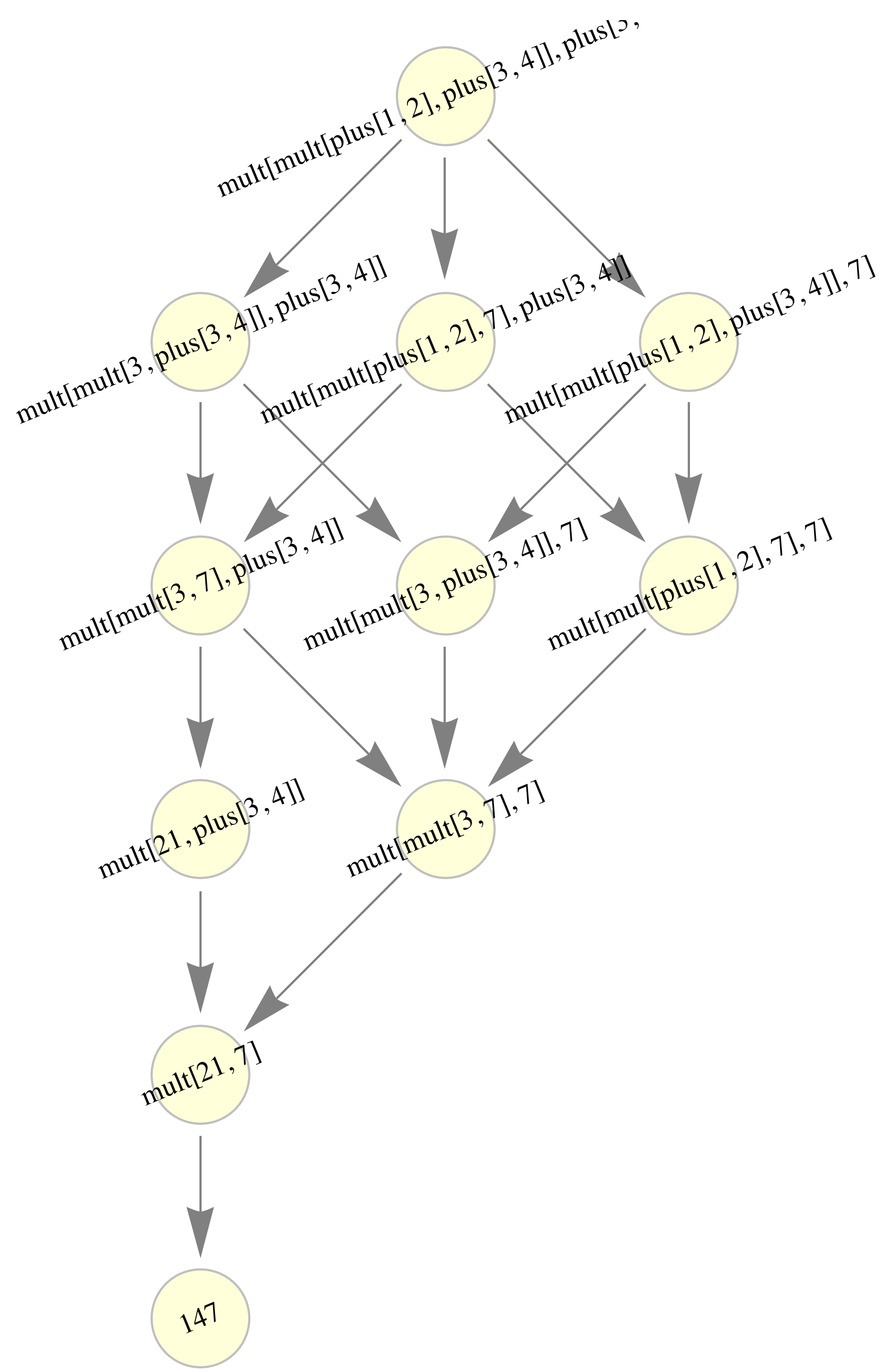
A relation $R \subseteq A \times A$ is confluent iff
$\forall a_1, a_2, a_3 \in A \colon a_1 R^* a_2 \wedge a_1 R^* a_3 \Rightarrow \exists a_4 \in A \colon a_2 R^* a_4 \wedge a_3 R^* a_4.$



Confluence

- A confluent transition relation guarantees a unique final configuration. Why?
- Indeed, it guarantees a unique irreducible configuration (normal form). Why?
- Confluence does not guarantee a single outcome? Why?

Confluence



mult[mult[plus[1,2],plus[3,4]],plus[3,
mult[mult[3,plus[3,4]],plus[3,4]]
mult[mult[plus[1,2],7],plus[3,4]]
mult[mult[plus[1,2],plus[3,4]],7]
mult[mult[3,7],plus[3,4]]
mult[mult[3,plus[3,4]],7]
mult[mult[plus[1,2],7],7]
mult[21,plus[3,4]]
mult[mult[3,7],7]
mult[21,7]
147

Strongly Normalising Relation

- A **strongly normalising** (or **terminating**) transition relation produces normal form via **every path**. A **weakly normalising** transition relation produces normal form via **at least one path**.

- **Well-typed** programs **don't get stuck**.

Example

One rule system:

$$g(x,y) \to x$$

Two rule system:

$$g(x,y) \to x$$

$$g(x,x) \to g(3,x)$$

Semantics Equivalence

$$\forall e \in Expr : e \mapsto n \Leftrightarrow e \leadsto^* n.$$

Type Preservation

$$\forall e, e' \in Expr : (\vdash e : t) \land e \rightsquigarrow e' \Rightarrow (\vdash e' : t).$$

Progress

$$\forall e \in Expr: (\vdash e : t) \Rightarrow e \in (Num \cup Bool) \vee \exists e' \in Expr: e \rightsquigarrow e'.$$

Termination

- Can be proved many ways.
- We show the proof based on the Energy function.
- First we define Energy function for basic expressions.

Termination

$$
\begin{aligned}
energy(v) &= 0, \\
energy(\triangle e) &= 1 + energy(e), \\
energy(e \odot e') &= 1 + energy(e) + energy(e'), \\
energy(\texttt{if}\ e\ \texttt{then}\ e'\ \texttt{else}\ e'' &= 1 + energy(e) + energy(e') \\
&\quad + energy(e'''),
\end{aligned}
$$

Termination

$$\forall e \in Expr\colon energy(e) \in N \text{ and}$$

$$\forall e, e' \in Expr\colon e \leadsto e' \Rightarrow energy(e) > energy(e').$$

Determinism

- A programming language is deterministic if there is exactly one possible outcome for any pair of program and inputs.

$$\forall v, v' \in (Num \cup Bool) \colon \forall e \in Expr \colon e \leadsto^{*} v \wedge e \leadsto^{*} v' \Rightarrow v = v'.$$