

# 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

$$S = \langle CF, \Rightarrow, FC, IF, OF \rangle$$

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

# Syntax

$$\begin{aligned} Expr ::= & Num \mid \\ & Bool \mid \\ & \Delta Expr \mid \\ & Expr \odot Expr \mid \\ & \text{if } Expr \text{ then } Expr \text{ else } Expr \end{aligned}$$

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

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

# Type System

$$\frac{}{\vdash n : \textit{Number}}$$
$$\frac{}{\vdash b : \textit{Boolean}}$$
$$\frac{\vdash e : \textit{Number}}{\vdash \Delta e : \textit{Number}}$$
$$\frac{\vdash e : \textit{Number} \quad \vdash e' : \textit{Number}}{\vdash e \odot e' : \textit{Number}}$$
$$\frac{\vdash e : \textit{Boolean} \quad \vdash e' : t \quad \vdash e'' : t}{\vdash \textit{if } e \textit{ then } e' \textit{ else } e'' : t}$$

# BOS

$$\overline{n \mapsto n}$$

$$\overline{b \mapsto b}$$

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

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

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

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

# SOS

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

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

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

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

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

# SOS

---

*if true then e' else e''*  $\rightsquigarrow$  *e'*

---

*if false then e' else e''*  $\rightsquigarrow$  *e''*

*e*  $\rightsquigarrow$  *e'''*

---

*if e then e' else e''*  $\rightsquigarrow$  *if e''' then e' else e''*

*e'*  $\rightsquigarrow$  *e'''*

---

*if e then e' else e''*  $\rightsquigarrow$  *if e then e''' else e''*

*e''*  $\rightsquigarrow$  *e'''*

---

*if e then e' else e''*  $\rightsquigarrow$  *if e then e' else e'''*

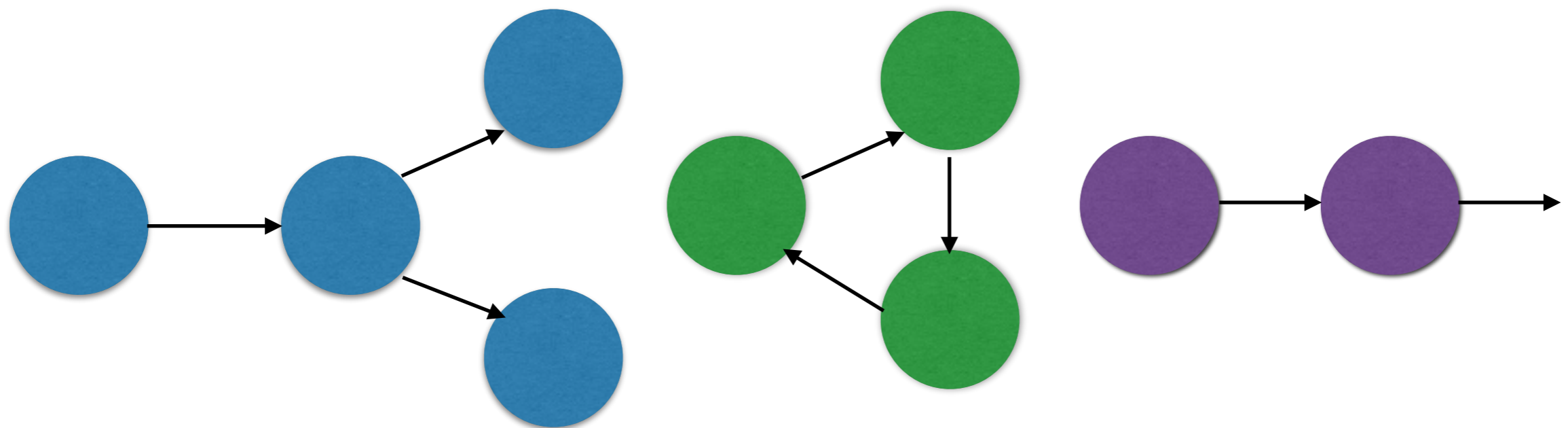


# Normal Form

- An irreducible configuration is also called a normal form.
- Each normal form is either a final configuration or a stuck state.

# Stuck States

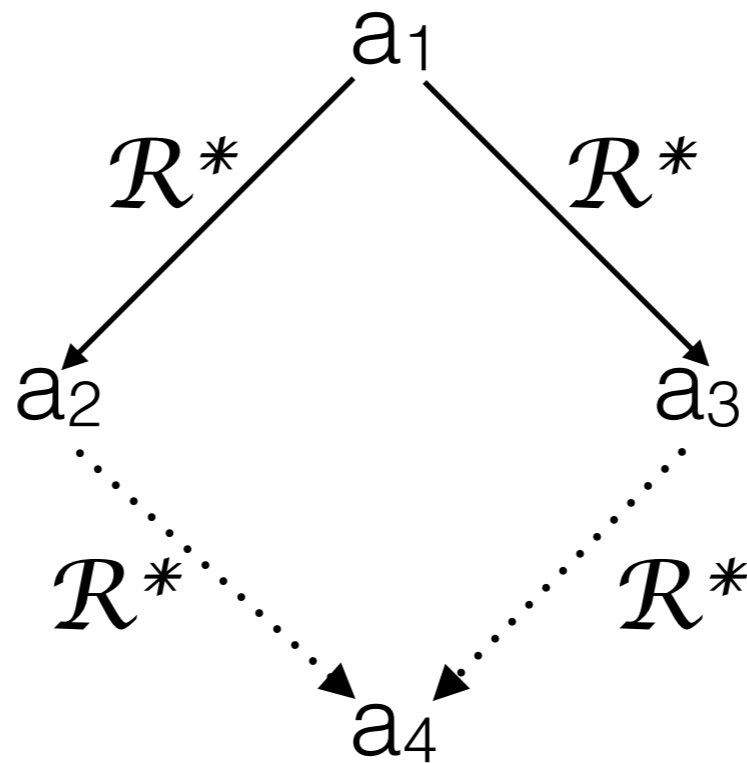
- Stuck States have no output according to output function.
- Stuck states include Cycling, Infinite Semantics, more than one normal form.



# Confluence

A relation  $R \subseteq A \times A$  is confluent iff

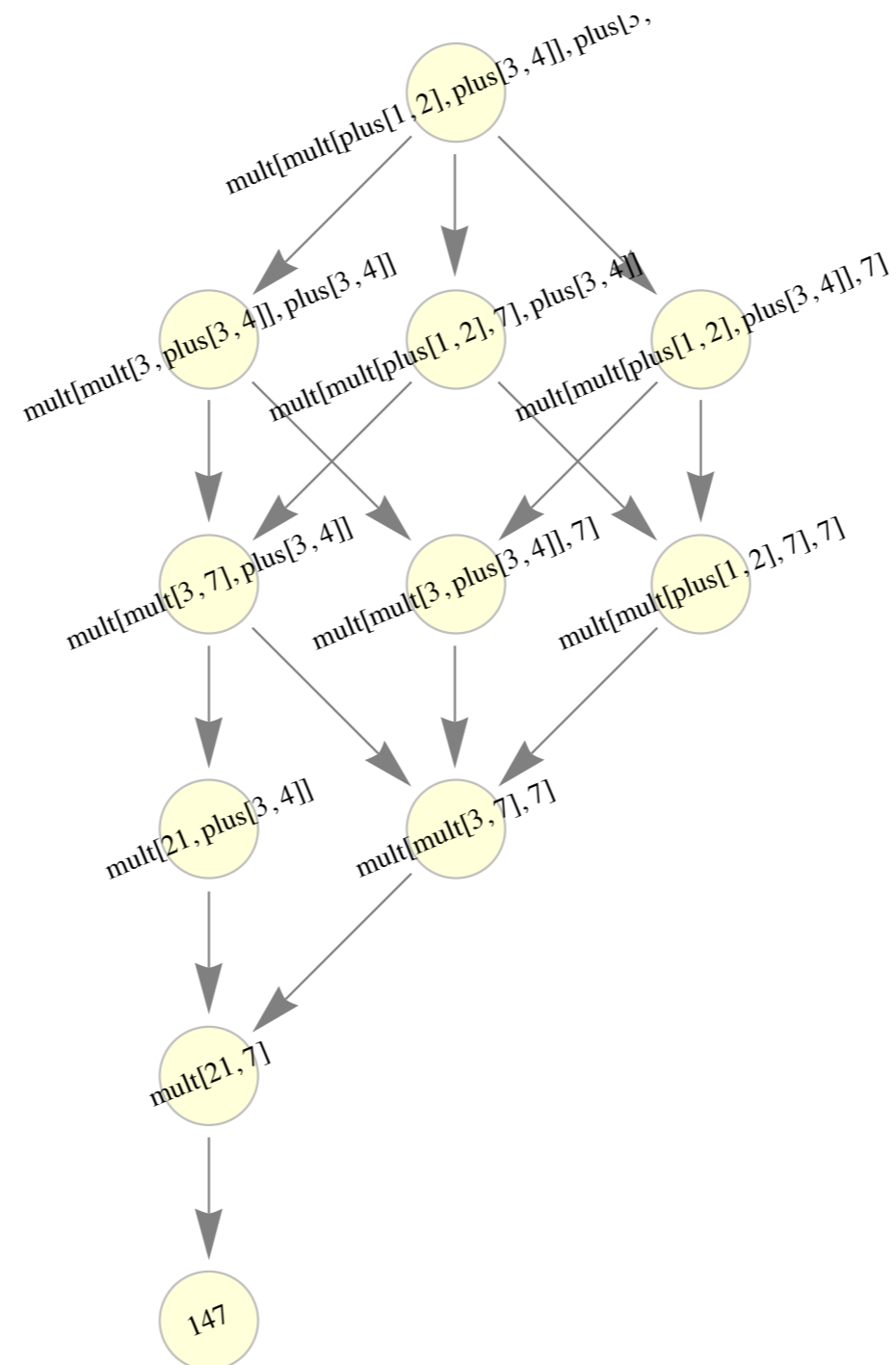
$$\forall a_1, a_2, a_3 \in A: a_1 R^* a_2 \wedge a_1 R^* a_3 \Rightarrow \exists a_4 \in A: a_2 R^* a_4 \wedge a_3 R^* a_4.$$



# Confluence

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

# Confluence



# 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. In special cases (universality and strong normalisation don't mix well), well typed programs also always terminate, i.e. they always produce an output value.

# Semantics Equivalence

$$\forall e \in \textit{Expr} : e \mapsto n \Leftrightarrow e \rightsquigarrow^* n.$$

# Type Preservation

$$\forall e, e' \in \textit{Expr} : (\vdash e : t) \wedge e \rightsquigarrow e' \Rightarrow (\vdash e' : t).$$



# 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.
- For more details see Turbak et al.

# Termination

$$\mathit{energy}(v) = 0,$$

$$\mathit{energy}(\Delta e) = 1 + \mathit{energy}(e),$$

$$\mathit{energy}(e \odot e') = 1 + \mathit{energy}(e) + \mathit{energy}(e'),$$

$$\mathit{energy}(\mathbf{if} \ e \ \mathbf{then} \ e' \ \mathbf{else} \ e'' = 1 + \mathit{energy}(e) + \mathit{energy}(e') \\ + \mathit{energy}(e'''),$$

# Termination

$\forall e \in Expr : energy(e) \in N$  and

$\forall e, e' \in Expr : e \rightsquigarrow e' \Rightarrow energy(e) > energy(e')$ .

# Determinism

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

$$\forall v, v' \in (Num \cup Bool) : \forall e \in Expr : e \rightsquigarrow^* v \wedge e \rightsquigarrow^* v' \Rightarrow v = v'.$$