Deciding equivalence using type checking

Colin Stirling

cps@inf.ed.ac.uk

LFCS
School of Informatics
University of Edinburgh

IHP Abstraction and Verification in Semantics
Paris, 27th June 2014

Methods for verifying finite and infinite state systems

- Notable success in Computer Science
- model checking + equivalence checking
- System = finite/infinite state transition graph

Methods for verifying finite and infinite state systems

- Notable success in Computer Science
- model checking + equivalence checking
- System = finite/infinite state transition graph
- Model checking: does state s have property Φ ?
- apply automata/game theoretic techniques to solve it: mostly computing monadic fixed points, reachability sets by traversing graph (possibly repeatedly)

Methods for verifying finite and infinite state systems

- Notable success in Computer Science
- model checking + equivalence checking
- System = finite/infinite state transition graph
- Model checking: does state s have property Φ ?
- apply automata/game theoretic techniques to solve it: mostly computing monadic fixed points, reachability sets by traversing graph (possibly repeatedly)
- Equivalence checking: is state s equivalent to t ?
- mostly computing dyadic fixed points e.g. bisimulations to solve it. May need algebraic/combinatorial properties of reachability sets/generators of graph

Transfer these techniques to systems with binding

1. Model checking higher-order trees/schemes
   Using parity automata and geometry of interaction or game semantics or Krivine machines; alternative type checking
   [Knapik, Niwinski, Urzyczyn 2002; Caucal 2002; Ong 2006; Hague, Murawski, Ong, Serre 2008, Kobayashi 2009; Kobayshi, Ong 2009; Salvati, Walukiewicz 2011; Kobayashi 2011 . . .]

Transfer these techniques to systems with binding

1. Model checking higher-order trees/schemes
   Using parity automata and geometry of interaction or game semantics or Krivine machines; alternative type checking

   [Knapik, Niwinski, Urzyczyn 2002; Caucal 2002; Ong 2006; Hague, Murawski, Ong, Serre 2008, Kobayashi 2009; Kobayshi, Ong 2009; Salvati, Walukiewicz 2011; Kobayashi 2011 . . .]

2. Deciding observational equivalence for fragments of idealized Algol and ML (w.r.t. finite value sets)
   Reduce to equivalence of automata (such as deterministic pushdown automata)

   [Ghica, McCusker 2000; Ong 2002; . . .; Hopkins, Murawski, Ong 2012; . . .]

Transfer these techniques to systems with binding

1. Model checking higher-order trees/schemes
   Using parity automata and geometry of interaction or game semantics or Krivine machines; alternative type checking
   [Knapik, Niwinski, Urzyczyn 2002; Caucal 2002; Ong 2006; Hague, Murawski, Ong, Serre 2008, Kobayashi 2009; Kobayshi, Ong 2009; Salvati, Walukiewicz 2011; Kobayashi 2011 . . .]
2. Deciding observational equivalence for fragments of idealized Algol and ML (w.r.t. finite value sets)
   Reduce to equivalence of automata (such as deterministic pushdown automata)
   [Ghica, McCusker 2000; Ong 2002; . . .; Hopkins, Murawski, Ong 2012; . . .]
3. $\vdots \quad \vdots \quad \vdots$

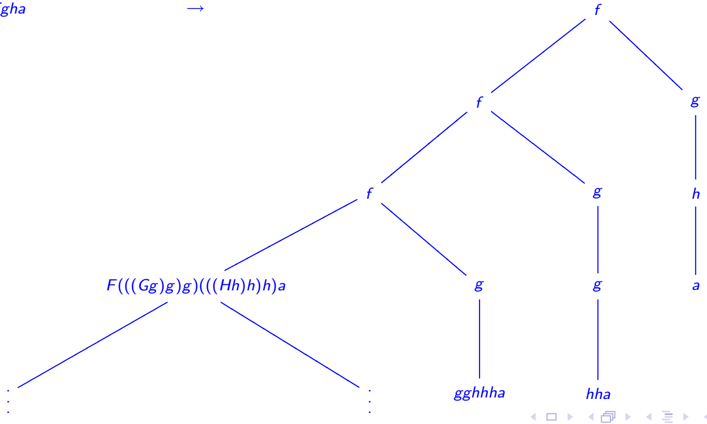
Example scheme: second-order

$$
\begin{array}{lcl}
F x_1 x_2 x_3 & \stackrel{\mathrm{def}}{=} & f\,(F(G x_1)(H x_2) x_3)\, x_1 (x_2 x_3) \\
G y_1 y_2 & \stackrel{\mathrm{def}}{=} & g(y_1(y_2)) \\
H z_1 z_2 & \stackrel{\mathrm{def}}{=} & h(z_1(z_2)) \\
F g h a & & \text{Start}
\end{array}
$$

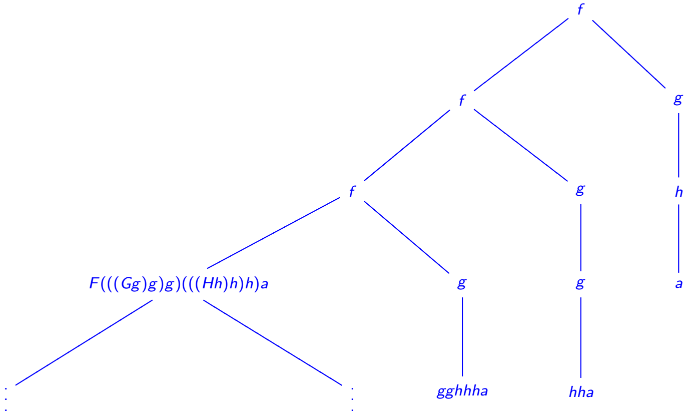
Example scheme: second-order

$$
\begin{aligned}
F x_1 x_2 x_3 &\stackrel{\text{def}}{=} f\,(F(Gx_1)(Hx_2)x_3)\,x_1(x_2x_3) \\
G y_1 y_2 &\stackrel{\text{def}}{=} g(y_1(y_2)) \\
H z_1 z_2 &\stackrel{\text{def}}{=} h(z_1(z_2)) \\
Fgha & \quad \textcolor{red}{\text{Start}}
\end{aligned}
$$

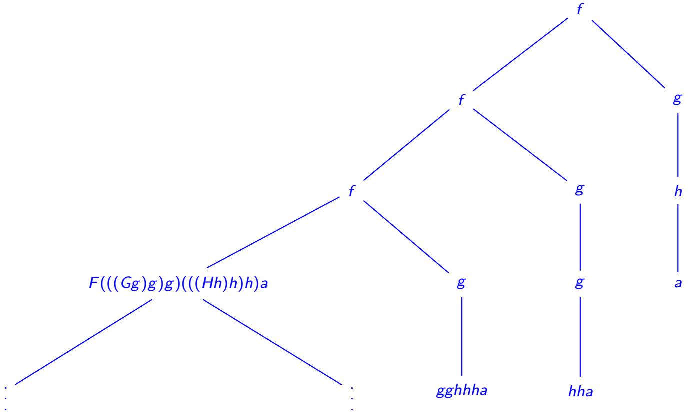
$Fgha \quad \longrightarrow$



Model checking a scheme

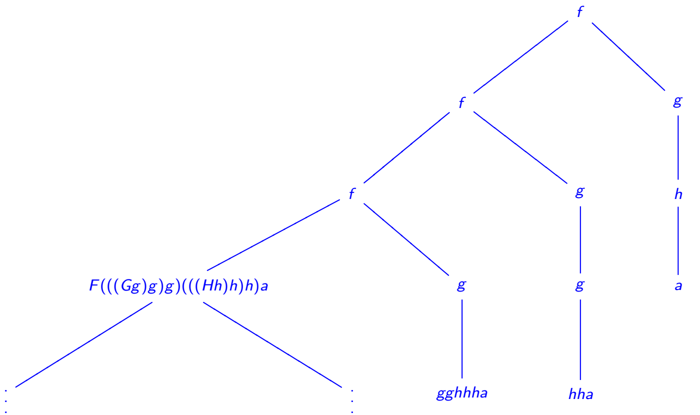


Model checking a scheme



Want to run a finite state automaton on tree generated by scheme

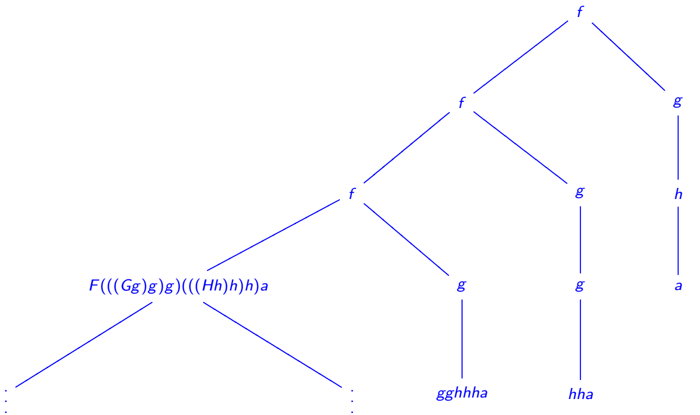
Model checking a scheme



Want to run a finite state automaton on tree generated by scheme

May have a transition $q \xrightarrow{f} (q_1, q_2)$

Model checking a scheme



Want to run a finite state automaton on tree generated by scheme

May have a transition $q \xrightarrow{f} (q_1, q_2)$

Need to do this on its finite (lambda calculus) description

Model checking a scheme

$$
\begin{array}{lcl}
F x_1 x_2 x_3 & \stackrel{\text{def}}{=} & f\,(F(Gx_1)(Hx_2)x_3)\,x_1(x_2x_3) \\
G y_1 y_2 & \stackrel{\text{def}}{=} & g(y_1(y_2)) \\
H z_1 z_2 & \stackrel{\text{def}}{=} & h(z_1(z_2)) \\
Fgha & & \text{Start}
\end{array}
$$

Want to run a finite state automaton on tree generated by scheme

May have a transition $q \xrightarrow{f} (q_1, q_2)$

Need to do this on its finite (lambda calculus) description

Model checking a scheme

$$\begin{array}{lcl} Fx_1x_2x_3 & \stackrel{\text{def}}{=} & f\,(F(Gx_1)(Hx_2)x_3)\,x_1(x_2x_3) \\ Gy_1y_2 & \stackrel{\text{def}}{=} & g(y_1(y_2)) \\ Hz_1z_2 & \stackrel{\text{def}}{=} & h(z_1(z_2)) \\ Fgha & & \text{Start} \end{array}$$

Want to run a finite state automaton on tree generated by scheme

May have a transition $q \xrightarrow{f} (q_1, q_2)$

Need to do this on its finite (lambda calculus) description

Application operator is essential component

Model checking a scheme

$$
\begin{array}{lcl}
F x_1 x_2 x_3 & \stackrel{\mathrm{def}}{=} & f\,(F(Gx_1)(Hx_2)x_3)\,x_1(x_2x_3) \\
G y_1 y_2 & \stackrel{\mathrm{def}}{=} & g(y_1(y_2)) \\
H z_1 z_2 & \stackrel{\mathrm{def}}{=} & h(z_1(z_2)) \\
Fgha & & \text{Start}
\end{array}
$$

Want to run a finite state automaton on tree generated by scheme

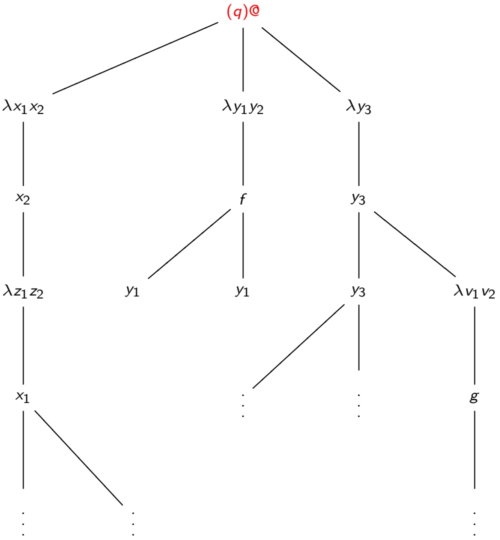
May have a transition $q \xrightarrow{f} (q_1, q_2)$

Need to do this on its finite (lambda calculus) description

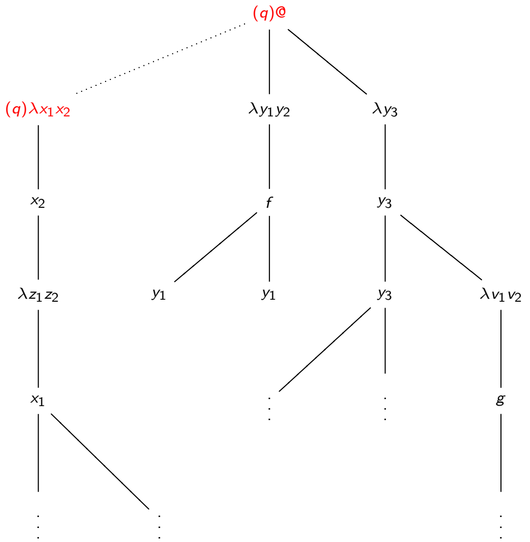
Application operator is essential component

Leads to non-standard automata

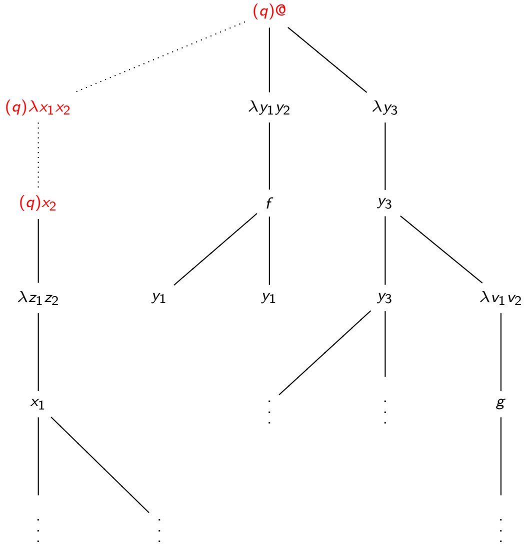
Automaton running on lambda term applied to two terms



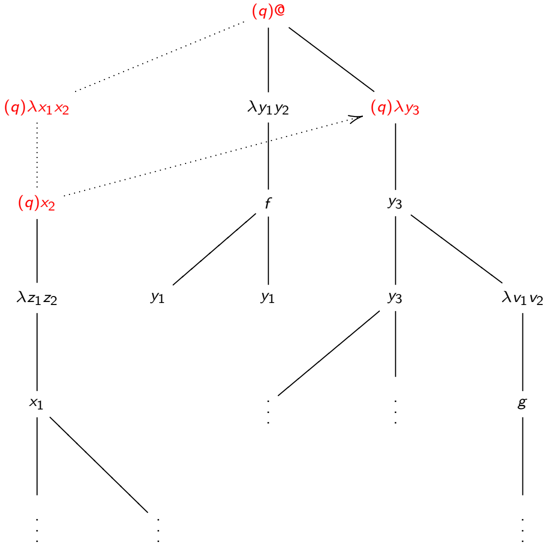
Automaton running on lambda term applied to two terms



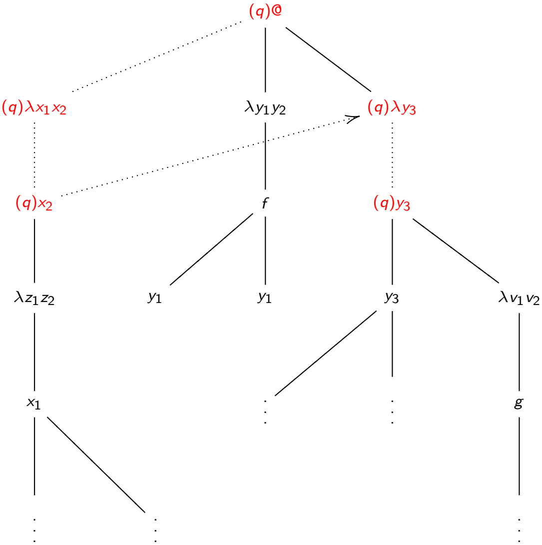
Automaton running on lambda term applied to two terms



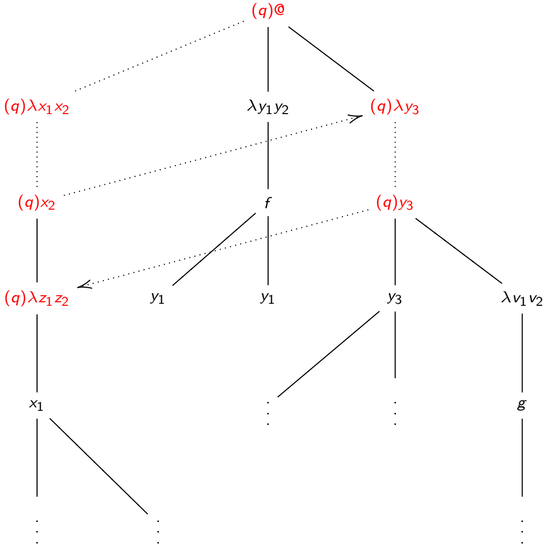
Automaton running on lambda term applied to two terms



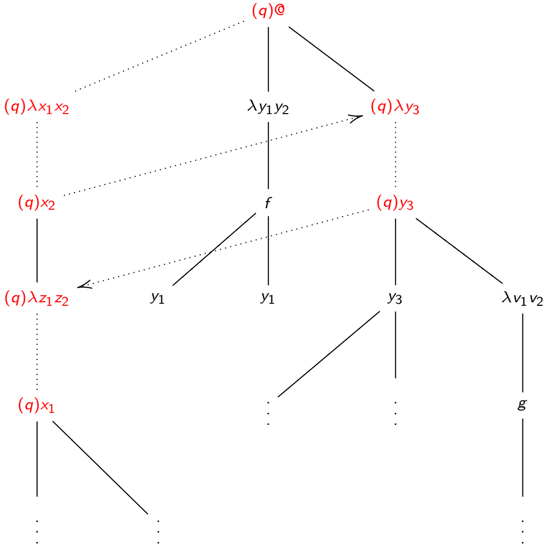
Automaton running on lambda term applied to two terms



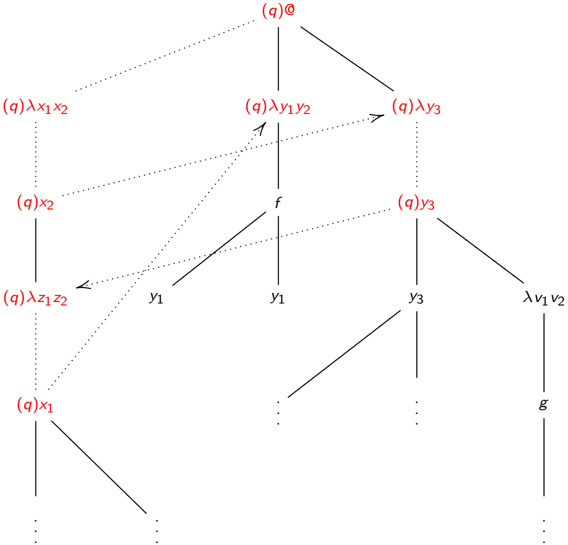
Automaton running on lambda term applied to two terms



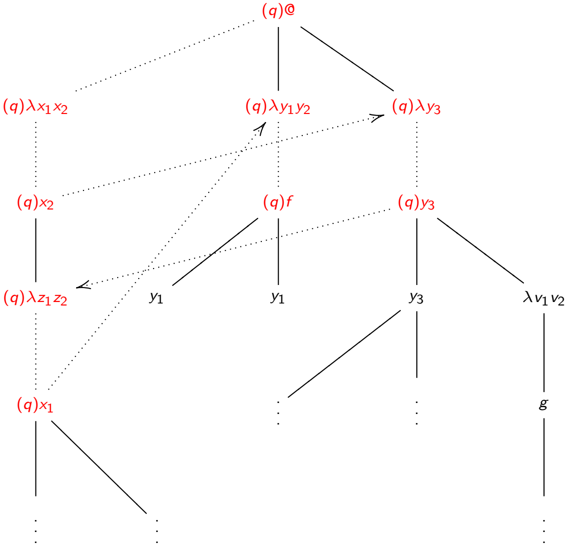
Automaton running on lambda term applied to two terms



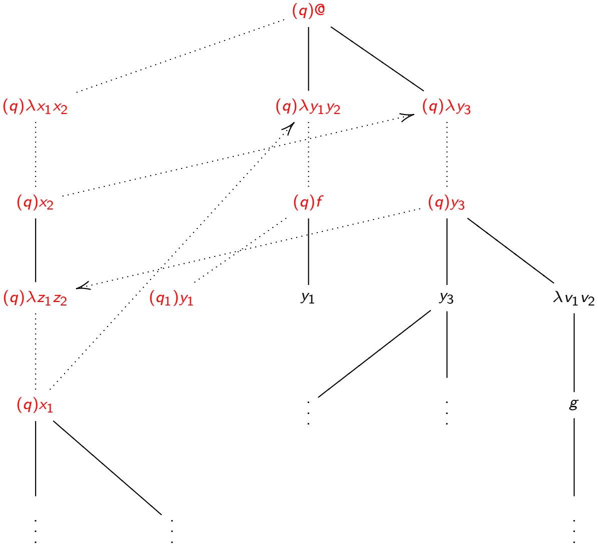
Automaton running on lambda term applied to two terms



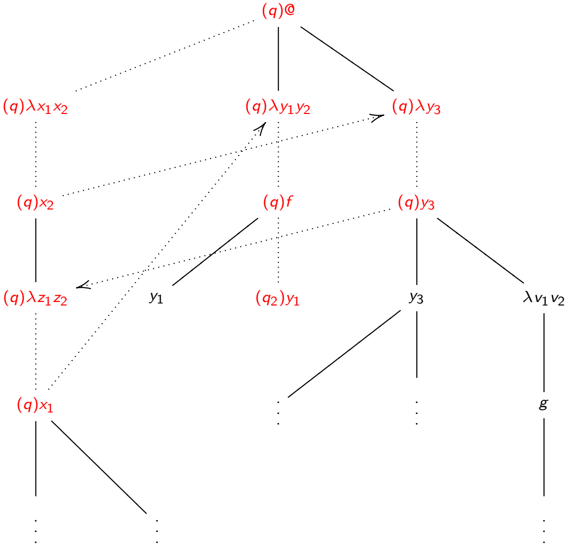
Automaton running on lambda term applied to two terms



Automaton running on lambda term applied to two terms



Automaton running on lambda term applied to two terms



Model checking as type checking

- Encode running of automaton using intersection types

Model checking as type checking

- Encode running of automaton using intersection types

$$
\begin{array}{lll}
\theta & := & q \mid \tau \to \theta \\
\tau & := & \bigwedge_{i_1 \in I_1} \theta_{i_1} \wedge \ldots \wedge \bigwedge_{i_m \in I_m} \theta_{i_m}
\end{array}
$$

q state of automaton

I_j finite

Model checking as type checking

- Encode running of automaton using intersection types

$$
\begin{array}{lcll}
\theta & := & q \mid \tau \to \theta & q \text{ state of automaton} \\
\tau & := & \bigwedge_{i_1 \in I_1} \theta_{i_1} \wedge \ldots \wedge \bigwedge_{i_m \in I_m} \theta_{i_m} & I_j \text{ finite}
\end{array}
$$

- Thm Automaton accepts Scheme iff $\vdash$ Scheme : initial state

Model checking as type checking

- Encode running of automaton using intersection types

$$
\begin{array}{lcll}
\theta & := & q \mid \tau \to \theta & q \text{ state of automaton} \\
\tau & := & \bigwedge_{i_1 \in I_1} \theta_{i_1} \wedge \ldots \wedge \bigwedge_{i_m \in I_m} \theta_{i_m} & I_j \text{ finite}
\end{array}
$$

- Thm Automaton accepts Scheme iff $\vdash$ Scheme : initial state
- Decision procedure: via the (finite) typing rules

Model checking as type checking

- Encode running of automaton using intersection types

$$\begin{array}{lcll} \theta & := & q \mid \tau \to \theta & q \text{ state of automaton} \\ \tau & := & \bigwedge_{i_1 \in I_1} \theta_{i_1} \wedge \ldots \wedge \bigwedge_{i_m \in I_m} \theta_{i_m} & I_j \text{ finite} \end{array}$$

- Thm Automaton accepts Scheme iff $\vdash$ Scheme : initial state
- Decision procedure: via the (finite) typing rules
- Can this technique work for equivalence checking ?

Model checking as type checking

- Encode running of automaton using intersection types

$$\begin{array}{lcll} \theta & := & q \mid \tau \to \theta & q \text{ state of automaton} \\ \tau & := & \bigwedge_{i_1 \in I_1} \theta_{i_1} \wedge \ldots \wedge \bigwedge_{i_m \in I_m} \theta_{i_m} & I_j \text{ finite} \end{array}$$

- Thm Automaton accepts Scheme iff $\vdash$ Scheme : initial state
- Decision procedure: via the (finite) typing rules
- Can this technique work for equivalence checking ?
- Use type checking to solve equivalence problem (for real-time strict deterministic pushdown automata)

Inspired by [Tsukada, Kobayshi 2012] which looks at special language inclusion problems

Real-time strict deterministic pushdown automata

Finite sets: Q states, Γ stack symbols, A alphabet and T basic transitions

Real-time strict deterministic pushdown automata

Finite sets: Q states, Γ stack symbols, A alphabet and T basic transitions

$pX \xrightarrow{a} q\alpha$ where $p, q \in Q$, $a \in A$, $X \in \Gamma$ and $\alpha \in \Gamma^*$

Real-time strict deterministic pushdown automata

Finite sets: Q states, Γ stack symbols, A alphabet and T basic transitions

$pX \xrightarrow{a} q\alpha$ where $p, q \in Q$, $a \in A$, $X \in \Gamma$ and $\alpha \in \Gamma^*$

Deterministic: if $pX \xrightarrow{a} q\alpha \in T$ and $pX \xrightarrow{a} r\gamma \in T$ then $q = r$ and $\alpha = \gamma$

Real-time strict deterministic pushdown automata

Finite sets: Q states, Γ stack symbols, A alphabet and T basic transitions
$pX \xrightarrow{a} q\alpha$ where $p, q \in Q$, $a \in A$, $X \in \Gamma$ and $\alpha \in \Gamma^*$
Deterministic: if $pX \xrightarrow{a} q\alpha \in T$ and $pX \xrightarrow{a} r\gamma \in T$ then $q = r$ and $\alpha = \gamma$
Configuration $p\alpha$ where $p \in Q$ and $\alpha \in \Gamma^*$

Real-time strict deterministic pushdown automata

Finite sets: Q states, Γ stack symbols, A alphabet and T basic transitions

$pX \xrightarrow{a} q\alpha$ where $p, q \in Q$, $a \in A$, $X \in \Gamma$ and $\alpha \in \Gamma^*$

Deterministic: if $pX \xrightarrow{a} q\alpha \in T$ and $pX \xrightarrow{a} r\gamma \in T$ then $q = r$ and $\alpha = \gamma$

Configuration $p\alpha$ where $p \in Q$ and $\alpha \in \Gamma^*$

Transitions of configuration: if $pX \xrightarrow{a} q\alpha \in T$ then $pX\beta \xrightarrow{a} q\alpha\beta$

Real-time strict deterministic pushdown automata

Finite sets: Q states, Γ stack symbols, A alphabet and T basic transitions

$pX \xrightarrow{a} q\alpha$ where $p, q \in Q$, $a \in A$, $X \in \Gamma$ and $\alpha \in \Gamma^*$

Deterministic: if $pX \xrightarrow{a} q\alpha \in T$ and $pX \xrightarrow{a} r\gamma \in T$ then $q = r$ and $\alpha = \gamma$

Configuration $p\alpha$ where $p \in Q$ and $\alpha \in \Gamma^*$

Transitions of configuration: if $pX \xrightarrow{a} q\alpha \in T$ then $pX\beta \xrightarrow{a} q\alpha\beta$

Transition relation extended to words $\xrightarrow{w}$, $w \in A^*$

Real-time strict deterministic pushdown automata

Finite sets: Q states, Γ stack symbols, A alphabet and T basic transitions

$pX \xrightarrow{a} q\alpha$ where $p, q \in Q$, $a \in A$, $X \in \Gamma$ and $\alpha \in \Gamma^*$

Deterministic: if $pX \xrightarrow{a} q\alpha \in T$ and $pX \xrightarrow{a} r\gamma \in T$ then $q = r$ and $\alpha = \gamma$

Configuration $p\alpha$ where $p \in Q$ and $\alpha \in \Gamma^*$

Transitions of configuration: if $pX \xrightarrow{a} q\alpha \in T$ then $pX\beta \xrightarrow{a} q\alpha\beta$

Transition relation extended to words $\xrightarrow{w}$, $w \in A^*$

Language accepted $L(p\alpha) = \{w \mid p\alpha \xrightarrow{w} q\varepsilon \text{ for some } q\}$

Real-time strict deterministic pushdown automata

Finite sets: Q states, Γ stack symbols, A alphabet and T basic transitions

$pX \xrightarrow{a} q\alpha$ where $p, q \in Q$, $a \in A$, $X \in \Gamma$ and $\alpha \in \Gamma^*$

Deterministic: if $pX \xrightarrow{a} q\alpha \in T$ and $pX \xrightarrow{a} r\gamma \in T$ then $q = r$ and $\alpha = \gamma$

Configuration $p\alpha$ where $p \in Q$ and $\alpha \in \Gamma^*$

Transitions of configuration: if $pX \xrightarrow{a} q\alpha \in T$ then $pX\beta \xrightarrow{a} q\alpha\beta$

Transition relation extended to words $\xrightarrow{w}$, $w \in A^*$

Language accepted $L(p\alpha) = \{w \mid p\alpha \xrightarrow{w} q\varepsilon \text{ for some } q\}$

Equivalence: given $p\alpha$ and $q\beta$ is $L(p\alpha) = L(q\beta)$?

Example

$Q = \{p, p_1, p_2, p_3\}$, $\Gamma = \{X, Y\}$ and $A = \{a, b, c\}$. T is

$$
\begin{array}{lll}
pX \xrightarrow{a} p_1X & pX \xrightarrow{b} p_2\epsilon & p_2X \xrightarrow{c} p_3X \\
p_1X \xrightarrow{a} pXX & p_1X \xrightarrow{b} p_3X & p_3X \xrightarrow{c} p_2\varepsilon \\
pY \xrightarrow{a} pYY & pY \xrightarrow{b} p_1\epsilon & p_1Y \xrightarrow{c} p_1\varepsilon
\end{array}
$$

Example

$Q = \{p, p_1, p_2, p_3\}$, $\Gamma = \{X, Y\}$ and $A = \{a, b, c\}$. T is

$$
\begin{array}{lll}
pX \xrightarrow{a} p_1X & pX \xrightarrow{b} p_2\epsilon & p_2X \xrightarrow{c} p_3X \\
p_1X \xrightarrow{a} pXX & p_1X \xrightarrow{b} p_3X & p_3X \xrightarrow{c} p_2\varepsilon \\
pY \xrightarrow{a} pYY & pY \xrightarrow{b} p_1\epsilon & p_1Y \xrightarrow{c} p_1\varepsilon
\end{array}
$$

$p_1YYY \xrightarrow{c} p_1YY$ because $p_1Y \xrightarrow{c} p_1\varepsilon \in T$

Example continued

$$
\begin{array}{ccccccccc}
pX & \xrightarrow{a} & p_1X & \xrightarrow{a} & pXX & \xrightarrow{a} & p_1XX & \xrightarrow{a} & \dots \\
\downarrow b & & \downarrow b & & \downarrow b & & \downarrow b & & \dots \\
p_2\varepsilon & \xleftarrow{c} & p_3X & \xleftarrow{c} & p_2X & \xleftarrow{c} & p_3XX & \xleftarrow{c} & \dots
\end{array}
$$

$$
\begin{array}{ccccccccc}
pY & \xrightarrow{a} & pYY & \xrightarrow{a} & pYYY & \xrightarrow{a} & pYYYY & \xrightarrow{a} & \dots \\
\downarrow b & & \downarrow b & & \downarrow b & & \downarrow b & & \dots \\
p_1\varepsilon & \xleftarrow{c} & p_1Y & \xleftarrow{c} & p_1YY & \xleftarrow{c} & p_1YYY & \xleftarrow{c} & \dots
\end{array}
$$

$$\text{For } n > 0, \quad L(pX^n) = L(pY^{2n-1})$$

Where is application?

- Assume states $\{p_1, \ldots, p_k\}$

Where is application?

- Assume states $\{p_1, \ldots, p_k\}$
- Configuration $p\gamma\alpha$ is $p\gamma$ applied to $p_1\alpha, \ldots, p_k\alpha$

Where is application?

- Assume states $\{p_1, \ldots, p_k\}$
- Configuration $p\gamma\alpha$ is $p\gamma$ applied to $p_1\alpha, \ldots, p_k\alpha$
- Types $\tau ::= (\theta_1, \ldots, \theta_k) \rightarrow q\beta$
- $q\beta$ is a configuration and θ_i finite set of configurations

Meaning of a type

$p\alpha : (\theta_1, \ldots, \theta_k) \to q\beta$ iff

Meaning of a type

$p\alpha : (\theta_1, \ldots, \theta_k) \to q\beta$ iff

1. $p\alpha \trianglelefteq q\beta$ (consonance)
2. if $p\alpha \xrightarrow{w} p_i\varepsilon$ and $q\beta \xrightarrow{w} q'\gamma$ then $q'\gamma \in \theta_i$

where $p\alpha \trianglelefteq q\beta$ iff

- if $w \in L(p\alpha)$, then there is a v, $wv \in L(q\beta)$
- if $w \in L(q\beta)$, then there is a prefix of w, $w' \in L(p\alpha)$

Meaning of a type

$p\alpha : (\theta_1, \dots, \theta_k) \to q\beta$ iff

1. $p\alpha \trianglelefteq q\beta$ (consonance)
2. if $p\alpha \xrightarrow{w} p_i\varepsilon$ and $q\beta \xrightarrow{w} q'\gamma$ then $q'\gamma \in \theta_i$

where $p\alpha \trianglelefteq q\beta$ iff

- if $w \in L(p\alpha)$, then there is a v, $wv \in L(q\beta)$
- if $w \in L(q\beta)$, then there is a prefix of w, $w' \in L(p\alpha)$

Key property: there is a m, if $p\alpha \trianglelefteq q\beta$ then there is a prefix β' of β, $p\alpha \trianglelefteq q\beta'$ and $|\beta'| \leq m|\alpha|$

Proof system

Type assumptions $qX : \tau \in \Delta$

Proof system

Type assumptions $qX : \tau \in \Delta$

Axiom

$\Delta \vdash pX : (\theta_1, \ldots, \theta_k) \to q\beta\delta$ if $pX : (\theta'_1, \ldots, \theta'_k) \to q\beta \in \Delta$ and $\theta_i = \{r\lambda\delta \mid r\lambda \in \theta'_i\}$

Proof system

Type assumptions $qX : \tau \in \Delta$

Axiom

$\Delta \vdash pX : (\theta_1, \ldots, \theta_k) \rightarrow q\beta\delta$ if $pX : (\theta'_1, \ldots, \theta'_k) \rightarrow q\beta \in \Delta$ and $\theta_i = \{r\lambda\delta \mid r\lambda \in \theta'_i\}$

Application rule

$$\frac{\Delta \vdash pX : (\theta'_1, \ldots, \theta'_k) \rightarrow q\beta \ \ldots, \Delta \vdash p_j Y\alpha : (\theta^{ji}_1, \ldots, \theta^{ji}_k) \rightarrow r_{ji}\lambda_{ji}}{\Delta \vdash pXY\alpha : (\theta_1, \ldots, \theta_k) \rightarrow q\beta}$$

for all j and $r_{ji}\lambda_{ji} \in \theta'_j$ and $\theta_m = \bigcup_j \bigcup_i \theta^{ji}_m$

Proof system

Type assumptions $qX : \tau \in \Delta$

Axiom

$\Delta \vdash pX : (\theta_1, \ldots, \theta_k) \to q\beta\delta$ if $pX : (\theta'_1, \ldots, \theta'_k) \to q\beta \in \Delta$ and $\theta_i = \{r\lambda\delta \mid r\lambda \in \theta'_i\}$

Application rule

$$\frac{\Delta \vdash pX : (\theta'_1, \ldots, \theta'_k) \to q\beta \ \ldots, \Delta \vdash p_j Y\alpha : (\theta^{ji}_1, \ldots, \theta^{ji}_k) \to r_{ji}\lambda_{ji}}{\Delta \vdash pXY\alpha : (\theta_1, \ldots, \theta_k) \to q\beta}$$

for all j and $r_{ji}\lambda_{ji} \in \theta'_j$ and $\theta_m = \bigcup_j \bigcup_i \theta^{ji}_m$

Δ needs to be closed under transitions

Δ closed under transitions

If $pX : (\theta_1, \ldots, \theta_k) \rightarrow q\beta \in \Delta$ then

Δ closed under transitions

If $pX : (\theta_1, \ldots, \theta_k) \rightarrow q\beta \in \Delta$ then

1. if $pX \xrightarrow{a} p'\alpha$ then $q\beta \xrightarrow{a} q'\beta'$ and vice versa

Δ closed under transitions

If $pX : (\theta_1, \ldots, \theta_k) \to q\beta \in \Delta$ then

1. if $pX \xrightarrow{a} p'\alpha$ then $q\beta \xrightarrow{a} q'\beta'$ and vice versa
2. if $pX \xrightarrow{a} p_i\varepsilon$ and $q\beta \xrightarrow{a} q'\beta'$ then $q'\beta' \in \theta_i$

Δ closed under transitions

If $pX : (\theta_1, \ldots, \theta_k) \rightarrow q\beta \in \Delta$ then

1. if $pX \xrightarrow{a} p'\alpha$ then $q\beta \xrightarrow{a} q'\beta'$ and vice versa
2. if $pX \xrightarrow{a} p_i\varepsilon$ and $q\beta \xrightarrow{a} q'\beta'$ then $q'\beta' \in \theta_i$
3. if $pX \xrightarrow{a} rZ\alpha$ and $q\beta \xrightarrow{a} q'\beta'$ then
   $\Delta \vdash rZ\alpha : (\theta_1', \ldots, \theta_k) \rightarrow q'\beta'$ for $\theta_i' \subseteq \theta_i$

Δ closed under transitions

If $pX : (\theta_1, \ldots, \theta_k) \rightarrow q\beta \in \Delta$ then

1. if $pX \xrightarrow{a} p'\alpha$ then $q\beta \xrightarrow{a} q'\beta'$ and vice versa
2. if $pX \xrightarrow{a} p_i\varepsilon$ and $q\beta \xrightarrow{a} q'\beta'$ then $q'\beta' \in \theta_i$
3. if $pX \xrightarrow{a} rZ\alpha$ and $q\beta \xrightarrow{a} q'\beta'$ then
   $\Delta \vdash rZ\alpha : (\theta'_1, \ldots, \theta_k) \rightarrow q'\beta'$ for $\theta'_i \subseteq \theta_i$
4. θ_i is union of cases 2 and 3

Equivalence checking as type checking

- Reduce $L(p\alpha) = L(q\beta)$? to

Equivalence checking as type checking

- Reduce $L(p\alpha) = L(q\beta)$? to
- $\Delta \vdash p\alpha : (\theta^0_1, \ldots, \theta^0_k) \rightarrow q\beta$? where each $\theta^0_i \subseteq \{p_1\varepsilon, \ldots, p_k\varepsilon\}$

Example

$$
\begin{array}{ccccccccc}
pX & \xrightarrow{a} & p_1X & \xrightarrow{a} & pXX & \xrightarrow{a} & p_1XX & \xrightarrow{a} & \ldots \\
\downarrow b & & \downarrow b & & \downarrow b & & \downarrow b & & \ldots \\
p_2\varepsilon & \xleftarrow{c} & p_3X & \xleftarrow{c} & p_2X & \xleftarrow{c} & p_3XX & \xleftarrow{c} & \ldots
\end{array}
$$

$$
\begin{array}{ccccccccc}
pY & \xrightarrow{a} & pYY & \xrightarrow{a} & pYYY & \xrightarrow{a} & pYYYY & \xrightarrow{a} & \ldots \\
\downarrow b & & \downarrow b & & \downarrow b & & \downarrow b & & \ldots \\
p_1\varepsilon & \xleftarrow{c} & p_1Y & \xleftarrow{c} & p_1YY & \xleftarrow{c} & p_1YYY & \xleftarrow{c} & \ldots
\end{array}
$$

Assume states are ordered p, p_1, p_2, p_3 and $\pi = (\emptyset, \emptyset, \{p_1\varepsilon\}, \emptyset)$

For any $n > 0$, $\quad \Delta \vdash pX^n : \pi \rightarrow pY^{2n-1}$

where $\Delta = \{pX : \pi \rightarrow pY, p_1X : \pi \rightarrow pYY, p_2X : \pi \rightarrow p_1YY, p_3X : \pi \rightarrow p_1Y\}$

Proof tree upside down

$$\cfrac{\Delta \vdash pX^4 : \pi \longrightarrow pY^7}{\Delta \vdash pX : (\emptyset, \emptyset, \{p_1 Y^6\}, \emptyset) \longrightarrow pY^7 \qquad \cfrac{\Delta \vdash p_2 X^3 : \pi \longrightarrow p_1 Y^6}{\Delta \vdash p_2 X : (\emptyset, \emptyset, \{p_1 Y^4\}, \emptyset) \longrightarrow p_1 Y^6 \qquad \ldots}}$$

where $\ldots$ is the subtree

$$\cfrac{\Delta \vdash p_2 X^2 : \pi \longrightarrow p_1 Y^4}{\Delta \vdash p_2 X : (\emptyset, \{p_1 Y^2\}, \emptyset, \emptyset) \longrightarrow p_1 Y^4 \qquad \Delta \vdash p_2 X : \pi \longrightarrow p_1 Y^2}$$

where $\Delta = \{pX : \pi \longrightarrow pY, p_1 X : \pi \longrightarrow pYY, p_2 X : \pi \longrightarrow p_1 YY, p_3 X : \pi \longrightarrow p_1 Y\}$ and $\pi = (\emptyset, \emptyset, \{p_1 \varepsilon\}, \emptyset)$

Δ closed under transitions

$$
\begin{array}{lcr}
pX & : \pi \longrightarrow & pY \\
\downarrow a & & \downarrow a \\
p_1X & : \pi \longrightarrow & pY^2
\end{array}
\qquad
\begin{array}{lcr}
pX & : \pi \longrightarrow & pY \\
\downarrow b & & \downarrow b \\
p_2\varepsilon & & p_1\varepsilon
\end{array}
\qquad
\begin{array}{lcr}
p_1X & : \pi \longrightarrow & pY^2 \\
\downarrow a & & \downarrow a \\
pX^2 & : \pi \longrightarrow & pY^3
\end{array}
$$

$$
\begin{array}{lcr}
p_1X & : \pi \longrightarrow & pY^2 \\
\downarrow b & & \downarrow b \\
p_3X & : \pi \longrightarrow & p_1Y
\end{array}
\qquad
\begin{array}{lcr}
p_2X & : \pi \longrightarrow & p_1Y^2 \\
\downarrow c & & \downarrow c \\
p_3X & : \pi \longrightarrow & p_1Y
\end{array}
\qquad
\begin{array}{lcr}
p_3X & : \pi \longrightarrow & p_1Y \\
\downarrow c & & \downarrow c \\
p_2\varepsilon & & p_1\varepsilon
\end{array}
$$

$$
\frac{\Delta \vdash pX : (\emptyset, \emptyset, \{p_1Y^2\}, \emptyset) \longrightarrow pY^3 \qquad \Delta \vdash p_2X : \pi \longrightarrow p_1Y^2}{\Delta \vdash pX^2 : \pi \longrightarrow pY^3}
$$

where $\Delta = \{pX : \pi \longrightarrow pY, p_1X : \pi \longrightarrow pYY, p_2X : \pi \longrightarrow p_1YY, p_3X : \pi \longrightarrow p_1Y\}$ and $\pi = (\emptyset, \emptyset, \{p_1\varepsilon\}, \emptyset)$

Conclusion

- Use type checking to solve equivalence problem (for real-time strict deterministic pushdown automata)

Conclusion

- Use type checking to solve equivalence problem (for real-time strict deterministic pushdown automata)
- [Oyamaguchi, Honda, Inagaki 1980] showed decidability without complexity upper bound

  Only known upper bound is the one for equivalence of full deterministic pushdown automata [Stirling 2002]

Conclusion

- Use type checking to solve equivalence problem (for real-time strict deterministic pushdown automata)
- [Oyamaguchi, Honda, Inagaki 1980] showed decidability without complexity upper bound

  Only known upper bound is the one for equivalence of full deterministic pushdown automata [Stirling 2002]
- Type checking algorithm and its correctness proof much simpler than [S 2002]. Like [OHI 1980] and unlike [S 2002] algorithm is nondeterminisitic

Conclusion

- Use type checking to solve equivalence problem (for real-time strict deterministic pushdown automata)
- [Oyamaguchi, Honda, Inagaki 1980] showed decidability without complexity upper bound

  Only known upper bound is the one for equivalence of full deterministic pushdown automata [Stirling 2002]
- Type checking algorithm and its correctness proof much simpler than [S 2002]. Like [OHI 1980] and unlike [S 2002] algorithm is nondeterminisitic
- At one stage I was convinced it did lead to better bound via an upper bound on m in key property

  Key property: there is a m, if $pX \trianglelefteq q\beta$ then there is a prefix β' of β, $pX \trianglelefteq q\beta'$ and $|\beta'| \leq m$

Conclusion

- Use type checking to solve equivalence problem (for real-time strict deterministic pushdown automata)
- [Oyamaguchi, Honda, Inagaki 1980] showed decidability without complexity upper bound

  Only known upper bound is the one for equivalence of full deterministic pushdown automata [Stirling 2002]
- Type checking algorithm and its correctness proof much simpler than [S 2002]. Like [OHI 1980] and unlike [S 2002] algorithm is nondeterminisitic
- At one stage I was convinced it did lead to better bound via an upper bound on m in key property

  Key property: there is a m, if $pX \trianglelefteq q\beta$ then there is a prefix β' of β, $pX \trianglelefteq q\beta'$ and $|\beta'| \leq m$
- Does the technique naturally extend to schema?