An Introduction to Decidability of Higher-Order Matching

Colin Stirling
cps@inf.ed.ac.uk

LFCS
School of Informatics
University of Edinburgh

GALOP, London, 18th July 2013

Simply typed λ-calculus

- Types $A ::= \mathbf{0} \mid A \to A$
  - $\mathbf{0}$ single base type (for simplicity)
  - $A \to B$ type of functions from A to B

Simply typed λ-calculus

- Types $A ::= \mathbf{0} \mid A \rightarrow A$
  - $\mathbf{0}$ single base type (for simplicity)
  - $A \rightarrow B$ type of functions from A to B
- If $A \neq \mathbf{0}$ then A has form $A_1 \rightarrow \ldots \rightarrow A_n \rightarrow \mathbf{0}$
  abbreviated to $(A_1, \ldots, A_n, \mathbf{0})$

Simply typed λ-calculus

- Types $A ::= \mathbf{0} \;|\; A \to A$
  - $\mathbf{0}$ single base type (for simplicity)
  - $A \to B$ type of functions from A to B
- If $A \neq \mathbf{0}$ then A has form $A_1 \to \ldots \to A_n \to \mathbf{0}$ abbreviated to $(A_1, \ldots, A_n, \mathbf{0})$
- Order of $\mathbf{0}$ is 1

Simply typed λ-calculus

- Types $A ::= \mathbf{0} \mid A \to A$
  - $\mathbf{0}$ single base type (for simplicity)
  - $A \to B$ type of functions from A to B
- If $A \neq \mathbf{0}$ then A has form $A_1 \to \ldots \to A_n \to \mathbf{0}$ abbreviated to $(A_1, \ldots, A_n, \mathbf{0})$
- Order of $\mathbf{0}$ is 1
- Order of $(A_1, \ldots, A_n, \mathbf{0})$ is $k + 1$ where k is maximum of orders of A_is

Terms of simply type λ-calculus

Variables and constants each have a unique type (Church style)

Terms of simply type λ-calculus

Variables and constants each have a unique type (Church style)

1. if x (f) has type A then $x : A$ ($f : A$)
2. if $t : B$ and $x : A$ then $\lambda x.t : A \rightarrow B$
3. if $t : A \rightarrow B$ and $u : A$ then $(tu) : B$

Terms of simply type λ-calculus

Variables and constants each have a unique type (Church style)

1. if x (f) has type A then $x : A$ ($f : A$)
2. if $t : B$ and $x : A$ then $\lambda x.t : A \to B$
3. if $t : A \to B$ and $u : A$ then $(tu) : B$

- order of $t : A$ is order A
- closed $t : A$ no free variables
- $t, t' : A$ are α-equivalent renamings of each other

Dynamics: reduction

$$(\beta) \quad (\lambda x.t)v \rightarrow_\beta t\{v/x\} \qquad \{\cdot/\cdot\} \text{ Substitution}$$
$$(\eta) \quad \lambda x.(tx) \rightarrow_\eta t \qquad x \text{ not free in } t$$

Dynamics: reduction

$$(\beta) \quad (\lambda x.t)v \to_\beta t\{v/x\} \qquad \{\cdot/\cdot\} \text{ Substitution}$$
$$(\eta) \quad \lambda x.(tx) \to_\eta t \qquad x \text{ not free in } t$$

- Facts
  Strong normalisation and confluence (of $\to_\beta$, $\to_\eta$, $\to_{\beta\eta}$)

Dynamics: reduction

$$(\beta) \quad (\lambda x.t)v \to_\beta t\{v/x\} \qquad \{\cdot/\cdot\} \text{ Substitution}$$
$$(\eta) \quad \lambda x.(tx) \to_\eta t \qquad x \text{ not free in } t$$

- Facts
  Strong normalisation and confluence (of $\to_\beta$, $\to_\eta$, $\to_{\beta\eta}$)
- Equivalence: $t =_{\beta\eta} t'$ if there are s, s'

$$\begin{array}{ccc} t & \to^*_{\beta\eta} & s \\ & & \shortparallel_\alpha \\ t' & \to^*_{\beta\eta} & s' \end{array}$$

Long normal forms

- t in β-normal form if no t' such that $t \rightarrow_\beta t'$

Long normal forms

- t in β-normal form if no t' such that $t \longrightarrow_\beta t'$
- t in η-long β-normal form if t is in β-normal form and there is no t' such that $t' \longrightarrow_\eta t$

Long normal forms

- t in β-normal form if no t' such that $t \to_\beta t'$
- t in η-long β-normal form if t is in β-normal form and there is no t' such that $t' \to_\eta t$
- lnf $= \eta$-long β-normal form
- If t, v are in lnf and $tv \to_\beta^* s$ in β-normal form then s is in lnf
  No η-reductions when terms in lnf

Long normal forms

- t in β-normal form if no t' such that $t \rightarrow_\beta t'$
- t in η-long β-normal form if t is in β-normal form and there is no t' such that $t' \rightarrow_\eta t$
- Inf $= \eta$-long β-normal form
- If t, v are in Inf and $tv \rightarrow^*_\beta s$ in β-normal form then s is in Inf
  No η-reductions when terms in Inf
- $T_A(C)$ is the set of closed Inf terms of type A whose constants belong to C.

Example

- $x : (\mathbf{0}, \mathbf{0})$ $\lambda x.x : ((\mathbf{0}, \mathbf{0}), \mathbf{0}, \mathbf{0})$ in β-normal form not Inf

Example

- $x : (\mathbf{0}, \mathbf{0})$ $\lambda x.x : ((\mathbf{0}, \mathbf{0}), \mathbf{0}, \mathbf{0})$ in β-normal form not Inf
- $z : \mathbf{0}$ then $\lambda z.xz : (\mathbf{0}, \mathbf{0})$ and $\lambda xz.xz : ((\mathbf{0}, \mathbf{0}), \mathbf{0}, \mathbf{0})$ are in Inf
- Notation: $\lambda x_1 \ldots x_k.t$ abbreviates $\lambda x_1. \ldots .\lambda x_k.t$

Example

- $x : (\mathbf{0},\mathbf{0})$ $\lambda x.x : ((\mathbf{0},\mathbf{0}),\mathbf{0},\mathbf{0})$ in β-normal form not Inf
- $z : \mathbf{0}$ then $\lambda z.xz : (\mathbf{0},\mathbf{0})$ and $\lambda xz.xz : ((\mathbf{0},\mathbf{0}),\mathbf{0},\mathbf{0})$ are in Inf
- Notation: $\lambda x_1 \ldots x_k.t$ abbreviates $\lambda x_1. \ldots \lambda x_k.t$
- Monster type $M = ((((\mathbf{0},\mathbf{0}),\mathbf{0}),\mathbf{0}),\mathbf{0},\mathbf{0})$ has order 5
- $\lambda xy.x(\lambda z_1.x(\lambda z_2.z_1 y)) \in T_M(\emptyset)$
  when $x : (((\mathbf{0},\mathbf{0}),\mathbf{0}),\mathbf{0})$, $z_i : (\mathbf{0},\mathbf{0})$ and $y : \mathbf{0}$.

Decision questions

- Higher-order unification
- $v = u$ contains free variables $x_1, \ldots, x_n$
- Solution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ such that $v\theta =_{\beta\eta} u\theta$
  Simultaneous substitution

Decision questions

- Higher-order unification
- $v = u$ contains free variables $x_1, \ldots, x_n$
- Solution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ such that $v\theta =_{\beta\eta} u\theta$
  Simultaneous substitution
- Decision question: given $v = u$, does it have a solution ?
- Order is max order of the x_is

Decision questions

- Higher-order unification
- $v = u$ contains free variables $x_1, \ldots, x_n$
- Solution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ such that $v\theta =_{\beta\eta} u\theta$
  Simultaneous substitution
- Decision question: given $v = u$, does it have a solution ?
- Order is max order of the x_is
- Undecidable (even at order 2) [Huet 1972;Goldfarb 1981]

Decision questions (cont)

- Higher-order matching
- $v = u$ contains free variables $x_1, \ldots, x_n$ BUT u closed
- Solution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ such that $v\theta =_{\beta\eta} u$
  W.l.o.g assume $v, u : \mathbf{0}$

Decision questions (cont)

- Higher-order matching
- $v = u$ contains free variables $x_1, \ldots, x_n$ BUT u closed
- Solution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ such that $v\theta =_{\beta\eta} u$
  W.l.o.g assume $v, u : \mathbf{0}$
- Decision question: given $v = u$, does it have a solution ?
- Order is max order of the x_is

Decision questions (cont)

- Higher-order matching
- $v = u$ contains free variables $x_1, \ldots, x_n$ BUT u closed
- Solution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ such that $v\theta =_{\beta\eta} u$
  W.l.o.g assume $v, u : \mathbf{0}$
- Decision question: given $v = u$, does it have a solution ?
- Order is max order of the x_is
- Huet conjecture decidable [Huet 1976]
- Up to order 4 decidable + special cases [Huet 1976, Dowek 1993, Padovani 2000, ...]

Decision questions (cont)

- Higher-order matching
- $v = u$ contains free variables $x_1, \ldots, x_n$ BUT u closed
- Solution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ such that $v\theta =_{\beta\eta} u$
  W.l.o.g assume $v, u : \mathbf{0}$
- Decision question: given $v = u$, does it have a solution ?
- Order is max order of the x_is
- Huet conjecture decidable [Huet 1976]
- Up to order 4 decidable + special cases [Huet 1976, Dowek 1993, Padovani 2000, ...]
- Undecidable for $=_\beta$ [Loader 2003]

Decision questions (cont)

- Higher-order matching
- $v = u$ contains free variables $x_1, \ldots, x_n$ BUT u closed
- Solution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ such that $v\theta =_{\beta\eta} u$
  W.l.o.g assume $v, u : \mathbf{0}$
- Decision question: given $v = u$, does it have a solution ?
- Order is max order of the x_is
- Huet conjecture decidable [Huet 1976]
- Up to order 4 decidable + special cases [Huet 1976, Dowek 1993, Padovani 2000, ...]
- Undecidable for $=_\beta$ [Loader 2003]
- Decidable for all orders [Stirling 2006; 2009; 2012]

Matching is essentially monadic

- Given $v = u$ with $x_1 : A_1, \ldots, x_n : A_n$ in v
- there is the matching problem $x(\lambda x_1 \ldots x_n.v) = u$
  where $x : ((A_1, \ldots, A_n, \mathbf{0}), \mathbf{0})$

Matching is essentially monadic

- Given $v = u$ with $x_1 : A_1, \ldots, x_n : A_n$ in v
- there is the matching problem $x(\lambda x_1 \ldots x_n.v) = u$ where $x : ((A_1, \ldots, A_n, \mathbf{0}), \mathbf{0})$
- Conceptually simpler problem: just one free variable
- Called "interpolation": $x\, w_1 \ldots w_k = u$ where $x : (B_1, \ldots, B_k, \mathbf{0})$, $w_i : B_i$, $u : \mathbf{0}$ in Inf.
- Solution t in Inf such that $tw_1 \ldots w_k \rightarrow_\beta^* u$

Matching is essentially monadic

- Given $v = u$ with $x_1 : A_1, \ldots, x_n : A_n$ in v
- there is the matching problem $x(\lambda x_1 \ldots x_n.v) = u$
  where $x : ((A_1, \ldots, A_n, \mathbf{0}), \mathbf{0})$

- Canonical solution $\lambda z.z\, t_1 \ldots t_n$ where t_is are closed
  Consequently $v\{t_1/x_1, \ldots, t_n/x_n\} \rightarrow^*_\beta u$
  So, $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ solves $v = u$
  Reduces matching to interpolation

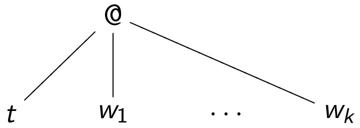
Matching is essentially monadic

- Given $v = u$ with $x_1 : A_1, \ldots, x_n : A_n$ in v
- there is the matching problem $x(\lambda x_1 \ldots x_n.v) = u$
  where $x : ((A_1, \ldots, A_n, \mathbf{0}), \mathbf{0})$

- Canonical solution $\lambda z.z\, t_1 \ldots t_n$ where t_is are closed
  Consequently $v\{t_1/x_1, \ldots, t_n/x_n\} \rightarrow^*_\beta u$
  So, $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ solves $v = u$
  Reduces matching to interpolation
- Restrict constants in solution terms to be those in u plus fresh $b : \mathbf{0}$ Restricts to finitely many constants

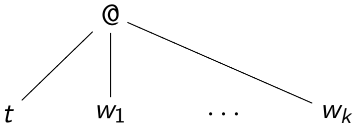
Interpolation trees

- t (of the right type and in Inf) a potential solution to $xw_1 \dots w_k = u$ (Assume u does not have bound variables)
- Interpolation tree



Interpolation trees

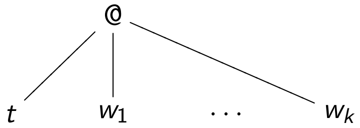
- t (of the right type and in Inf) a potential solution to $xw_1 \dots w_k = u$ (Assume u does not have bound variables)
- Interpolation tree



- Goal: understand the reduction of $tw_1 \dots w_k$ to normal form by only examining the interpolation tree. Three ways of doing this

Interpolation trees

- t (of the right type and in Inf) a potential solution to $xw_1 \ldots w_k = u$ (Assume u does not have bound variables)
- Interpolation tree

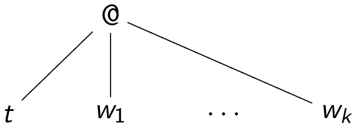


- Goal: understand the reduction of $tw_1 \ldots w_k$ to normal form by only examining the interpolation tree. Three ways of doing this

    1. Typing with intersection types

Interpolation trees

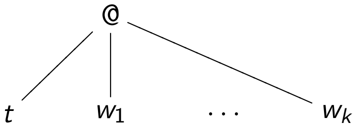
- t (of the right type and in Inf) a potential solution to $xw_1 \dots w_k = u$ (Assume u does not have bound variables)
- Interpolation tree



- Goal: understand the reduction of $tw_1 \dots w_k$ to normal form by only examining the interpolation tree. Three ways of doing this
  1. Typing with intersection types
  2. Tree automata

Interpolation trees

- t (of the right type and in Inf) a potential solution to $xw_1 \dots w_k = u$ (Assume u does not have bound variables)
- Interpolation tree

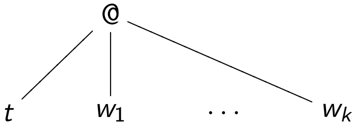


- Goal: understand the reduction of $tw_1 \dots w_k$ to normal form by only examining the interpolation tree. Three ways of doing this

  1. Typing with intersection types
  2. Tree automata
  3. Games/(Nonstandard) Automata

Intersection types

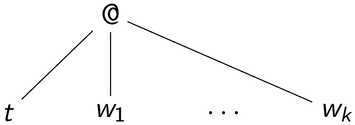
- t (of the right type and in Inf) a potential solution to $xw_1 \ldots w_k = u$ (Assume u does not have bound variables)
- Interpolation tree



- Goal: understand the reduction of $tw_1 \ldots w_k$ to normal form by only examining the interpolation tree

Intersection types

- t (of the right type and in Inf) a potential solution to $xw_1 \ldots w_k = u$ (Assume u does not have bound variables)
- Interpolation tree



- Goal: understand the reduction of $tw_1 \ldots w_k$ to normal form by only examining the interpolation tree
- Based on [Kobayashi 2009, Kobayashi and Ong 2009]; use intersection types

$$
\begin{array}{lcll}
\theta & := & r \mid \tau \to \theta & r \text{ subterm } u \\
\tau & := & \bigwedge_{i_1 \in I_1} \theta_{i_1} \wedge \ldots \wedge \bigwedge_{i_m \in I_m} \theta_{i_m} & I_j \text{ finite}
\end{array}
$$

Typing rules

$$\Gamma \vdash a : a \qquad \Gamma, x : \bigwedge_{i \in I} \theta_i \vdash x : \theta_j \quad j \in I$$

$$\frac{\Gamma \vdash t_1 : r_1 \qquad \Gamma \vdash t_m : r_m}{\Gamma \vdash f\ t_1 \ldots t_m : f r_1 \ldots r_m}$$

$$\frac{\Gamma \vdash t : \bigwedge_{i_1 \in I_1} \theta_{i_1} \wedge \ldots \wedge \bigwedge_{i_m \in I_m} \theta_{i_m} \rightarrow r \qquad \Gamma \vdash v_j : \theta_{i_j} \text{ for all } i_j \in I_j}{\Gamma \vdash t v_1 \ldots v_m : r}$$

$$\frac{\Gamma, x_1 : \bigwedge_{i_1 \in I_1} \theta_{i_1}, \ldots, x_m : \bigwedge_{i_m \in I_m} \theta_{i_m} \vdash t : r}{\Gamma \vdash \lambda x_1 \ldots x_m . t : \bigwedge_{i_1 \in J_1} \theta_{i_1} \wedge \ldots \wedge \bigwedge_{i_m \in J_m} \theta_{i_m} \rightarrow r}\ J_i \subseteq I_i$$

Matching and typing

- Assume $\Gamma \vdash w_j : \theta_i$ for all $i \in I_j$

Matching and typing

- Assume $\Gamma \vdash w_j : \theta_i$ for all $i \in I_j$
- So, $xw_1 \ldots w_k = u$ has canonical solution iff

Matching and typing

- Assume $\Gamma \vdash w_j : \theta_i$ for all $i \in I_j$
- So, $x w_1 \dots w_k = u$ has canonical solution iff
- some canonical term t has type $\bigwedge_{i \in I_1} \theta_i \wedge \dots \wedge \bigwedge_{i \in I_k} \theta_i \to u$

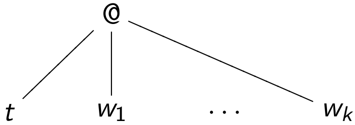
Matching and typing

- Assume $\Gamma \vdash w_j : \theta_i$ for all $i \in I_j$
- So, $xw_1 \dots w_k = u$ has canonical solution iff
- some canonical term t has type $\bigwedge_{i \in I_1} \theta_i \wedge \dots \wedge \bigwedge_{i \in I_k} \theta_i \rightarrow u$
- Reduces matching to inhabitation problem for intersection types

Matching and typing

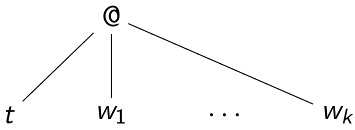
- Assume $\Gamma \vdash w_j : \theta_i$ for all $i \in I_j$
- So, $xw_1 \dots w_k = u$ has canonical solution iff
- some canonical term t has type $\bigwedge_{i \in I_1} \theta_i \wedge \dots \wedge \bigwedge_{i \in I_k} \theta_i \rightarrow u$
- Reduces matching to inhabitation problem for intersection types
- BUT, inhabitation is undecidable [Urzyczyn 1999]

Tree automata



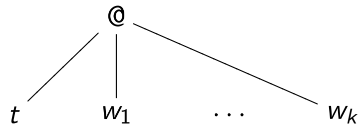
- Are solutions recognisable by an automaton ?

Tree automata



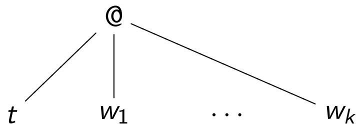
- Are solutions recognisable by an automaton ?
- Tree automaton
  Finite sets: states Q, alphabet Σ, final states $F \subseteq Q$,
  transitions Δ of form $sq_1 \dots q_k \Rightarrow q$,
  $k \geq 0$, $s \in \Sigma$, arity$(s) = k$ and $q, q_i \in Q$

Tree automata



- Are solutions recognisable by an automaton ?
- Tree automaton
  Finite sets: states Q, alphabet Σ, final states $F \subseteq Q$,
  transitions Δ of form $sq_1 \dots q_k \Rightarrow q$,
  $k \geq 0$, $s \in \Sigma$, $\text{arity}(s) = k$ and $q, q_i \in Q$
- Using transitions label tree bottom-up with states
  Automaton accepts tree iff root can be labelled with a final state

Tree automata



- Are solutions recognisable by an automaton ?
- Tree automaton
  Finite sets: states Q, alphabet Σ, final states $F \subseteq Q$,
  transitions Δ of form $sq_1 \ldots q_k \Rightarrow q$,
  $k \geq 0$, $s \in \Sigma$, $\text{arity}(s) = k$ and $q, q_i \in Q$
- Using transitions label tree bottom-up with states
  Automaton accepts tree iff root can be labelled with a final state
- Non-emptiness decidable; sets recognised are regular

Tree automata for 4th-order [Comon and Jurski 1997]

- 4th-order $xw_1 \dots w_k = u$ and C constants in u plus b
- For finite alphabet Σ consider a potential solution term $t = \lambda x_1 \dots x_k.s$

$$s ::= z_j^n \mid x_i v_1 \dots v_m \mid f s_1 \dots s_m \qquad f \in C,\ m \geq 0$$
$$v ::= s \mid \lambda z_1^n \dots z_m^n.s \qquad m \geq 0 \quad \text{NOTE: } z_i^n : \mathbf{0}$$

Tree automata for 4th-order [Comon and Jurski 1997]

- 4th-order $xw_1 \dots w_k = u$ and C constants in u plus b
- For finite alphabet Σ consider a potential solution term $t = \lambda x_1 \dots x_k.s$

$$s ::= z_j^n \mid x_i v_1 \dots v_m \mid f s_1 \dots s_m \qquad f \in C, \;\; m \geq 0$$
$$v ::= s \mid \lambda z_1^n \dots z_m^n.s \qquad m \geq 0 \quad \text{NOTE: } z_i^n : \mathbf{0}$$

- Node n of t is labelled with a variable/constant; let t^n be subterm rooted at n

Tree automata for 4th-order [Comon and Jurski 1997]

- 4th-order $xw_1 \dots w_k = u$ and C constants in u plus b
- For finite alphabet Σ consider a potential solution term $t = \lambda x_1 \dots x_k.s$

$$
\begin{array}{ll}
s ::= z_j^n \mid x_i v_1 \dots v_m \mid f s_1 \dots s_m & f \in C,\ m \geq 0 \\
v ::= s \mid \lambda z_1^n \dots z_m^n.s \quad m \geq 0 & \text{NOTE: } z_i^n : \mathbf{0}
\end{array}
$$

- Node n of t is labelled with a variable/constant; let t^n be subterm rooted at n
- n is inaccessible
  if it does not contribute to normal form of $tw_1 \dots w_k$
  (Therefore, can replace t^n with $b : \mathbf{0}$; preserve normal form)

Tree automata for 4th-order [Comon and Jurski 1997]

- 4th-order $xw_1 \dots w_k = u$ and C constants in u plus b
- For finite alphabet Σ consider a potential solution term $t = \lambda x_1 \dots x_k.s$

$$s ::= z_j^n \mid x_i v_1 \dots v_m \mid f s_1 \dots s_m \qquad f \in C,\ m \geq 0$$
$$v ::= s \mid \lambda z_1^n \dots z_m^n.s \qquad m \geq 0 \quad \text{NOTE: } z_i^n : \mathbf{0}$$

- Node n of t is labelled with a variable/constant; let t^n be subterm rooted at n
- n is inaccessible
  if it does not contribute to normal form of $tw_1 \dots w_k$
  (Therefore, can replace t^n with $b : \mathbf{0}$; preserve normal form)
- n is accessible; look at evaluation of t^n
  which is normal form of $t^n\{w_1/x_1, \dots, w_k/x_k\}$
  this is a subterm of u when its subterms can be replaced by leaf variables z_j^i

Tree automata for 4th-order [Comon and Jurski 1997]

- 4th-order $xw_1 \dots w_k = u$ and C constants in u plus b
- For finite alphabet Σ consider a potential solution term $t = \lambda x_1 \dots x_k.s$

$$s ::= z_j^n \mid x_i v_1 \dots v_m \mid f s_1 \dots s_m \qquad f \in C,\ m \geq 0$$
$$v ::= s \mid \lambda z_1^n \dots z_m^n.s \qquad m \geq 0 \quad \text{NOTE: } z_i^n : \mathbf{0}$$

- Node n of t is labelled with a variable/constant; let t^n be subterm rooted at n
- n is inaccessible
  if it does not contribute to normal form of $tw_1 \dots w_k$
  (Therefore, can replace t^n with $b : \mathbf{0}$; preserve normal form)
- n is accessible; look at evaluation of t^n
  which is normal form of $t^n\{w_1/x_1, \dots, w_k/x_k\}$
  this is a subterm of u when its subterms can be replaced by leaf variables z_j^i
- Guarantees finite alphabet for accessible nodes: reuse variables

Tree automata for 4th-order [Comon and Jurski 1997]

- Finite states:
  inaccessible nodes: $-$ for "doesn't matter"
  accessible nodes: U is subterms of u closed under replacement of subterms with leaves labelled with finitely many different variables $z^i_j : \mathbf{0}$

$$\{[e], [\lambda z^n_1 \dots z^n_m . e], [\lambda x_1 \dots x_k . u] \mid e \in U \cup \{-\}\}$$

Tree automata for 4th-order [Comon and Jurski 1997]

- Finite states:
  inaccessible nodes: $-$ for "doesn't matter"
  accessible nodes: U is subterms of u closed under replacement of subterms with leaves labelled with finitely many different variables $z^i_j : \mathbf{0}$

$$\{[e], [\lambda z^n_1 \ldots z^n_m.e], [\lambda x_1 \ldots x_k.u] \mid e \in U \cup \{-\}\}$$

- Final states $\{[\lambda x_1 \ldots x_k.u]\}$

Tree automata for 4th-order [Comon and Jurski 1997]

- Finite states:
  inaccessible nodes: $-$ for "doesn't matter"
  accessible nodes: U is subterms of u closed under replacement of subterms with leaves labelled with finitely many different variables $z_j^i : \mathbf{0}$

$$\{[e], [\lambda z_1^n \dots z_m^n.e], [\lambda x_1 \dots x_k.u] \mid e \in U \cup \{-\}\}$$

- Final states $\{[\lambda x_1 \dots x_k.u]\}$
- Finite transitions: such as

$$\begin{array}{ll} z \Rightarrow [z] \quad z \Rightarrow [-] & a \Rightarrow [a] \quad c \Rightarrow [-] \\ \lambda x_1[ga] \Rightarrow [\lambda x_1.ga] & x_1[-][gz] \Rightarrow [gz] \\ \dots & \end{array}$$

Tree automata for 4th-order [Comon and Jurski 1997]

- Finite states:
  inaccessible nodes: $-$ for "doesn't matter"
  accessible nodes: U is subterms of u closed under replacement of subterms with leaves labelled with finitely many different variables $z^i_j : \mathbf{0}$

$$\{[e], [\lambda z^n_1 \dots z^n_m.e], [\lambda x_1 \dots x_k.u] \mid e \in U \cup \{-\}\}$$

- Final states $\{[\lambda x_1 \dots x_k.u]\}$
- Finite transitions: such as

$$\begin{array}{llll} z \Rightarrow [z] & z \Rightarrow [-] & a \Rightarrow [a] & c \Rightarrow [-] \\ \lambda x_1[ga] \Rightarrow [\lambda x_1.ga] & & x_1[-][gz] \Rightarrow [gz] & \\ \dots & & & \end{array}$$

- Theorem The tree automaton associated with a 4th-order $xw_1 \dots w_k = u$ accepts t iff t solves $xw_1 \dots w_k = u$

Tree automata for 5th-order: PROBLEMS

- Finite alphabet ?

  can stack 2nd-order variables

  $\lambda xy.x(\lambda z_1.x(\lambda z_2 \ldots (\lambda z_n.z_n(z_{n-1}(\ldots z_1(y)) \ldots)) \ldots))$

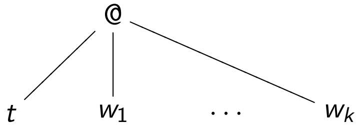
  Need an infinite alphabet

Tree automata for 5th-order: PROBLEMS

- Finite alphabet ?
  can stack 2nd-order variables
  $\lambda xy.x(\lambda z_1.x(\lambda z_2 \ldots (\lambda z_n.z_n(z_{n-1}(\ldots z_1(y))\ldots))\ldots))$
  Need an infinite alphabet
- Finite number of states ?
  stack same 2nd-order variable
  $z_n(z_n(\ldots(z_n a)\ldots))$
  Number of evaluations can be infinite

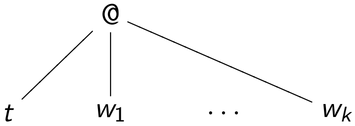
Interpolation trees

- t (of the right type and in Inf) a potential solution to $xw_1 \ldots w_k = u$ (Assume u does not have bound variables)
- Interpolation tree



Interpolation trees

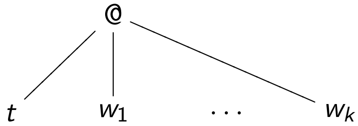
- t (of the right type and in Inf) a potential solution to $xw_1 \ldots w_k = u$ (Assume u does not have bound variables)
- Interpolation tree



- Special representation of term trees

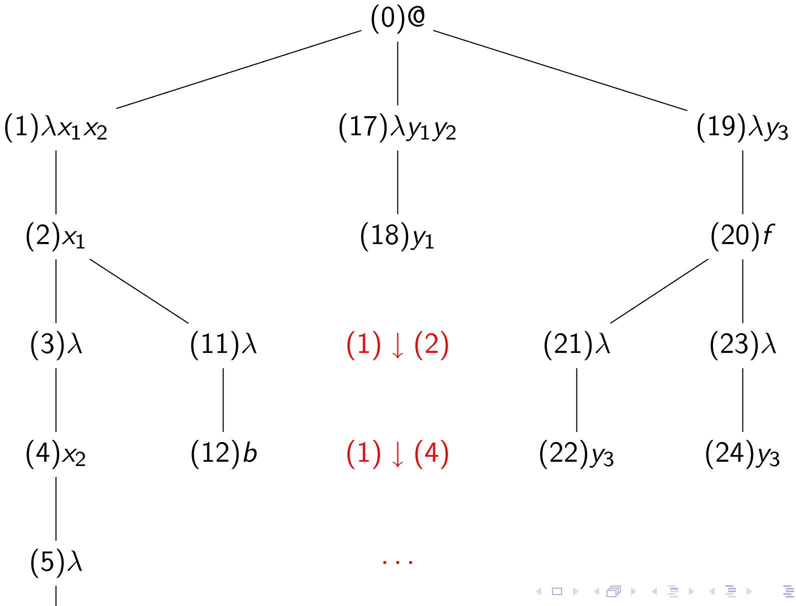
Interpolation trees

- t (of the right type and in Inf) a potential solution to $xw_1 \ldots w_k = u$ (Assume u does not have bound variables)
- Interpolation tree



- Special representation of term trees
- $t, w_1, \ldots, w_k$ binding trees with dummy lambdas and binding relation $\downarrow$ between nodes
- $n \downarrow m$ if n labelled $\lambda \overline{y}$ binds y_j, label at m

Example: $x(\lambda y_1 y_2 . y_1)(\lambda y_3 . f y_3 y_3) = faa$



Tree automata for 5th-order

- Finite alphabet ? can stack 2nd-order variables
  $\lambda xy.x(\lambda z_1.x(\lambda z_2 \ldots (\lambda z_n.z_n(z_{n-1}(\ldots z_1(y)) \ldots)) \ldots))$
  Need an infinite alphabet
- Finite number of states ? stack same 2nd-order variable
  $z_n(z_n(\ldots(z_n a)\ldots))$
  Number of evaluations can be infinite

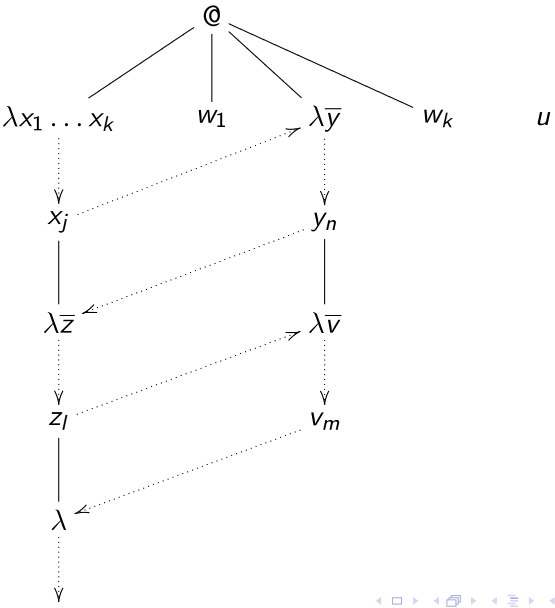
Tree automata for 5th-order

- Finite alphabet ? can stack 2nd-order variables
  $\lambda xy.x(\lambda z_1.x(\lambda z_2 \ldots (\lambda z_n.z_n(z_{n-1}(\ldots z_1(y))\ldots))\ldots))$
  Need an infinite alphabet
- Finite number of states ? stack same 2nd-order variable
  $z_n(z_n(\ldots(z_n a)\ldots))$
  Number of evaluations can be infinite
- Overcome the first problem: employ binding trees
  $\lambda xy.x(\lambda z.x(\lambda z \ldots x(\lambda z.z(\lambda.z(\ldots \lambda.z(\lambda.y))\ldots))\ldots))$
  $\downarrow$ from the first node labelled λz to the last node labelled z, and so on
  Fact For all A and finite C, there is a finite Σ such that every $t \in T_A(C)$ up to α-equivalence is a binding Σ-tree.

Tree automata for 5th-order

- Finite alphabet ? can stack 2nd-order variables
  $\lambda xy.x(\lambda z_1.x(\lambda z_2 \ldots (\lambda z_n.z_n(z_{n-1}(\ldots z_1(y))\ldots))\ldots))$
  Need an infinite alphabet
- Finite number of states ? stack same 2nd-order variable
  $z_n(z_n(\ldots(z_n a)\ldots))$
  Number of evaluations can be infinite
- Overcome the first problem: employ binding trees
  $\lambda xy.x(\lambda z.x(\lambda z \ldots x(\lambda z.z(\lambda.z(\ldots \lambda.z(\lambda.y))\ldots))\ldots))$
  $\downarrow$ from the first node labelled λz to the last node labelled z, and so on
  Fact For all A and finite C, there is a finite Σ such that every $t \in T_A(C)$ up to α-equivalence is a binding Σ-tree.
- Second problem ? need finer analysis than evaluation of an accessible node

Games: automaton moving around interpolation tree



Game/automaton $G(t, E)$

- Play: sequence $n_1 q_1 \theta_1, \dots, n_l q_l \theta_l$
  - n_i is a node of the interpolation tree,
  - q_i is a state $[u']$ where u' subterm of u or final
  - θ_i is look-up table: tells where to jump when at a variable

Game/automaton G(t, E)

- Play: sequence $n_1 q_1 \theta_1, \ldots, n_l q_l \theta_l$
  - n_i is a node of the interpolation tree,
  - q_i is a state $[u']$ where u' subterm of u or final
  - θ_i is look-up table: tells where to jump when at a variable
- Initial position at @, $q_1 = [\,u\,]$ and θ_1 is empty
- $\forall$ loses the play if the final state $q_l = [\,\exists\,]$

Game/automaton $G(t, E)$

- Play: sequence $n_1 q_1 \theta_1, \ldots, n_l q_l \theta_l$
  - n_i is a node of the interpolation tree,
  - q_i is a state $[u']$ where u' subterm of u or final
  - θ_i is look-up table: tells where to jump when at a variable
- Initial position at @, $q_1 = [\,u\,]$ and θ_1 is empty
- $\forall$ loses the play if the final state $q_l = [\,\exists\,]$
- Current position is $n[r]\theta$; next position
  - @ then $n1[r]\theta'$ where $\theta' = \theta\{((n2, \ldots, n(k+1)), \theta)/n1\}$
  - $\lambda\overline{y}$ then $n1[r]\theta$

Game/automaton $G(t, E)$

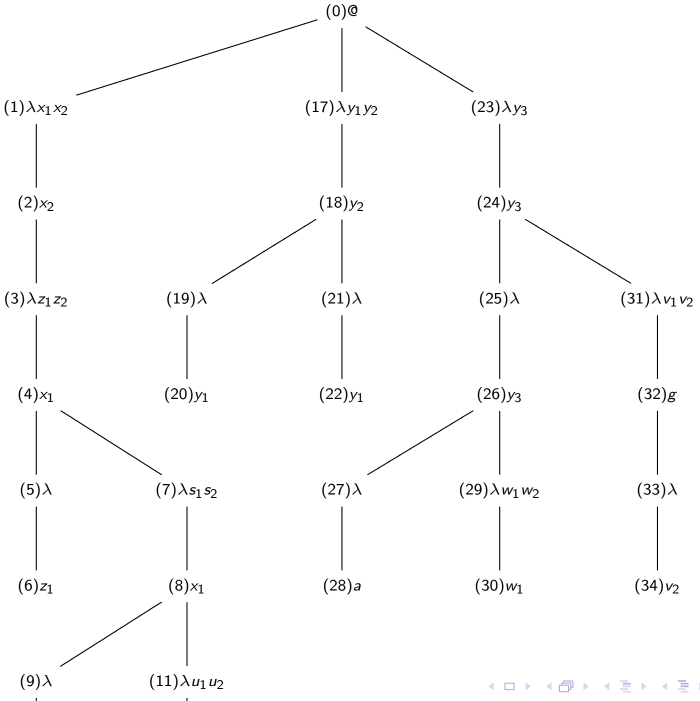
- Play: sequence $n_1 q_1 \theta_1, \ldots, n_l q_l \theta_l$
  - n_i is a node of the interpolation tree,
  - q_i is a state $[u']$ where u' subterm of u or final
  - θ_i is look-up table: tells where to jump when at a variable
- Initial position at @, $q_1 = [\, u\,]$ and θ_1 is empty
- $\forall$ loses the play if the final state $q_l = [\,\exists\,]$
- Current position is $n[r]\theta$; next position
  - @ then $n1[r]\theta'$ where $\theta' = \theta\{((n2, \ldots, n(k+1)), \theta)/n1\}$
  - $\lambda\overline{y}$ then $n1[r]\theta$
  - $a : \mathbf{0}$ if $r = a$ then $n[\,\exists\,]\theta$ else $n[\forall\,]\theta$
  - $f : (B_1, \ldots, B_p, \mathbf{0})$ if $r = fr_1 \ldots r_p$ then $\forall$ chooses $j \in \{1, \ldots, p\}$ and $nj\,[r_j]\theta$ else $n[\forall\,]\theta$

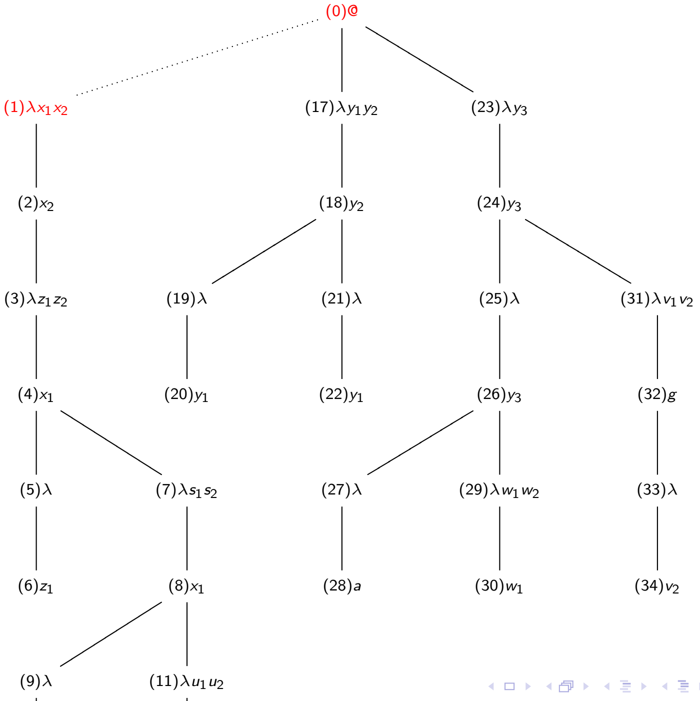
Game/automaton $G(t, E)$

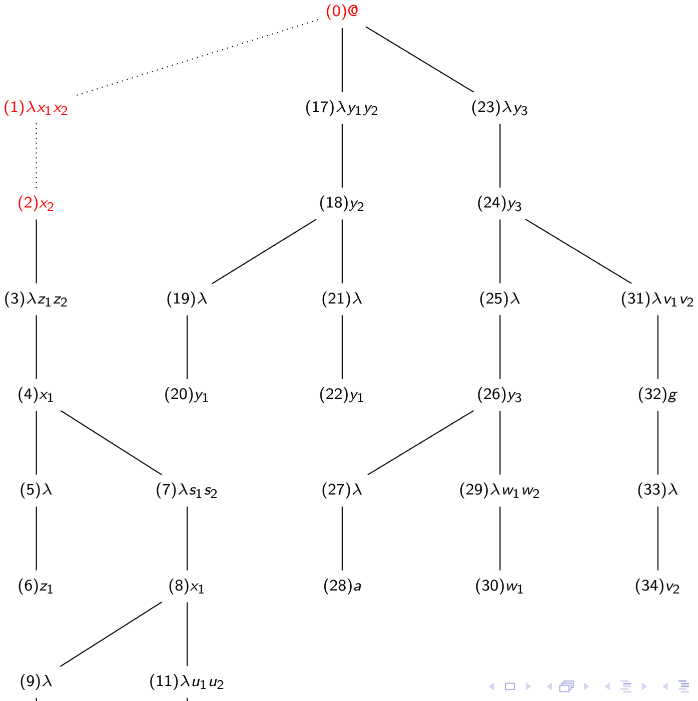
- Play: sequence $n_1 q_1 \theta_1, \ldots, n_l q_l \theta_l$
  - n_i is a node of the interpolation tree,
  - q_i is a state $[u']$ where u' subterm of u or final
  - θ_i is look-up table: tells where to jump when at a variable
- Initial position at @, $q_1 = [\,u\,]$ and θ_1 is empty
- $\forall$ loses the play if the final state $q_l = [\,\exists\,]$
- Current position is $n[r]\theta$; next position
  - @ then $n1[r]\theta'$ where $\theta' = \theta\{((n2, \ldots, n(k+1)), \theta)/n1\}$
  - $\lambda\overline{y}$ then $n1[r]\theta$
  - $a : \mathbf{0}$ if $r = a$ then $n[\,\exists\,]\theta$ else $n[\,\forall\,]\theta$
  - $f : (B_1, \ldots, B_p, \mathbf{0})$ if $r = fr_1 \ldots r_p$ then $\forall$ chooses $j \in \{1, \ldots, p\}$ and $nj\,[r_j]\theta$ else $n[\,\forall\,]\theta$
  - $y_j : \mathbf{0}$ if $m \downarrow n$ and $\theta(m) = ((m_1, \ldots, m_l), \theta')$ then $m_j\,[r]\theta'$
  - $y_j : (B_1, \ldots, B_p, \mathbf{0})$ if $m \downarrow n$ and $\theta(m) = ((m_1, \ldots, m_l), \theta')$ then $m_j\,[r]\theta''$ where $\theta'' = \theta'\{((n1, \ldots, np), \theta)/m_j\}$

Game/automaton $G(t, E)$

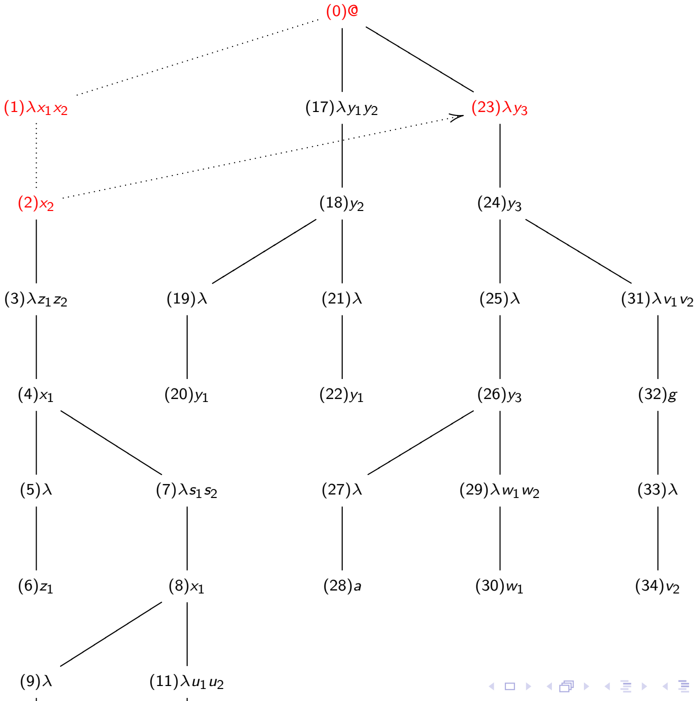
- Play: sequence $n_1 q_1 \theta_1, \ldots, n_l q_l \theta_l$
  - n_i is a node of the interpolation tree,
  - q_i is a state $[u']$ where u' subterm of u or final
  - θ_i is look-up table: tells where to jump when at a variable
- Initial position at @, $q_1 = [\,u\,]$ and θ_1 is empty
- $\forall$ loses the play if the final state $q_l = [\,\exists\,]$
- Current position is $n[r]\theta$; next position
  - @ then $n1[r]\theta'$ where $\theta' = \theta\{((n2, \ldots, n(k+1)), \theta)/n1\}$
  - $\lambda\overline{y}$ then $n1[r]\theta$
  - $a : \mathbf{0}$ if $r = a$ then $n[\,\exists\,]\theta$ else $n[\,\forall\,]\theta$
  - $f : (B_1, \ldots, B_p, \mathbf{0})$ if $r = fr_1 \ldots r_p$ then $\forall$ chooses $j \in \{1, \ldots, p\}$ and $nj\,[r_j]\theta$ else $n[\,\forall\,]\theta$
  - $y_j : \mathbf{0}$ if $m \downarrow n$ and $\theta(m) = ((m_1, \ldots, m_l), \theta')$ then $m_j\,[r]\theta'$
  - $y_j : (B_1, \ldots, B_p, \mathbf{0})$ if $m \downarrow n$ and $\theta(m) = ((m_1, \ldots, m_l), \theta')$ then $m_j\,[r]\theta''$ where $\theta'' = \theta'\{((n1, \ldots, np), \theta)/m_j\}$
- Player $\forall$ loses every play in $G(t, E)$ iff t solves E

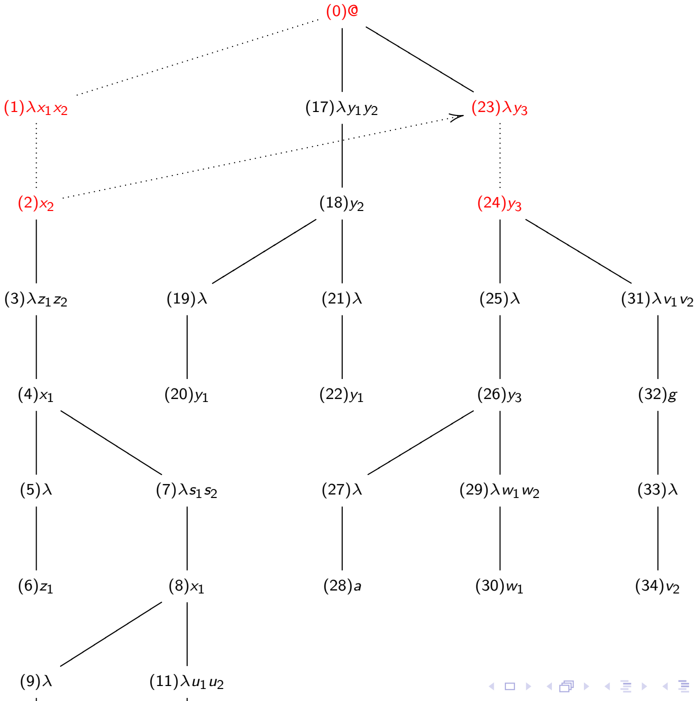

(0)@
(1)$\lambda x_1 x_2$
(17)$\lambda y_1 y_2$
(23)λy_3
(2)x_2
(18)y_2
(24)y_3
(3)$\lambda z_1 z_2$
(19)λ
(21)λ
(25)λ
(31)$\lambda v_1 v_2$
(4)x_1
(20)y_1
(22)y_1
(26)y_3
(32)g
(5)λ
(7)$\lambda s_1 s_2$
(27)λ
(29)$\lambda w_1 w_2$
(33)λ
(6)z_1
(8)x_1
(28)a
(30)w_1
(34)v_2
(9)λ
(11)$\lambda u_1 u_2$

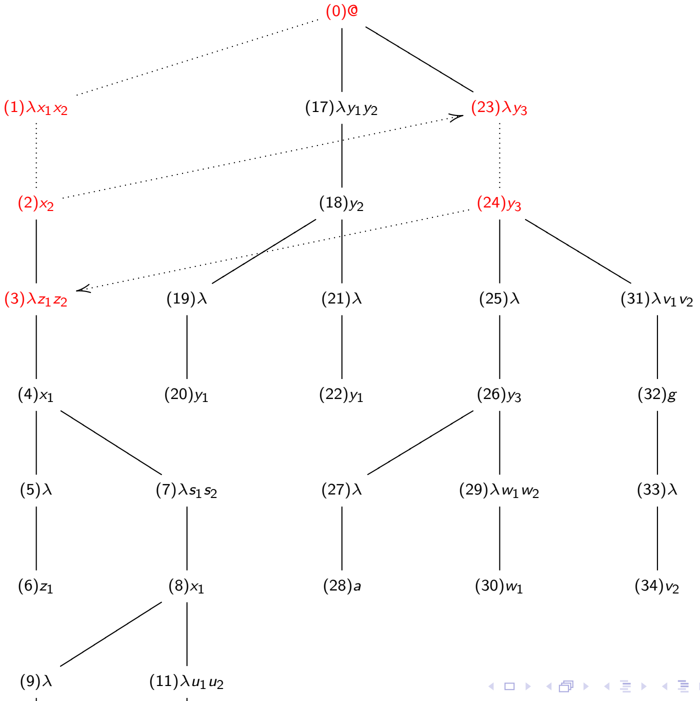

(0)@
(1)$\lambda x_1 x_2$
(17)$\lambda y_1 y_2$
(23)λy_3
(2)x_2
(18)y_2
(24)y_3
(3)$\lambda z_1 z_2$
(19)λ
(21)λ
(25)λ
(31)$\lambda v_1 v_2$
(4)x_1
(20)y_1
(22)y_1
(26)y_3
(32)g
(5)λ
(7)$\lambda s_1 s_2$
(27)λ
(29)$\lambda w_1 w_2$
(33)λ
(6)z_1
(8)x_1
(28)a
(30)w_1
(34)v_2
(9)λ
(11)$\lambda u_1 u_2$

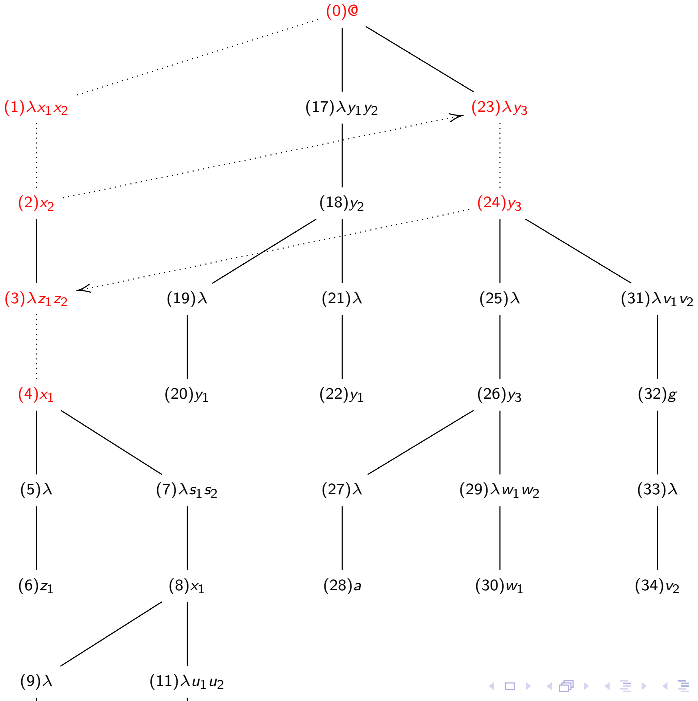


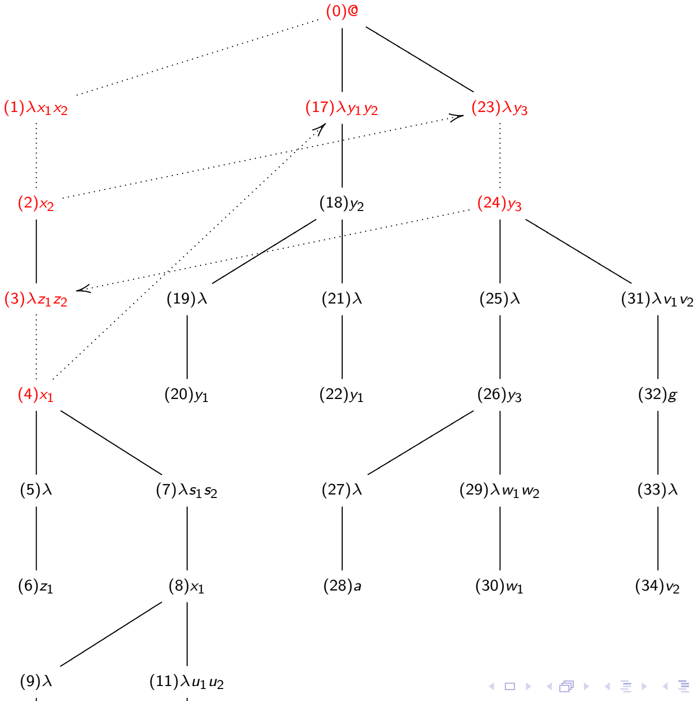
(0)@
(1)λx1x2
(17)λy1y2
(23)λy3
(2)x2
(18)y2
(24)y3
(3)λz1z2
(19)λ
(21)λ
(25)λ
(31)λv1v2
(4)x1
(20)y1
(22)y1
(26)y3
(32)g
(5)λ
(7)λs1s2
(27)λ
(29)λw1w2
(33)λ
(6)z1
(8)x1
(28)a
(30)w1
(34)v2
(9)λ
(11)λu1u2

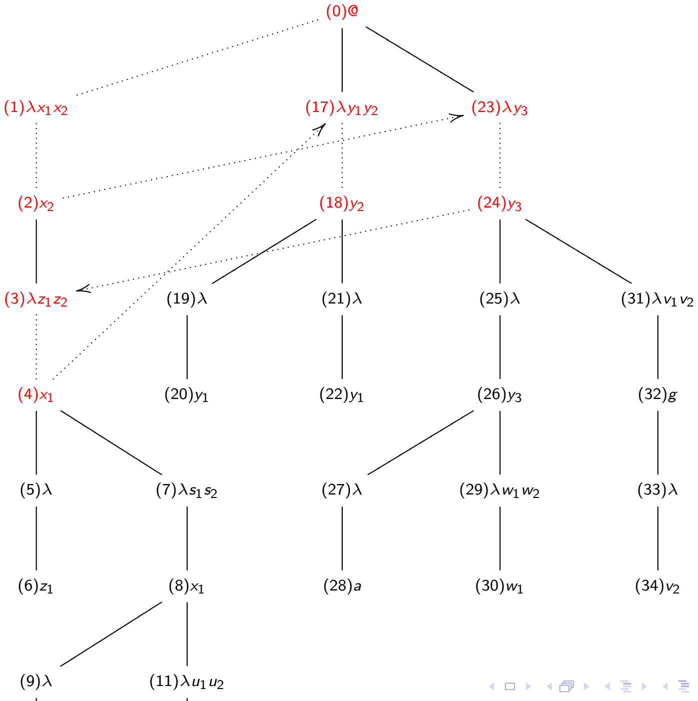

(0)@
(1)λx1 x2
(17)λy1 y2
(23)λy3
(2)x2
(18)y2
(24)y3
(3)λz1 z2
(19)λ
(21)λ
(25)λ
(31)λv1 v2
(4)x1
(20)y1
(22)y1
(26)y3
(32)g
(5)λ
(7)λs1 s2
(27)λ
(29)λw1 w2
(33)λ
(6)z1
(8)x1
(28)a
(30)w1
(34)v2
(9)λ
(11)λu1 u2

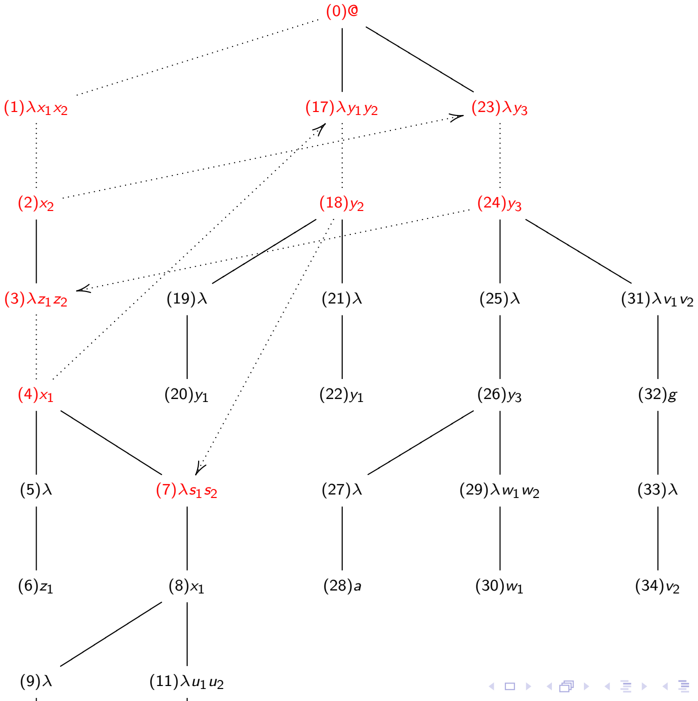

(0)@
(1)λx1x2
(17)λy1y2
(23)λy3
(2)x2
(18)y2
(24)y3
(3)λz1z2
(19)λ
(21)λ
(25)λ
(31)λv1v2
(4)x1
(20)y1
(22)y1
(26)y3
(32)g
(5)λ
(7)λs1s2
(27)λ
(29)λw1w2
(33)λ
(6)z1
(8)x1
(28)a
(30)w1
(34)v2
(9)λ
(11)λu1u2

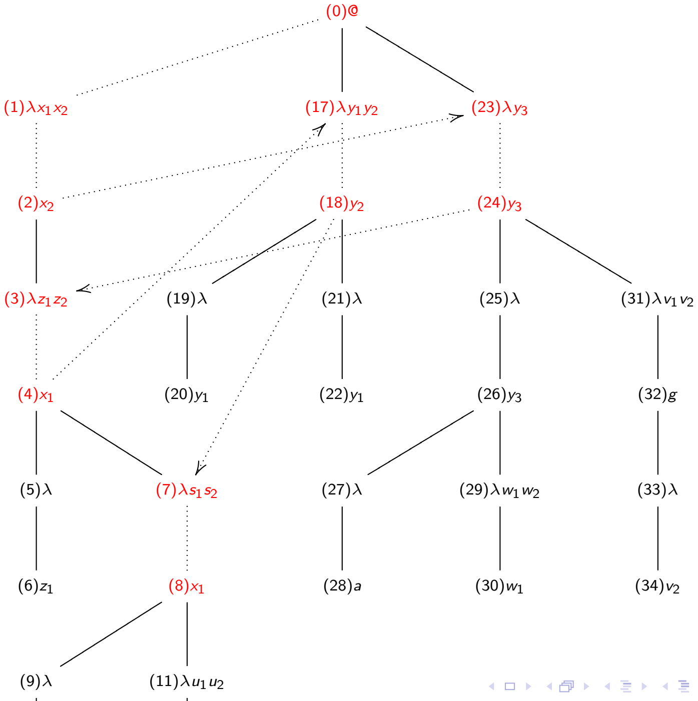

(0)@
(1)λx1x2
(17)λy1y2
(23)λy3
(2)x2
(18)y2
(24)y3
(3)λz1z2
(19)λ
(21)λ
(25)λ
(31)λv1v2
(4)x1
(20)y1
(22)y1
(26)y3
(32)g
(5)λ
(7)λs1s2
(27)λ
(29)λw1w2
(33)λ
(6)z1
(8)x1
(28)a
(30)w1
(34)v2
(9)λ
(11)λu1u2

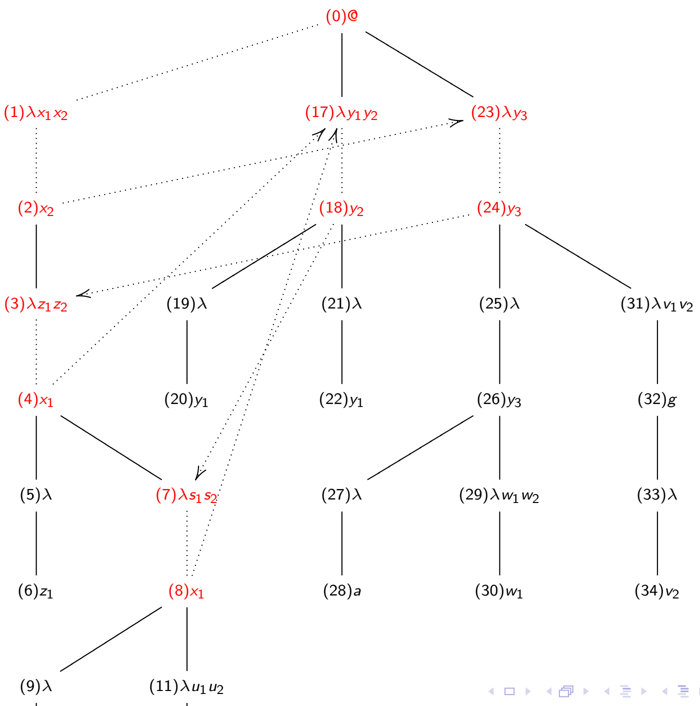

(0)@
(1)λx1 x2
(17)λy1 y2
(23)λy3
(2)x2
(18)y2
(24)y3
(3)λz1 z2
(19)λ
(21)λ
(25)λ
(31)λv1 v2
(4)x1
(20)y1
(22)y1
(26)y3
(32)g
(5)λ
(7)λs1 s2
(27)λ
(29)λw1 w2
(33)λ
(6)z1
(8)x1
(28)a
(30)w1
(34)v2
(9)λ
(11)λu1 u2

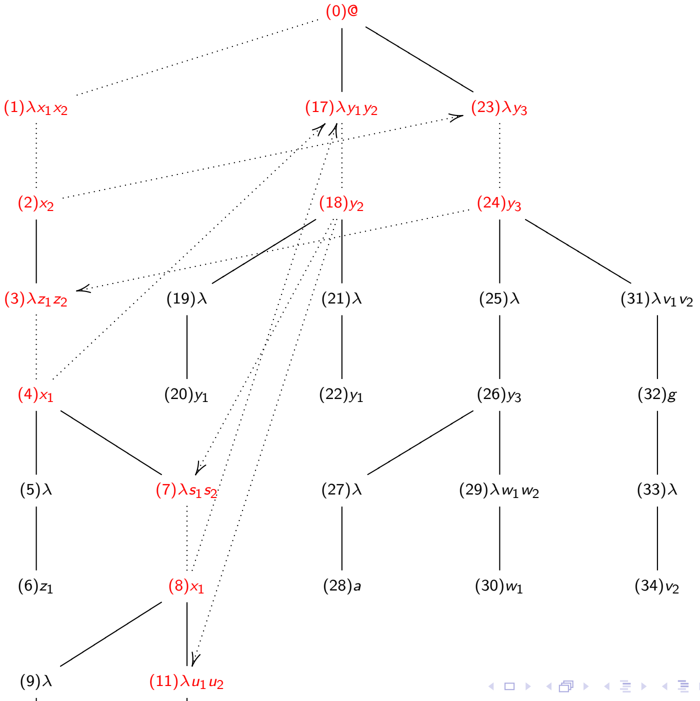

(0)@
(1)λx1x2
(17)λy1y2
(23)λy3
(2)x2
(18)y2
(24)y3
(3)λz1z2
(19)λ
(21)λ
(25)λ
(31)λv1v2
(4)x1
(20)y1
(22)y1
(26)y3
(32)g
(5)λ
(7)λs1s2
(27)λ
(29)λw1w2
(33)λ
(6)z1
(8)x1
(28)a
(30)w1
(34)v2
(9)λ
(11)λu1u2


(0)@
(1)$\lambda x_1 x_2$
(17)$\lambda y_1 y_2$
(23)λy_3
(2)x_2
(18)y_2
(24)y_3
(3)$\lambda z_1 z_2$
(19)λ
(21)λ
(25)λ
(31)$\lambda v_1 v_2$
(4)x_1
(20)y_1
(22)y_1
(26)y_3
(32)g
(5)λ
(7)$\lambda s_1 s_2$
(27)λ
(29)$\lambda w_1 w_2$
(33)λ
(6)z_1
(8)x_1
(28)a
(30)w_1
(34)v_2
(9)λ
(11)$\lambda u_1 u_2$


(0)@
(1)λx1x2
(17)λy1y2
(23)λy3
(2)x2
(18)y2
(24)y3
(3)λz1z2
(19)λ
(21)λ
(25)λ
(31)λv1v2
(4)x1
(20)y1
(22)y1
(26)y3
(32)g
(5)λ
(7)λs1s2
(27)λ
(29)λw1w2
(33)λ
(6)z1
(8)x1
(28)a
(30)w1
(34)v2
(9)λ
(11)λu1u2


(0)@
(1)$\lambda x_1 x_2$
(17)$\lambda y_1 y_2$
(23)λy_3
(2)x_2
(18)y_2
(24)y_3
(3)$\lambda z_1 z_2$
(19)λ
(21)λ
(25)λ
(31)$\lambda v_1 v_2$
(4)x_1
(20)y_1
(22)y_1
(26)y_3
(32)g
(5)λ
(7)$\lambda s_1 s_2$
(27)λ
(29)$\lambda w_1 w_2$
(33)λ
(6)z_1
(8)x_1
(28)a
(30)w_1
(34)v_2
(9)λ
(11)$\lambda u_1 u_2$


(0)@
(1)$\lambda x_1 x_2$
(17)$\lambda y_1 y_2$
(23)λy_3
(2)x_2
(18)y_2
(24)y_3
(3)$\lambda z_1 z_2$
(19)λ
(21)λ
(25)λ
(31)$\lambda v_1 v_2$
(4)x_1
(20)y_1
(22)y_1
(26)y_3
(32)g
(5)λ
(7)$\lambda s_1 s_2$
(27)λ
(29)$\lambda w_1 w_2$
(33)λ
(6)z_1
(8)x_1
(28)a
(30)w_1
(34)v_2
(9)λ
(11)$\lambda u_1 u_2$


(0)@
(1)$\lambda x_1 x_2$
(17)$\lambda y_1 y_2$
(23)λy_3
(2)x_2
(18)y_2
(24)y_3
(3)$\lambda z_1 z_2$
(19)λ
(21)λ
(25)λ
(31)$\lambda v_1 v_2$
(4)x_1
(20)y_1
(22)y_1
(26)y_3
(32)g
(5)λ
(7)$\lambda s_1 s_2$
(27)λ
(29)$\lambda w_1 w_2$
(33)λ
(6)z_1
(8)x_1
(28)a
(30)w_1
(34)v_2
(9)λ
(11)$\lambda u_1 u_2$

Application of games to define tree automata

- Let E be interpolation equation of arbitrary order
- Two problems extending Comon and Jurski's result
  1. Ensuring finitely many states in automaton
  2. Ensuring finite alphabet in automaton

Application of games to define tree automata

- Let E be interpolation equation of arbitrary order
- Two problems extending Comon and Jurski's result
  1. Ensuring finitely many states in automaton
  2. Ensuring finite alphabet in automaton
- Instead of using evaluation of a node for states use (version of) variable profiles from [Ong 2006]
- Variable profile: abstraction from sequences of positions from a node $\{(x_2, ga, \{(y_3, ga, \{(z_2, ga, \{(v_2, a, \emptyset)\})\})\})\}$

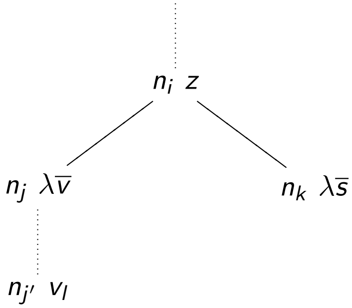
Application of games to define tree automata

- Let E be interpolation equation of arbitrary order
- Two problems extending Comon and Jurski's result
  1. Ensuring finitely many states in automaton
  2. Ensuring finite alphabet in automaton
- Instead of using evaluation of a node for states use (version of) variable profiles from [Ong 2006]
- Variable profile: abstraction from sequences of positions from a node $\{(x_2, ga, \{(y_3, ga, \{(z_2, ga, \{(v_2, a, \emptyset)\})\})\})\}$
- Theorem The set of solutions of E built out of a fixed finite alphabet of variables is recognised by a tree automaton [Stirling 2007]

Application of games to define tree automata

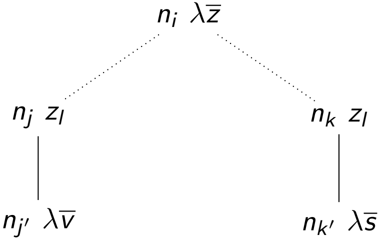
- Let E be interpolation equation of arbitrary order
- Two problems extending Comon and Jurski's result
  1. Ensuring finitely many states in automaton
  2. Ensuring finite alphabet in automaton
- Instead of using evaluation of a node for states use (version of) variable profiles from [Ong 2006]
- Variable profile: abstraction from sequences of positions from a node $\{(x_2, ga, \{(y_3, ga, \{(z_2, ga, \{(v_2, a, \emptyset)\})\})\})\}$
- Theorem The set of solutions of E built out of a fixed finite alphabet of variables is recognised by a tree automaton [Stirling 2007]
- Theorem The set of solutions of E is recognised by a special alternating binding tree automaton BUT non-emptiness of such automata undecidable [Stirling 2009; Ong, Tzevelekos 2009]

Uniformity properties of game playing I



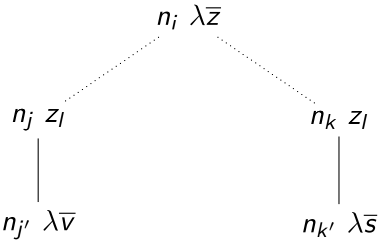
- If play is at n_i then at n_j and then later at n_k then inbetween there must have been a position at an $n_{j'}$ bound by n_j

Uniformity properties of game playing II



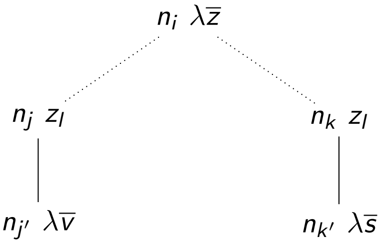
- If play is at n_i then at n_j and then later at $n_{j'}$; then at n_k then there must be a corresponding sequence to that between n_j and $n_{j'}$ between n_k and $n_{k'}$ unless there is a different $\forall$ choice

Uniformity properties of game playing II



- If play is at n_i then at n_j and then later at $n_{j'}$; then at n_k then there must be a corresponding sequence to that between n_j and $n_{j'}$ between n_k and $n_{k'}$ unless there is a different $\forall$ choice
- Especially relevant if n_k is somewhere below n_j ("embedded")

Uniformity properties of game playing II



- If play is at n_i then at n_j and then later at $n_{j'}$; then at n_k then there must be a corresponding sequence to that between n_j and $n_{j'}$ between n_k and $n_{k'}$ unless there is a different $\forall$ choice
- Especially relevant if n_k is somewhere below n_j ("embedded")
- Property not enforced in a tree automaton

Application of games to decidability of matching

- Combinatorial argument based on uniformities of play

Application of games to decidability of matching

- Combinatorial argument based on uniformities of play
- Two steps in proof assuming an arbitrary solution term

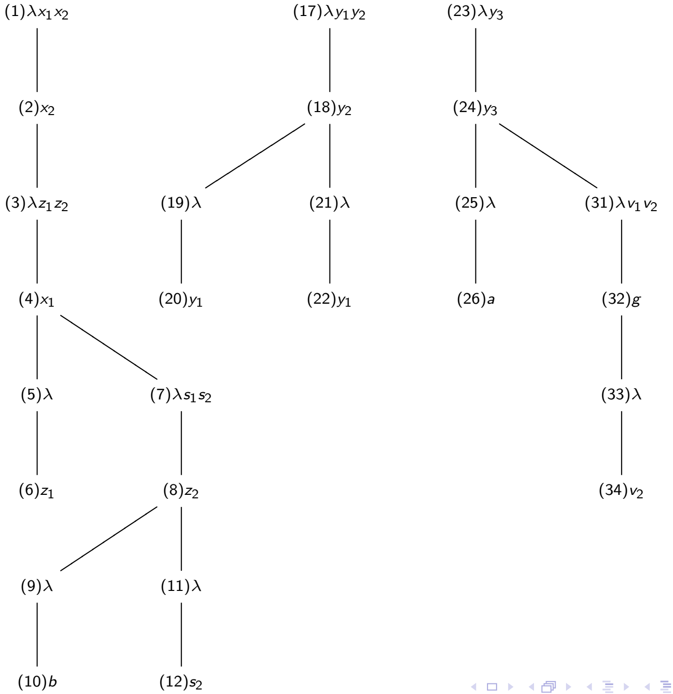
Application of games to decidability of matching

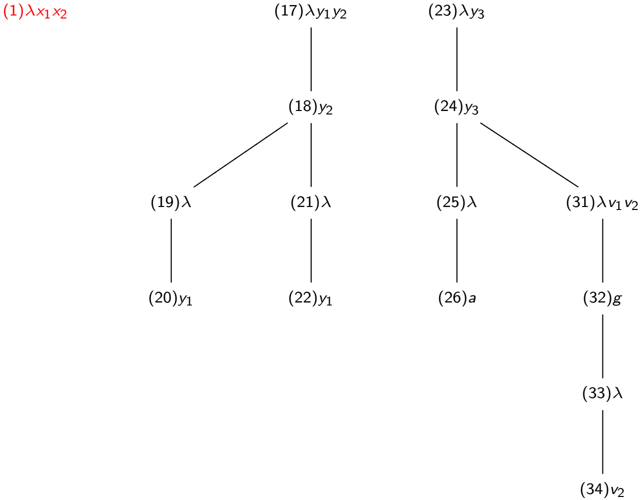
- Combinatorial argument based on uniformities of play
- Two steps in proof assuming an arbitrary solution term
  1. partition each play

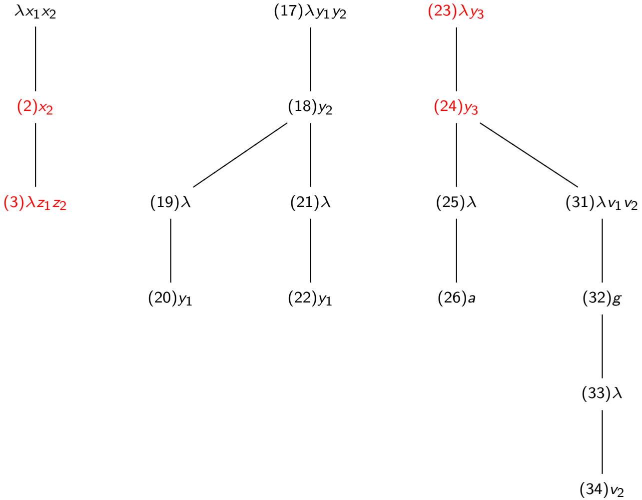
Application of games to decidability of matching

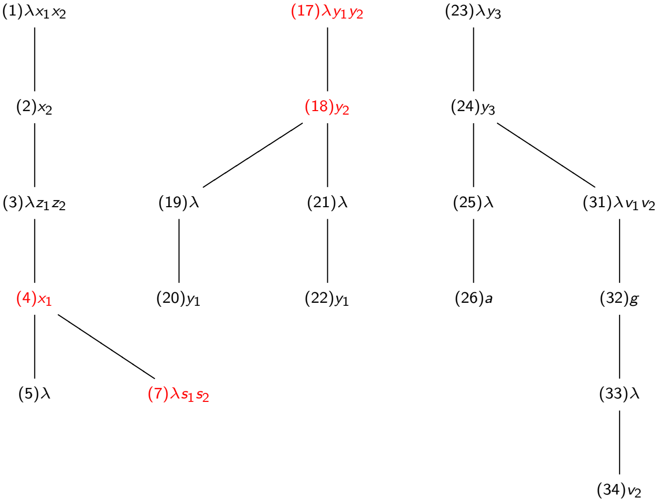
- Combinatorial argument based on uniformities of play
- Two steps in proof assuming an arbitrary solution term
    1. partition each play
- 5th-order example

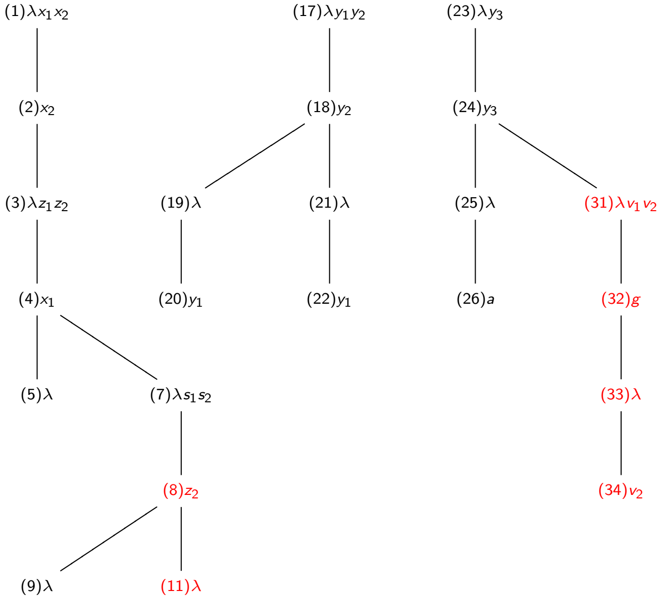
$$x(\lambda y_1 y_2 . y_2 y_1 y_1)(\lambda y_3 . y_3 a(\lambda v_1 v_2 . g v_2)) \;=\; ga$$

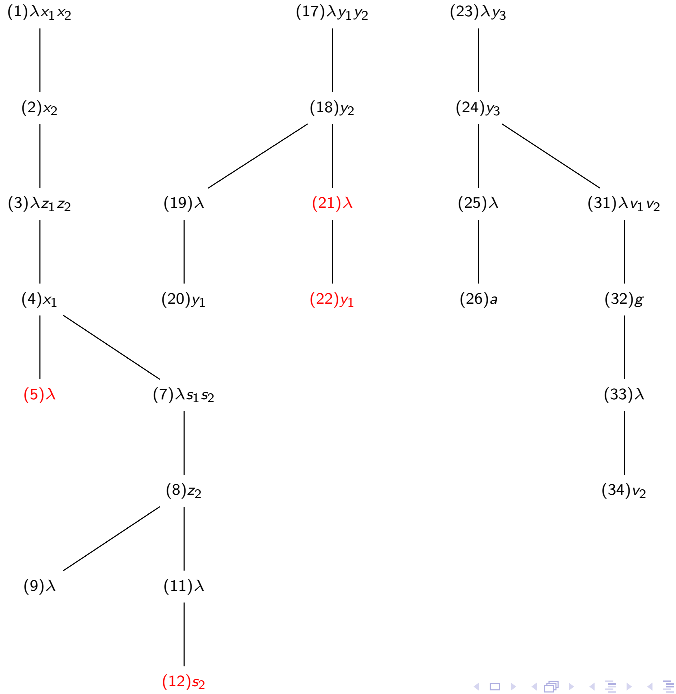

(1)$\lambda x_1 x_2$
(2)x_2
(3)$\lambda z_1 z_2$
(4)x_1
(5)λ
(6)z_1
(7)$\lambda s_1 s_2$
(8)z_2
(9)λ
(10)b
(11)λ
(12)s_2
(17)$\lambda y_1 y_2$
(18)y_2
(19)λ
(20)y_1
(21)λ
(22)y_1
(23)λy_3
(24)y_3
(25)λ
(26)a
(31)$\lambda v_1 v_2$
(32)g
(33)λ
(34)v_2


(1)$\lambda x_1 x_2$
(17)$\lambda y_1 y_2$
(23)λy_3
(18)y_2
(24)y_3
(19)λ
(21)λ
(25)λ
(31)$\lambda v_1 v_2$
(20)y_1
(22)y_1
(26)a
(32)g
(33)λ
(34)v_2

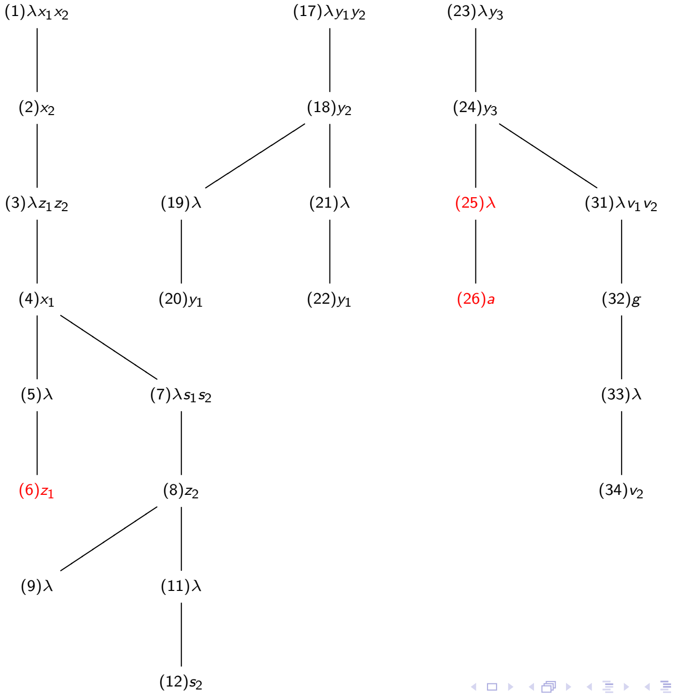

$\lambda x_1 x_2$
(2)x_2
(3)$\lambda z_1 z_2$
(17)$\lambda y_1 y_2$
(18)y_2
(19)λ
(20)y_1
(21)λ
(22)y_1
(23)λy_3
(24)y_3
(25)λ
(26)a
(31)$\lambda v_1 v_2$
(32)g
(33)λ
(34)v_2


(1)$\lambda x_1 x_2$
(2)x_2
(3)$\lambda z_1 z_2$
(4)x_1
(5)λ
(7)$\lambda s_1 s_2$
(17)$\lambda y_1 y_2$
(18)y_2
(19)λ
(20)y_1
(21)λ
(22)y_1
(23)λy_3
(24)y_3
(25)λ
(26)a
(31)$\lambda v_1 v_2$
(32)g
(33)λ
(34)v_2


(1)λx1x2
(2)x2
(3)λz1z2
(4)x1
(5)λ
(7)λs1s2
(8)z2
(9)λ
(11)λ
(17)λy1y2
(18)y2
(19)λ
(20)y1
(21)λ
(22)y1
(23)λy3
(24)y3
(25)λ
(26)a
(31)λv1v2
(32)g
(33)λ
(34)v2



(1)$\lambda x_1 x_2$
(2)x_2
(3)$\lambda z_1 z_2$
(4)x_1
(5)λ
(7)$\lambda s_1 s_2$
(8)z_2
(9)λ
(11)λ
(12)s_2
(17)$\lambda y_1 y_2$
(18)y_2
(19)λ
(20)y_1
(21)λ
(22)y_1
(23)λy_3
(24)y_3
(25)λ
(26)a
(31)$\lambda v_1 v_2$
(32)g
(33)λ
(34)v_2


(1)$\lambda x_1 x_2$
(2)x_2
(3)$\lambda z_1 z_2$
(4)x_1
(5)λ
(6)z_1
(7)$\lambda s_1 s_2$
(8)z_2
(9)λ
(11)λ
(12)s_2
(17)$\lambda y_1 y_2$
(18)y_2
(19)λ
(20)y_1
(21)λ
(22)y_1
(23)λy_3
(24)y_3
(25)λ
(26)a
(31)$\lambda v_1 v_2$
(32)g
(33)λ
(34)v_2

Application of games to decidability of matching

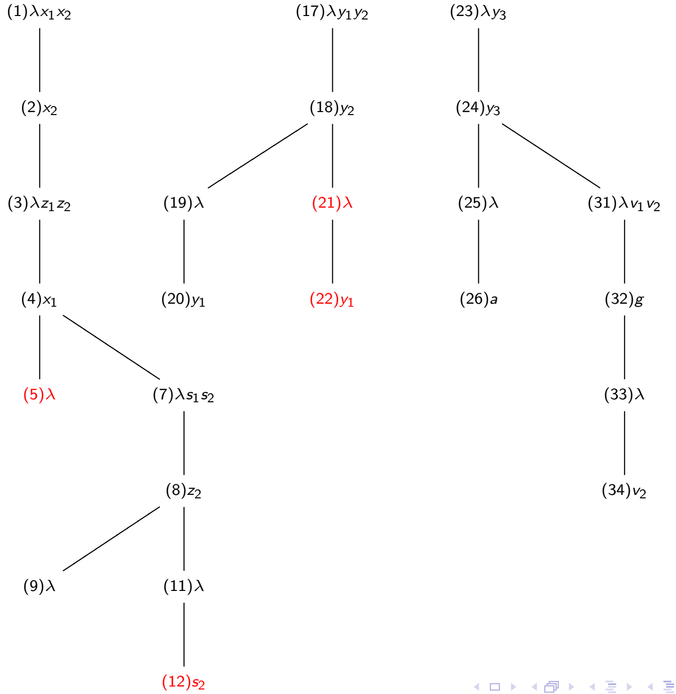
- Combinatorial argument based on uniformities of play.
- Two steps in proof assuming an arbitrary solution term
  1. partition each play
- 5th-order example

$$x(\lambda y_1 y_2.y_2 y_1 y_1)(\lambda y_3.y_3 a(\lambda v_1 v_2.g v_2)) \;=\; ga$$

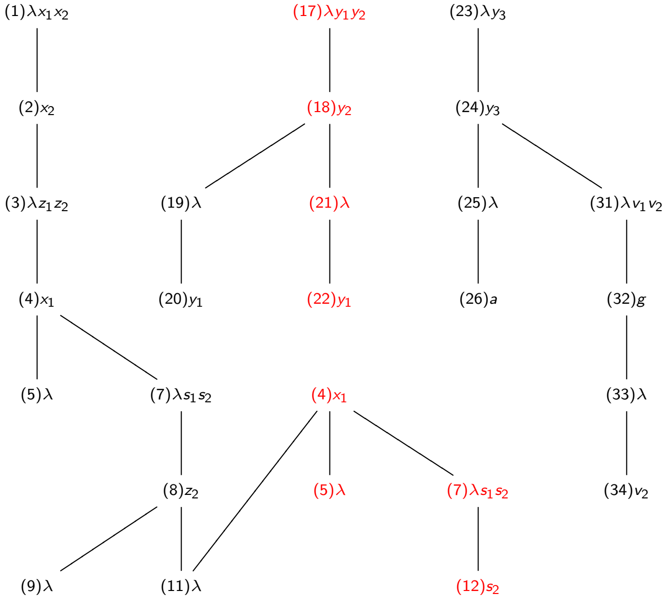
Application of games to decidability of matching

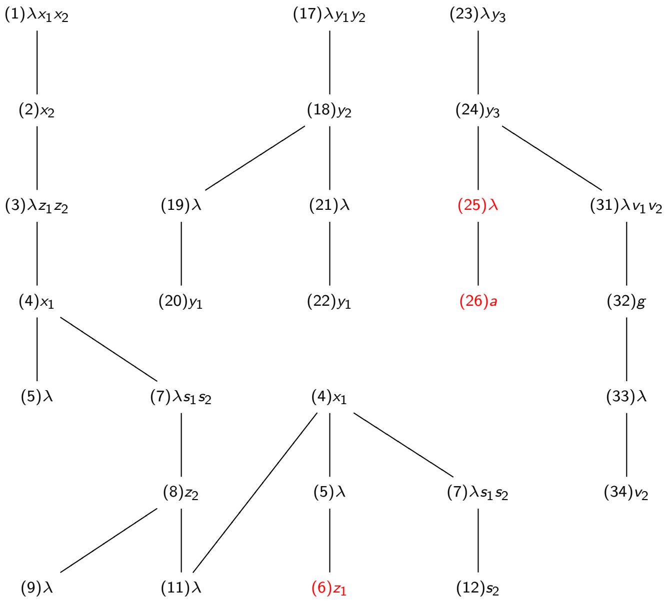
- Combinatorial argument based on uniformities of play.
- Two steps in proof assuming an arbitrary solution term
  1. partition each play
  2. unfold into a small solution (by adding prefixes) and then removing redundant parts of term
- 5th-order example

$$x(\lambda y_1 y_2 . y_2 y_1 y_1)(\lambda y_3 . y_3 a(\lambda v_1 v_2 . g v_2)) \;=\; ga$$

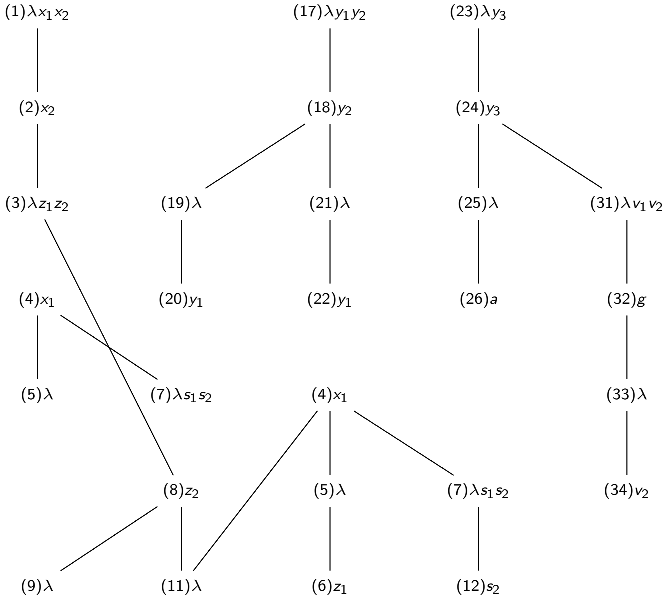


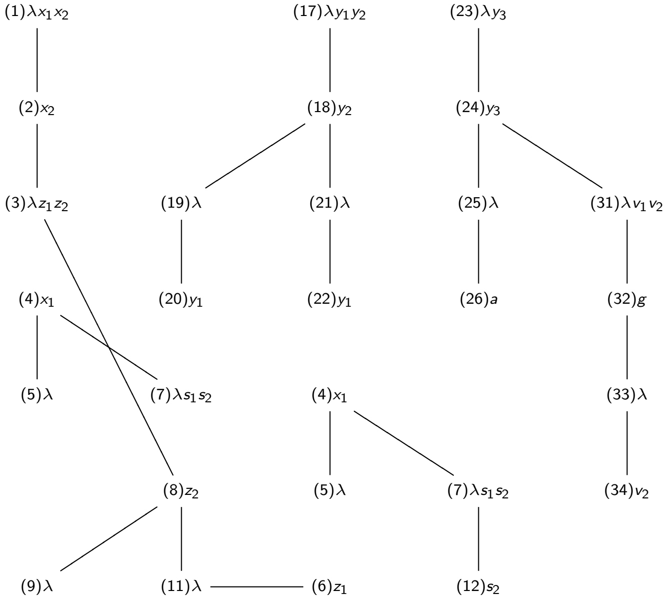
(1)$\lambda x_1 x_2$
(2)x_2
(3)$\lambda z_1 z_2$
(4)x_1
(5)λ
(7)$\lambda s_1 s_2$
(8)z_2
(9)λ
(11)λ
(12)s_2
(17)$\lambda y_1 y_2$
(18)y_2
(19)λ
(20)y_1
(21)λ
(22)y_1
(23)λy_3
(24)y_3
(25)λ
(26)a
(31)$\lambda v_1 v_2$
(32)g
(33)λ
(34)v_2


(1)λx1x2
(17)λy1y2
(23)λy3
(2)x2
(18)y2
(24)y3
(3)λz1z2
(19)λ
(21)λ
(25)λ
(31)λv1v2
(4)x1
(20)y1
(22)y1
(26)a
(32)g
(5)λ
(7)λs1s2
(4)x1
(33)λ
(8)z2
(5)λ
(7)λs1s2
(34)v2
(9)λ
(11)λ
(12)s2

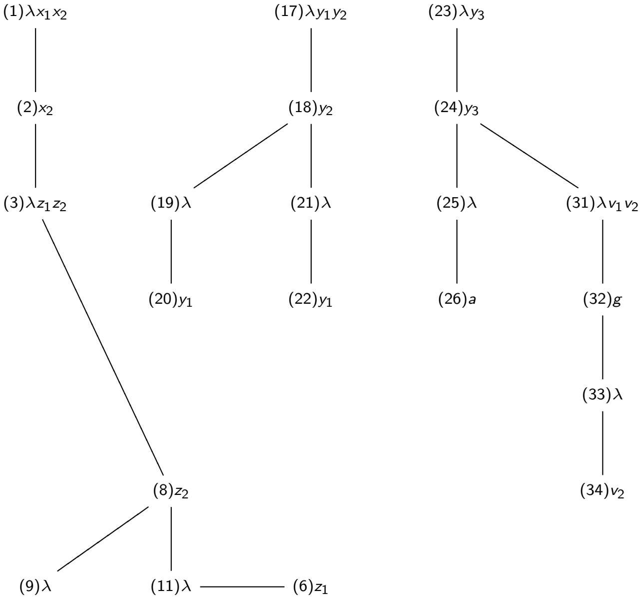


(1)$\lambda x_1 x_2$
(2)x_2
(3)$\lambda z_1 z_2$
(4)x_1
(5)λ
(7)$\lambda s_1 s_2$
(8)z_2
(9)λ
(11)λ
(17)$\lambda y_1 y_2$
(18)y_2
(19)λ
(20)y_1
(21)λ
(22)y_1
(4)x_1
(5)λ
(6)z_1
(7)$\lambda s_1 s_2$
(12)s_2
(23)λy_3
(24)y_3
(25)λ
(26)a
(31)$\lambda v_1 v_2$
(32)g
(33)λ
(34)v_2


(1)λx1x2
(17)λy1y2
(23)λy3
(2)x2
(18)y2
(24)y3
(3)λz1z2
(19)λ
(21)λ
(25)λ
(31)λv1v2
(4)x1
(20)y1
(22)y1
(26)a
(32)g
(5)λ
(7)λs1s2
(4)x1
(33)λ
(8)z2
(5)λ
(7)λs1s2
(34)v2
(9)λ
(11)λ
(6)z1
(12)s2



(1)$\lambda x_1 x_2$
(2)x_2
(3)$\lambda z_1 z_2$
(4)x_1
(5)λ
(7)$\lambda s_1 s_2$
(8)z_2
(9)λ
(11)λ
(6)z_1
(17)$\lambda y_1 y_2$
(18)y_2
(19)λ
(20)y_1
(21)λ
(22)y_1
(4)x_1
(5)λ
(7)$\lambda s_1 s_2$
(12)s_2
(23)λy_3
(24)y_3
(25)λ
(26)a
(31)$\lambda v_1 v_2$
(32)g
(33)λ
(34)v_2


(1)$\lambda x_1 x_2$
(2)x_2
(3)$\lambda z_1 z_2$
(8)z_2
(9)λ
(11)λ
(6)z_1
(17)$\lambda y_1 y_2$
(18)y_2
(19)λ
(20)y_1
(21)λ
(22)y_1
(23)λy_3
(24)y_3
(25)λ
(26)a
(31)$\lambda v_1 v_2$
(32)g
(33)λ
(34)v_2

Application of games to decidability of matching

- Combinatorial argument based on uniformities of play.
- Two steps in proof assuming an arbitrary solution term
    1. partition each play
    2. unfold into a small solution (by adding prefixes) and then remove redundant parts of term

Application of games to decidability of matching

- Combinatorial argument based on uniformities of play.
- Two steps in proof assuming an arbitrary solution term
  1. partition each play
  2. unfold into a small solution (by adding prefixes) and then remove redundant parts of term
- Small term property

Application of games to decidability of matching

- Combinatorial argument based on uniformities of play.
- Two steps in proof assuming an arbitrary solution term
  1. partition each play
  2. unfold into a small solution (by adding prefixes) and then remove redundant parts of term
- Small term property
- Theorem If $x : A$ and A has order $2p + 2$ or $2p + 3$ and arity q then $xw_1 \ldots w_k = u$ has a (canonical) solution iff it has a (canonical) solution of depth at most $O(p^2 q^{2p} |u|)$

The retract problem

Type A is a retract of B, if there are $t : A \longrightarrow B$ and $s : B \longrightarrow A$ such that $s(tx) =_{\beta\eta} x$

DECISION PROBLEM: given A, B, is A a retract of B ?

The retract problem

Type A is a retract of B, if there are $t : A \rightarrow B$ and $s : B \rightarrow A$ such that $s(tx) =_{\beta\eta} x$

DECISION PROBLEM: given A, B, is A a retract of B ?

- Bruce and Longo [STOC 85] solve problem for $=_\beta$ by providing a simple proof system. Much harder for $=_{\beta\eta}$

The retract problem

Type A is a retract of B, if there are $t : A \rightarrow B$ and $s : B \rightarrow A$ such that $s(tx) =_{\beta\eta} x$

DECISION PROBLEM: given A, B, is A a retract of B ?

- Bruce and Longo [STOC 85] solve problem for $=_{\beta}$ by providing a simple proof system. Much harder for $=_{\beta\eta}$
- De Liguro, Piperno and Statman [LICS 92] solve affine case for $=_{\beta\eta}$ when single base type by providing a proof system
- Generalised by Regnier and Urzyczyn [2002] to arbitrary base types: proof system for affine case provides NP decision procedure

The retract problem

Type A is a retract of B, if there are $t : A \rightarrow B$ and $s : B \rightarrow A$ such that $s(tx) =_{\beta\eta} x$

DECISION PROBLEM: given A, B, is A a retract of B ?

- Bruce and Longo [STOC 85] solve problem for $=_{\beta}$ by providing a simple proof system. Much harder for $=_{\beta\eta}$
- De Liguro, Piperno and Statman [LICS 92] solve affine case for $=_{\beta\eta}$ when single base type by providing a proof system
- Generalised by Regnier and Urzyczyn [2002] to arbitrary base types: proof system for affine case provides NP decision procedure
- Padovani [TLCA 01] shows decidability for general case when single base type (no complexity bound)

The retract problem

Type A is a retract of B, if there are $t : A \longrightarrow B$ and $s : B \longrightarrow A$ such that $s(tx) =_{\beta\eta} x$

DECISION PROBLEM: given A, B, is A a retract of B ?

- Bruce and Longo [STOC 85] solve problem for $=_{\beta}$ by providing a simple proof system. Much harder for $=_{\beta\eta}$
- De Liguro, Piperno and Statman [LICS 92] solve affine case for $=_{\beta\eta}$ when single base type by providing a proof system
- Generalised by Regnier and Urzyczyn [2002] to arbitrary base types: proof system for affine case provides NP decision procedure
- Padovani [TLCA 01] shows decidability for general case when single base type (no complexity bound)
- Decidability of general case follows from decidability of higher-order matching; non-elementary complexity bound

The retract problem

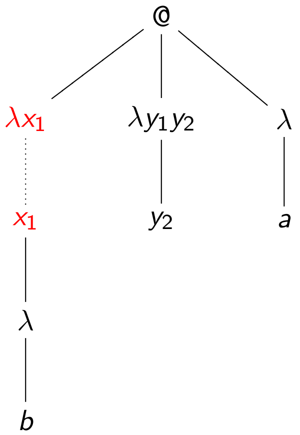
Type A is a retract of B, if there are $t : A \rightarrow B$ and $s : B \rightarrow A$ such that $s(tx) =_{\beta\eta} x$

DECISION PROBLEM: given A, B, is A a retract of B ?

- Bruce and Longo [STOC 85] solve problem for $=_\beta$ by providing a simple proof system. Much harder for $=_{\beta\eta}$
- De Liguro, Piperno and Statman [LICS 92] solve affine case for $=_{\beta\eta}$ when single base type by providing a proof system
- Generalised by Regnier and Urzyczyn [2002] to arbitrary base types: proof system for affine case provides NP decision procedure
- Padovani [TLCA 01] shows decidability for general case when single base type (no complexity bound)
- Decidability of general case follows from decidability of higher-order matching; non-elementary complexity bound
- Proof system for general case with EXPSPACE upper bound; soundness and completeness uses games [ICALP 13]

Games only work for long normal forms

- $x : ((\mathbf{0},\mathbf{0},\mathbf{0}),\mathbf{0},\mathbf{0})$ β-interpolation problem $x(\lambda y_1 y_2.y_2)a = a$



- $\lambda x_1.x_1 b(\lambda y_1 y_2.y_2)a \to_\beta^* a$
- No natural game?