Higher-Order Matching, Games and Automata

Colin Stirling
School of Informatics
University of Edinburgh

- Notable success in computer science: methods for verifying finite + infinite state systems.

  model checking + equivalence checking

- System = finite/infinite state transition graph

Model checking: $s_0 \models \phi$?

initial state

temporal property

apply automata-theoretic/game-theoretic techiques to solve it. Mostly computing monadic fixed points by traversing graph (possibly repeatedly)

Equivalence checking: $s \equiv t$?

mostly computing dyadic fixed points e.g., bisimulation equivalence. May need some algebraic/combinatorial properties of generators of transition graph.

Active research goal : transfer these techniques to finite/infinite state systems with binding

- Deciding observational equivalence for fragments of idealized Algol (w.r.t finite value sets)
  [Ghica + McCusker 2000, Ong LICS 2002, ...]
- Model checking higher order trees
  [Knapik, Niwinski, Urzyczyn 2002, Caucal 2002, Ong LICS 2006, ...]
- ...
- Application of tree automata to matching
  [Comon + Jurski 1997]

generalize here

Higher order schemes

Damm, ... 1970s

Base type 0 - finite/infinite trees with nodes labelled by elements of $f_1, \ldots, f_n$:

each f_i has arity ≥ 0

Scheme

typed variables

$$F_i(x_1, \ldots, x_{n_i}) = t_i \quad 1 \leq i \leq m$$

each F_i is typed

$t_i : 0$ built from x_j's, F_j's and f_j's + application

$+$

start closed expression

$$S : 0$$

(Avoiding λ-terms)

Interpretation of a scheme: finite/infinite tree generated by S.

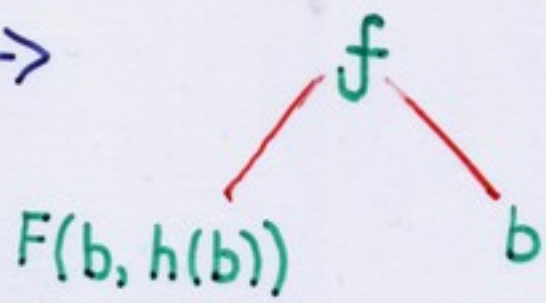
Example : first-order

variables x_1, x_2
ground type

$$F(x_1, x_2) = f(F(x_1, h(x_2)), x_2)$$

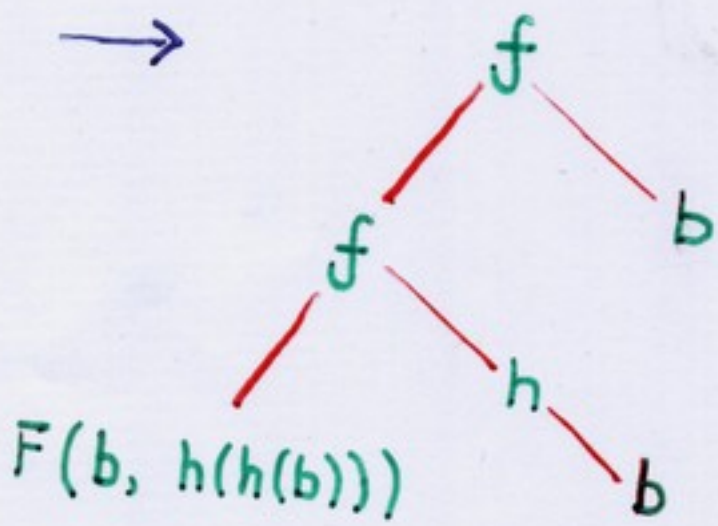
$F(b, b)$ start

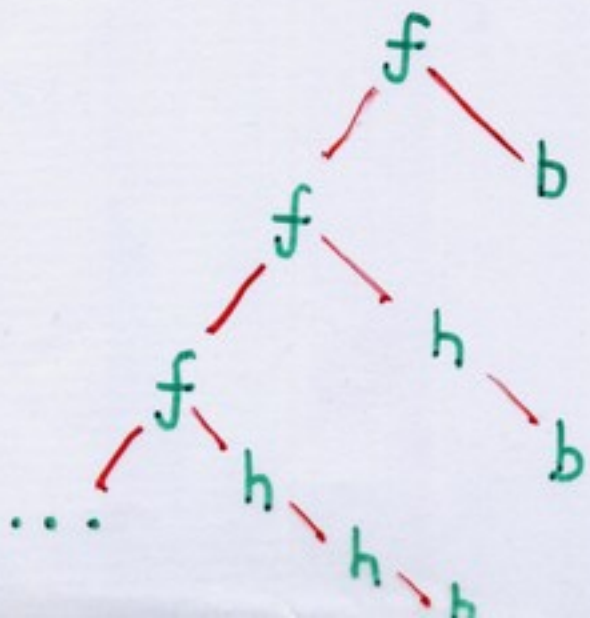
$F(b,b) \longrightarrow$



f arity 2
h " 1
b arity 0

$\longrightarrow$





infinite tree

2nd-order example

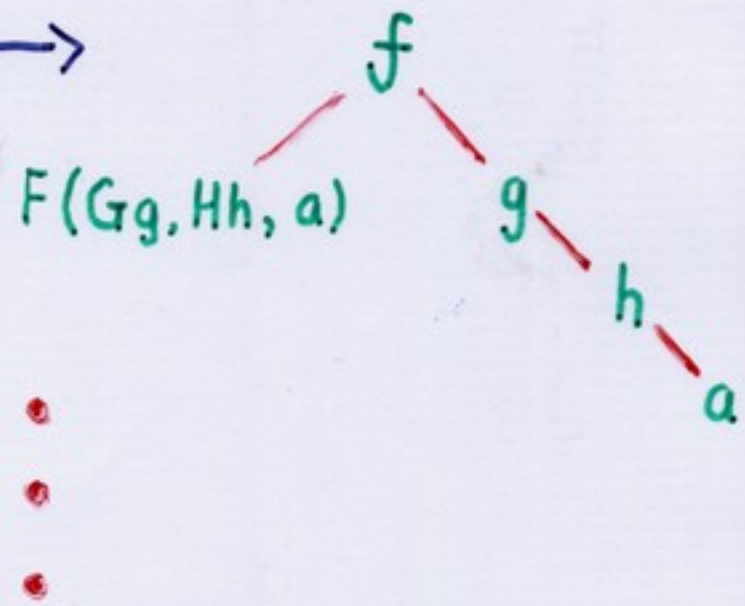
$$F(x_1, x_2, x_3) = f(F(Gx_1, Hx_2, x_3), x_1(x_2(x_3)))$$

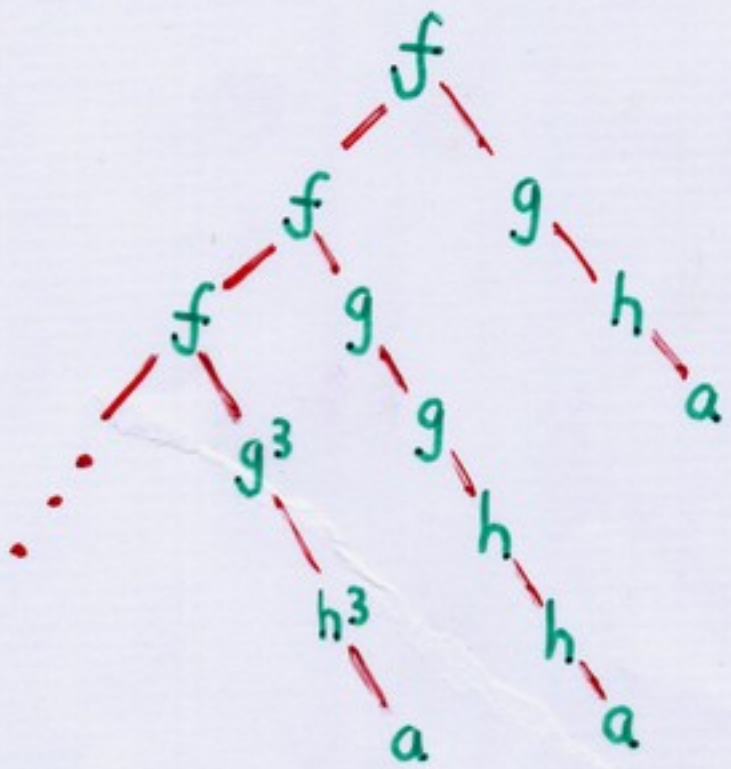
$$G(x_1, x_2) = g(x_1(x_2))$$

$$H(x_1, x_2) = h(x_1(x_2))$$

$F(g, h, a)$ start

$F(g,h,a) \longrightarrow$





infinite tree

Model checking problem

- Given S, does its tree have a decidable monadic 2nd-order theory?

- Solved +vely for a subset of schemes, safe schemes [Knapik, Niwinski, Urzyczyn Fossacs 2002]

  uses geometry of interaction on infinite λ-terms + higher order push down automata

- Extended to all schemes [Ong LICS 2006]

  uses game semantics on infinite λ-terms

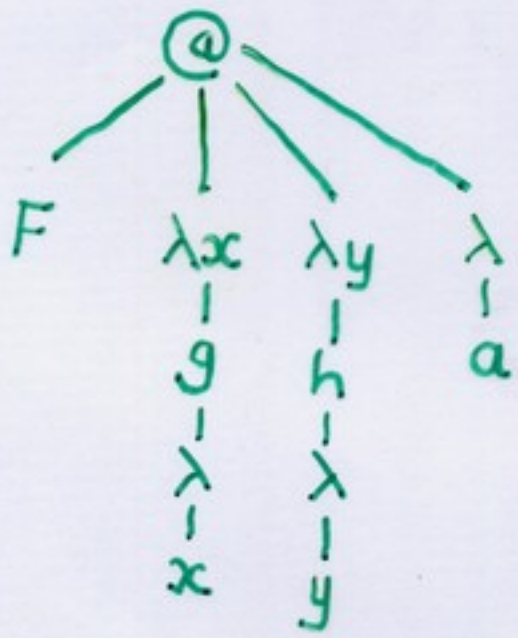
Fgha

$$F x_1 x_2 x_3 = f(F(Gx_1)(Hx_2)x_3,\ x_1(x_2(x_3)))$$

$$G x_1 x_2 = g(x_1(x_2))$$

$$H x_1 x_2 = h(x_1(x_2))$$

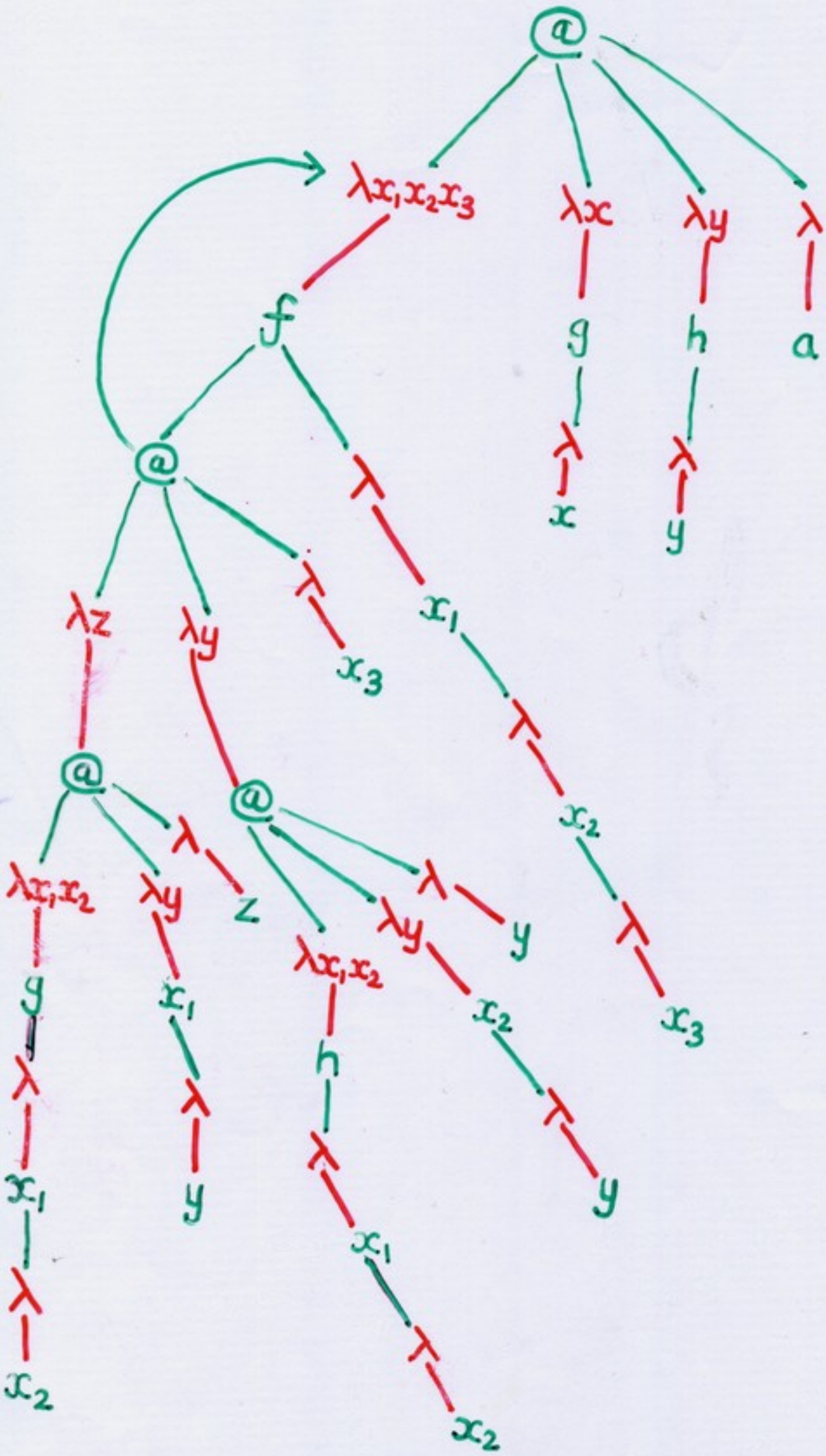
Follow Ong's transformation into normal form



$$F = \lambda x_1 x_2 x_3 .\ f(@\ \ldots\ldots,\ \lambda.\ x_1(\lambda.\ x_2(\lambda.x_3)))$$

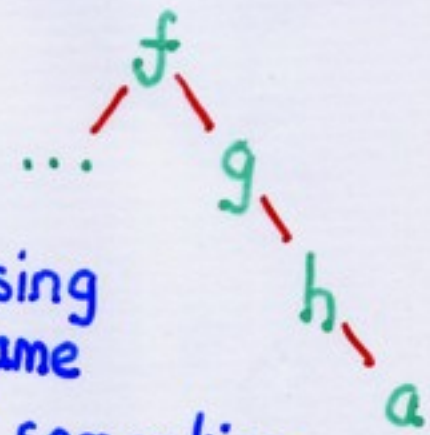
$\vdots$

Infinite λ-term (can be folded into a finite tree with back edges)


@
λx₁x₂x₃
λx
λy
λ
f
g
h
a
λ
λ
x
y
@
λ
x₁
λz
λy
λ
x₃
x₂
λ
@
@
λx₁x₂
λy
λ
z
λx₁x₂
λy
λ
y
x₁
x₂
x₃
y
λ
λ
h
λ
y
λ
x₁
λ
x₂
λ
x₂

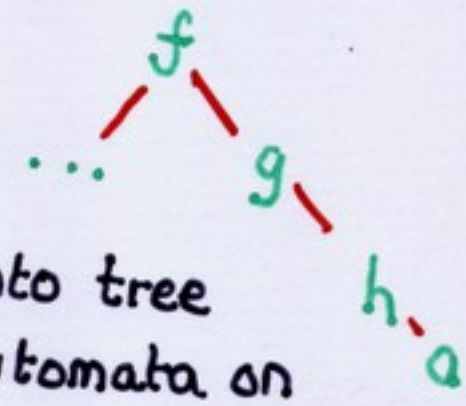
"capture"
infinite tree

using
game
semantics



$\Downarrow$

transform parity
tree automata on



into tree
automata on
"finite" unfolding
of λ-term

Equivalence problem

from 1970's

- Given two schemes S_1, S_2 do they generate the same tree?

$$\boxed{S_1 \sim S_2 \ ?}$$

bisimilarity

- For order 1:

$$S_1 \sim S_2 \ ?$$
$$|||$$

Courcelle 1978

DPDA equivalence problem

do 2 configurations of a deterministic pushdown automaton generate the same language?

$\therefore$ Solved +vely by Sénizergues TCS 2001.

simpler proof with complexity upper bound ICALP 2002

- Order > 1

$$\boxed{\text{Open + hard}}$$

decidable for a subset of 2nd-order: uses 2nd-order grammars + combinatorics from concur 2006.

Simply typed λ-calculus

Church style

base type

types $\quad A ::= O \mid A \to A$

a type always has form $A_1 \to (\dots (A_n \to O) \dots)$

$$\boxed{(A_1, \dots, A_n, O)}$$

order

$\text{order}(O) = 1$

$\text{order}((A_1, \dots, A_n, O)) = \max\{\text{order}(A_i)\} + 1$

terms $\quad T$

variables, constants have unique type

$x : A,\ f : A \Rightarrow x : A,\ f : A \in T$

$x : A,\ t : B \in T \Rightarrow \lambda x.t : A \to B \in T$

$t : A \to B,\ u : A \in T \Rightarrow t\,u : B$

closed $\quad t : A$ contains no free variables

α-equivalence $\quad t, t'$ are renamings of each other w.r.t bound variables

Reduction et al

substitution

(β) $(\lambda x.t)v \longrightarrow_\beta t\{v/x\}$

(η) $\lambda x.(t\,x) \longrightarrow_\eta t$ $\quad$ x not free in t

...

Fact Each term has a unique $\beta\eta$-normal form (up to α-equivalence)

no further reductions

$\boxed{=_{\beta\eta} \quad \text{same normal form}}$

η-long normal form

constant/variable

$(B_1, \ldots, B_k, O)$ + long normal form

$t : O \qquad u : O \quad \text{or} \quad u\, t_1 \ldots t_k$

$t : (A_1, \ldots, A_n, O) \qquad \lambda x_1 \ldots x_n.\, t'$

abbreviates $\lambda x_1 \ldots \lambda x_n.\, t'$

$t' : O$ in long normal form

Fact When η-long normal forms $=_\beta$ is $=_{\beta\eta}$

Higher-order unification

Huet 1976

- $V = U$ contains free variables $x_1, \ldots, x_n$

Solution: terms $t_1, \ldots, t_n$ (t_i same type as x_i)
such that $V\theta =_{\beta\eta} U\theta$ $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$

Decision question: given $V = U$ does it have a solution?

order is largest order of variable x_i

- Undecidable (even at order 2)

[Goldfarb '81]

can encode $+, \times$ over Church numbers
$\therefore$ Hilberts 10th Problem as a unification problem

[Huet '72]

Post's correspondence

Huet 76

Higher order matching

- $v = u$ BUT u closed

also assume $v, u : o$ without loss of generality

Solution : terms $t_1, \ldots, t_n$ such that

$$v\{t_1/x_1, \ldots, t_n/x_n\} =_{\beta\eta} u$$

Decision question : given $v = u$ does it have a solution?

order : largest order of x_i in v.

- Decidable order 2 (Huet '76)
  order 3 (Dowek '93)
  order 4 (Padovani 2000)
  \+ special cases (Padovani, Schubert, ... Schimdt-Schauß, ...)

- Undecidable for $=_\beta$ [Loader 2003]

Matching essentially monadic

Schubert
Padovani

Given $v = u$ with $x_1 : A_1, \ldots, x_n : A_n$ free in v

$\Downarrow$

$x\ \lambda x_1 \ldots x_n . v = u$ $\quad x : ((A_1, \ldots, A_n, 0), 0)$

one free variable

"conceptually" simpler problem

interpolation problem

$x\ v_1 \ldots v_n = u$

closed + in η-long normal form

$x : (A_1, \ldots, A_n, 0)$

Solution: closed t in normal form

$t\ v_1 \ldots v_n =_{\beta\eta} u$

Padovani Proof at Order 4

- $t, t' : A$

  $t \equiv_A^u t'$ if t, t' solve same interpolation problems with right term u

  $\boxed{\equiv_A^u : \text{"observational equivalence"}}$

- Set of normal forms : $A / \equiv_A^u$ is finite

  quotient

- Up to order 4, representatives for each equivalence class can be decided.

- [Comon + Jurski 1997]

  Fact Set of solutions to 4th order interpolation is regular (modulo ...) $\doteq$ recognizable by a tree automaton.

  States of automaton use Padovani representatives $\equiv_{A'}^{u'}$ for subterms u' of u, subtypes A' of A.

Model checking vs Satisfiability checking

- Given M and ϕ, $M \models \phi$?

  Given ϕ, is there an M s.t. $M \models \phi$?

Similar for matching:

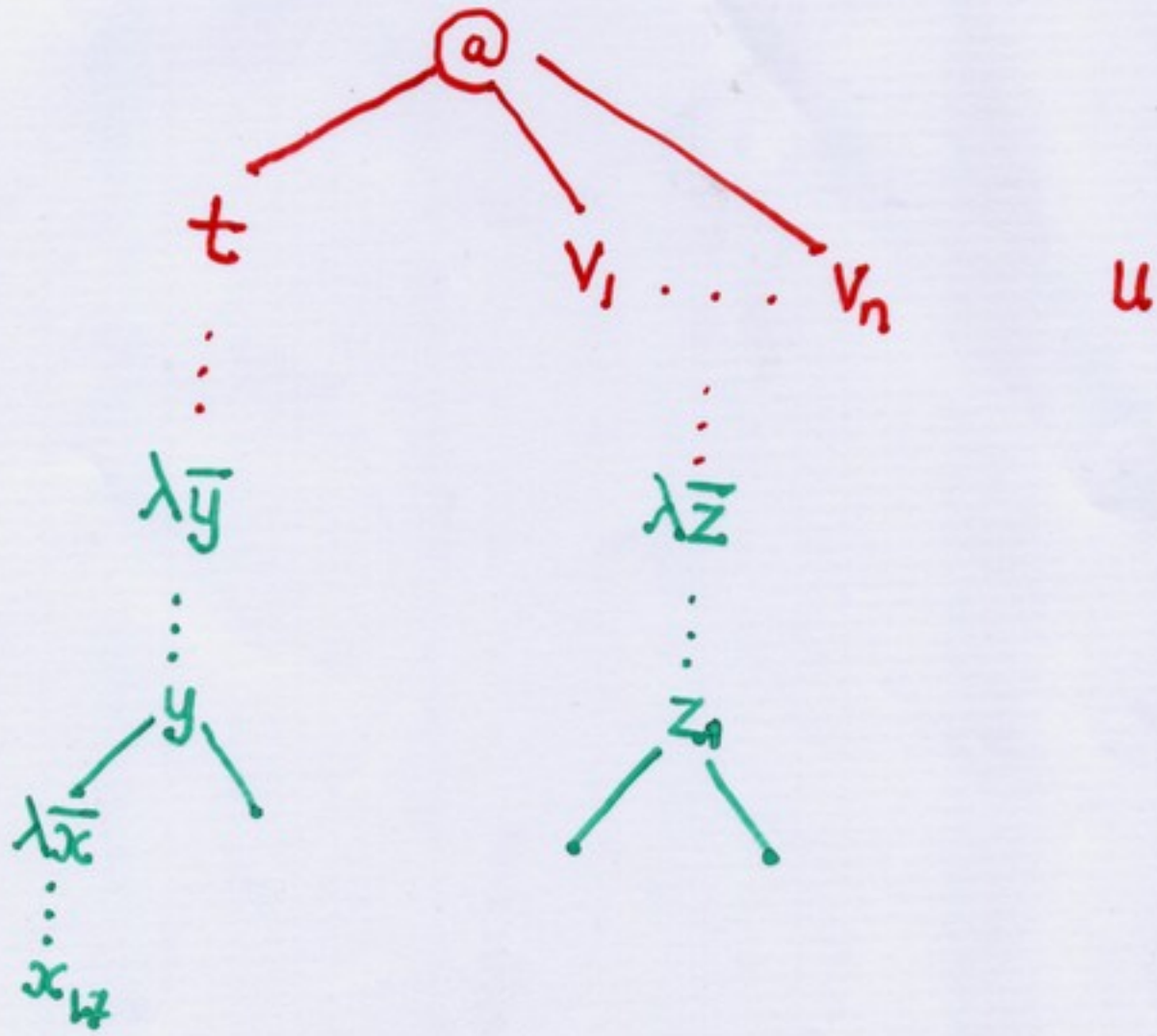
given $x\, v_1 .. v_n = u$
and t is $t\, v_1 .. v_n =_{\beta\eta} u$

given $x\, v_1 .. v_n = u$
is there a non-trivial t
s.t. $t\, v_1 .. v_n =_{\beta\eta} u$

Interpolation game

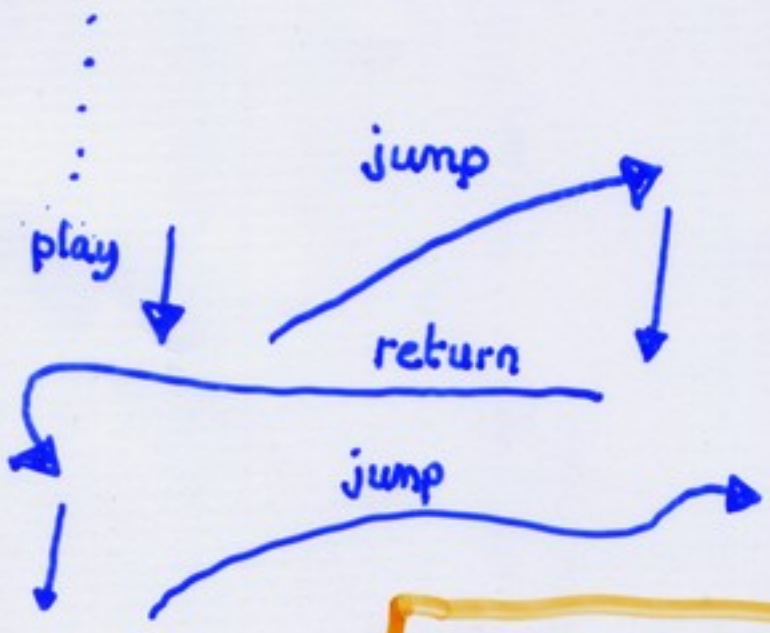
- Given $x\, v_1 \ldots v_n = u$ with $x : A$, and $t : A$ in normal form.

- Define game whose board is :



Like game semantics

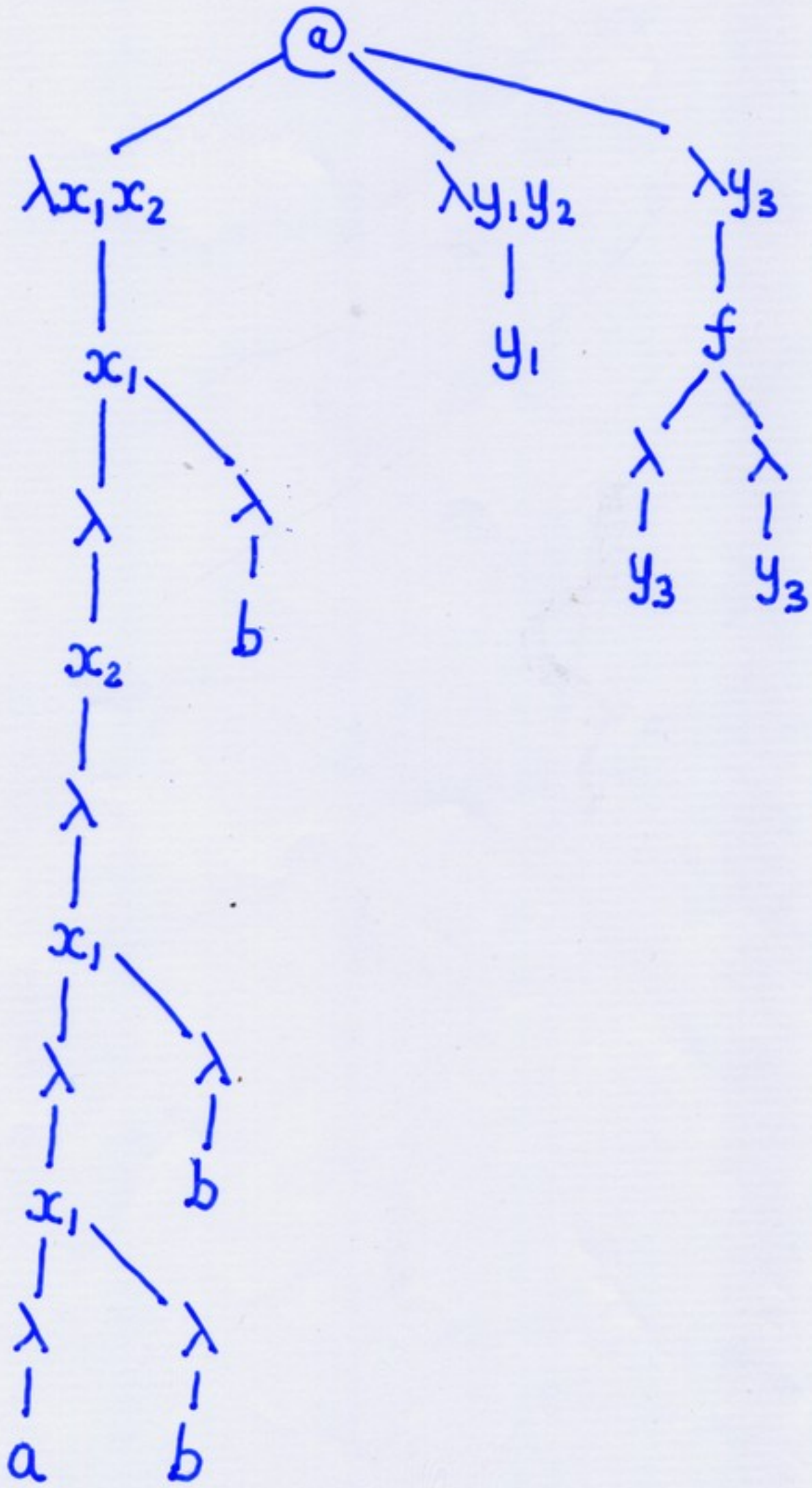
play starts here



choices: determined by branches of u

Model β-reduction without destroying terms by substituting into them

$$x\ (\lambda y_1 y_2.\ y_1)\ (\lambda y_3.\ f\ y_3\ y_3) = f\ a\ a$$



- Given $x v_1 .. v_n = u$, $x : A$ and $t : A$ there is the game $G(t, v_1 \ldots v_n, u)$

<u>Fact</u> $\forall$ loses $G(t, v_1 .. v_n, u)$ iff $t v_1 .. v_n =_{\beta\eta} u$.

- Abstract from game playing (jumping) to a tree automaton

Restrict variables + constants allowed in a t

- Variable assumptions

based on/borrowed from [Ong 2006]

$z : O$ $\quad (z, r, \emptyset)$

r subterm of u

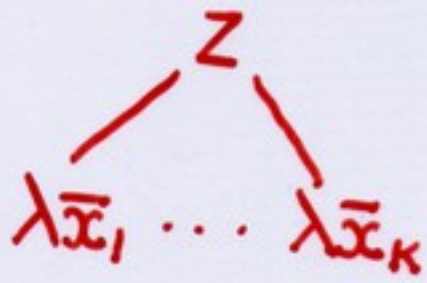
$z : (A_1, \ldots, A_n, O)$ $\quad (z, r, \Gamma_1 \cup \ldots \cup \Gamma_n)$

Γ_i is a set of assumptions of variables of type A_i

- State of automaton: $\{(r_1, \Gamma_1), \ldots, (r_k, \Gamma_k)\}$

Γ_i's are variable assumptions

- Transitions of form $z\,\Lambda_1 \ldots \Lambda_k \Rightarrow \Lambda$



if $(r, \Gamma) \in \Lambda$ then $(z, r, \Gamma') \in \Gamma$

and Γ' is

$$\{ (y_{11}, r_{11}, \Gamma'_{11}), \ldots, (y_{1m_1}, r_{1m_1}, \Gamma'_{1m_1}),$$
$$\vdots \qquad\qquad \vdots$$
$$(y_{k1}, r_{k1}, \Gamma'_{k1}), \ldots, (y_{km_k}, r_{km_k}, \Gamma'_{km_k}) \}$$

and $(r_{ij}, \Gamma'_{ij} \cup \Sigma_{ij}) \in \Lambda_i$

and $\ldots$

$\vdots$

Fact Given $x\, v_1 \ldots v_n = u$ with $x : A$ order$(A) > 0$, the set of solutions built out of a fixed finite alphabet of variables/constants is regular (recognized by a tree automaton).

[LICS 2007]

generalizes Comon + Jurski's result

- proof technique more elementary (games instead of observational equivalence classes)

- Not enough to prove decidability for order$(A) > 4$ of matching

Problem

When A has order > 4, set of closed terms may require infinitely many different variables.

$$A = ((((0,0),0),0),0,0)$$

Monster type

$\lambda x_1 x_2$

$x_1 : (((0,0),0),0)$

λz_1

$x_1 : (((0,0),0),0)$

λz_n

$z_n : (0,0)$

$z_1 : (0,0)$

$x_2 : 0$

each occurrence of x_1 introduces new variables z_i that have to be distinct

z_i's are embedded

Satisfiability checking

does ϕ have a model?

Model checking games can solve satisfiability checking.

Prove small model property

if there is a model for ϕ, then there is one of size at most $f(\phi)$

tree model property

Techniques

no play returns to same state

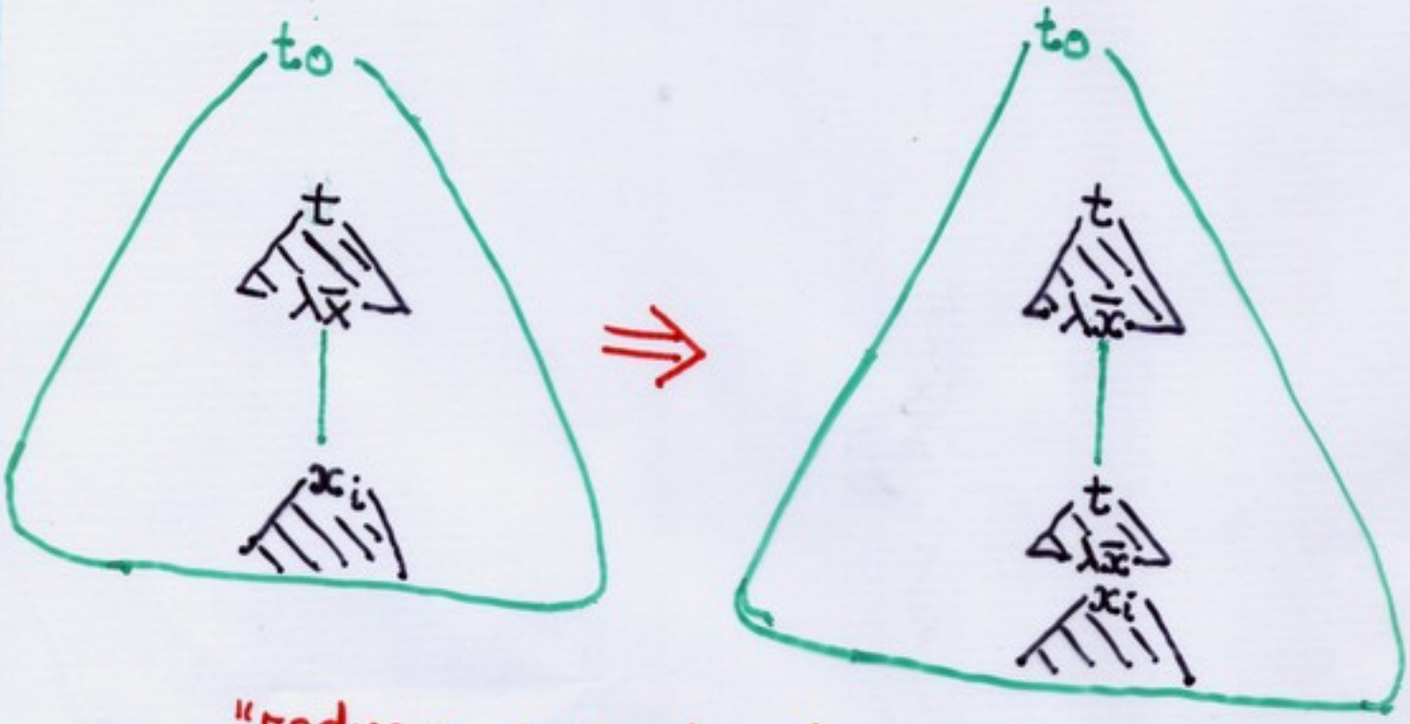
- unravel model into tree
- no play enters a subtree: delete it
- two different plays have positions $(s, EX\phi)$, $(s, EX\psi)$ and only one state t. $s \to t$. Transform model

Play separation: $\exists$ chooses left t in $(s, EX\phi)$
$\exists$ chooses right t in $(s, EX\psi)$

For matching analogous

approximate tree model property

- unravel = unfold w.r.t playing which incorporates binding



"reduces game play jumping"
(localizes it)

- Very complicated. [ICALP 2006]

Non-elementary: small model property grows exponentially with order

- Find a simpler proof!