Transfer Theorem

Sylvain Salvati and Igor Walukiewicz

Bordeaux University

$$M \xrightarrow{eval} BT(M)$$

$$M \xrightarrow{eval} BT(M)$$

Transfer Theorem

For all φ exists $\widehat{\varphi}$ s.t.

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi$$

$$M \xrightarrow{eval} BT(M)$$

Transfer Theorem

For all $\Sigma, \mathcal{T}, \mathcal{X}$.
For all φ exists $\widehat{\varphi}$ s.t. for all $M \in Terms(\Sigma, \mathcal{T}, \mathcal{X})$:

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi$$

Example: Unfolding

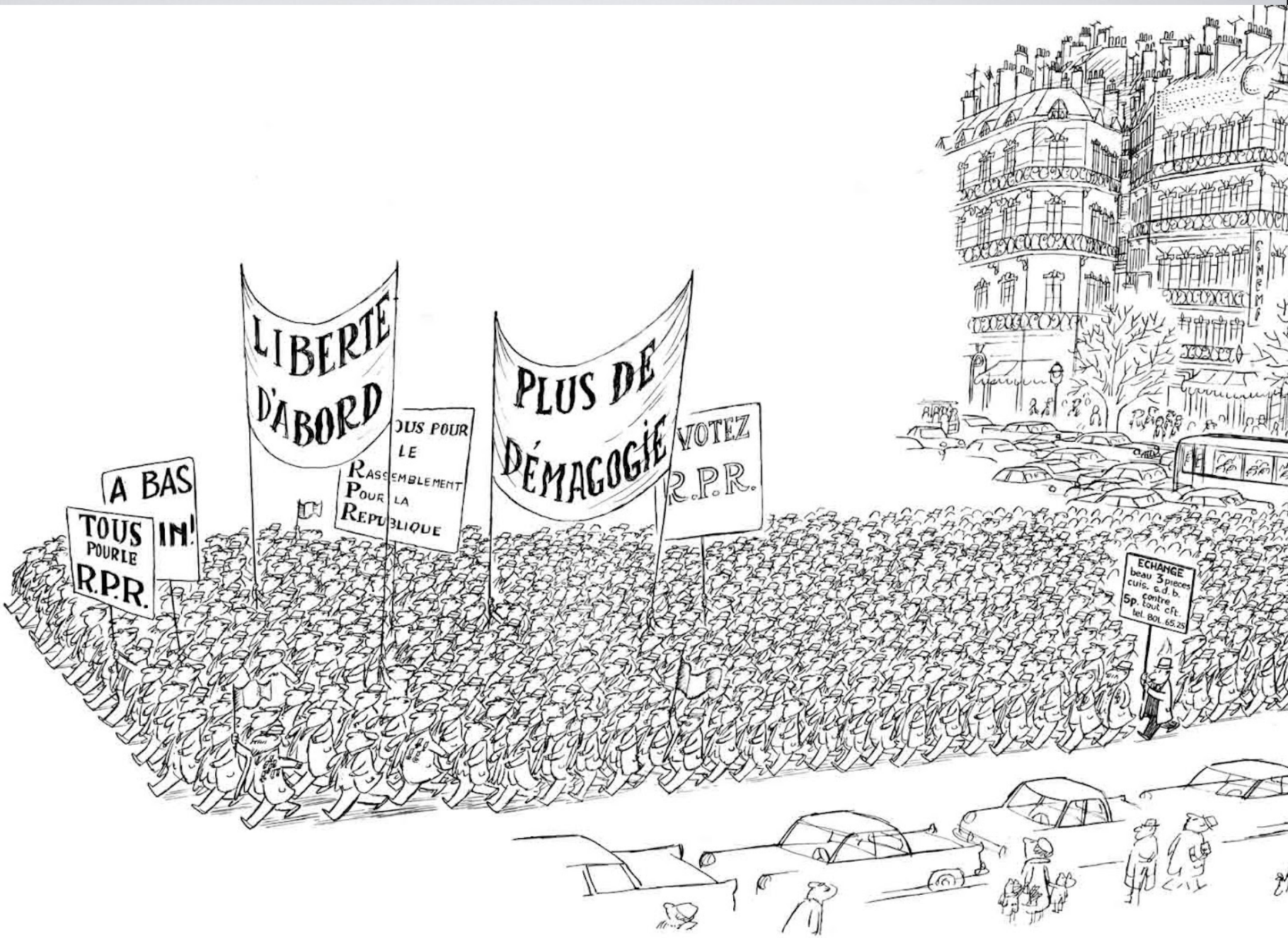
$$\text{Graph} \xrightarrow{unfold} \text{Tree}$$

MSO-compatibility of unfolding

For all Σ.
For all φ exists $\widehat{\varphi}$ s.t. for all $G \in Graph(\Sigma)$:

$$G \vDash \widehat{\varphi} \quad \text{iff} \quad Unf(G) \vDash \varphi$$

Rem: This theorem implies Rabin's Theorem.


A BAS
TOUS POUR LE R.P.R.
IN!
LIBERTÉ D'ABORD
OUS POUR
LE
RASSEMBLEMENT
POUR LA
REPUBLIQUE
PLUS DE DÉMAGOGIE
VOTEZ
R.P.R.
ECHANGE
beau 3 pieces
cuis. s.d.b.
5e. contre
tout cft.
tel. BOL. 65.25

PCF (Programming Computable Functions)

$$search \equiv \lambda p : nat \to bool.$$
$$\mathbf{letrec}\ f(x : nat) : nat = \mathbf{if}\ (px)\ \mathbf{then}\ x\ \mathbf{else}\ f(x+1)\ \mathbf{in}\ f0$$

- Proposed by Scott (1969)
- Mitchell "Foundations for Programming Languages" (1996):

  Designed to be easily analyzed, rather than practical language for writing programs. However with some syntactic sugar it is possible to write many functional programs in a comfortable style.

- PCF has been in the center of interest of semantics
  - "sequentially computable functional", parallel OR, full abstraction.

Finitary PCF: base types are finite.

$$search \equiv \lambda p : "nat" \to bool.$$
$$\textbf{letrec } f(x : "nat") : "nat" = \textbf{if } (px) \textbf{ then } x \textbf{ else } f(x+1) \textbf{ in } f0$$

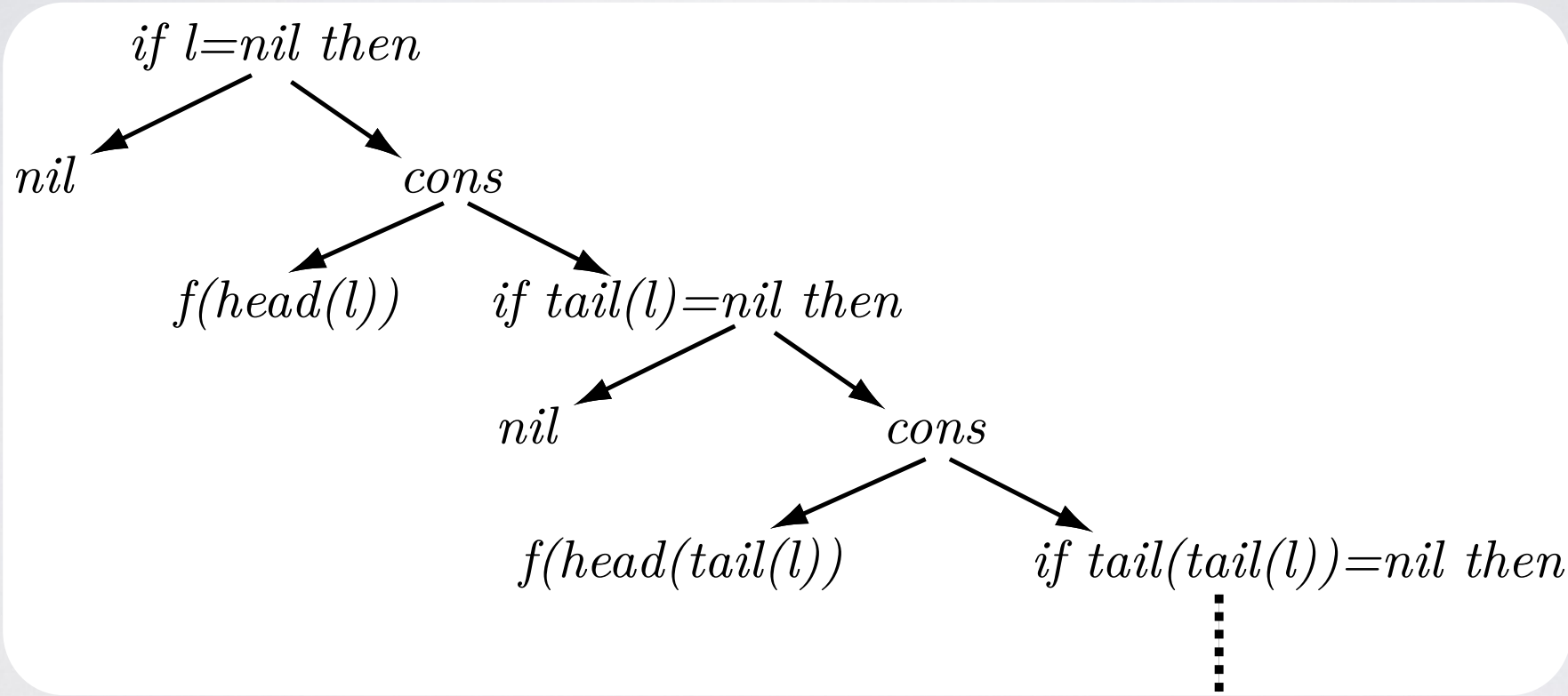
- **[Statman'04]:** $\beta\delta$-equality on terms is undecidable.
- **[Loader'96]:** There is no recursive fully-abstract model

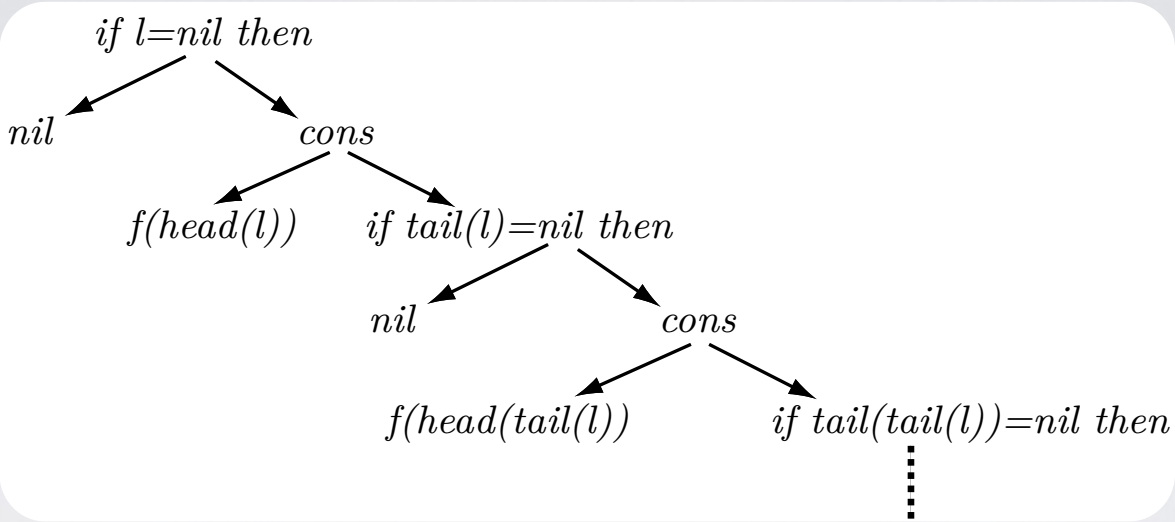
Finitary PCF $\equiv$ λY-calculus
simply-typed λ calculus with fixpoint operators.

$$map(f, l) \equiv \textbf{if } l = nil \textbf{ then } nil \textbf{ else } \textbf{cons}(f(\textbf{head}(l)), map(f, \textbf{tail}(l)))$$

$$map(f, (a, b, c)) = (f(a), f(b), f(c))$$

$$map(f, l) \equiv \textbf{if } l = nil \textbf{ then } nil$$
$$\textbf{else cons}(f(\textbf{head}(l)), map(f, \textbf{tail}(l)))$$



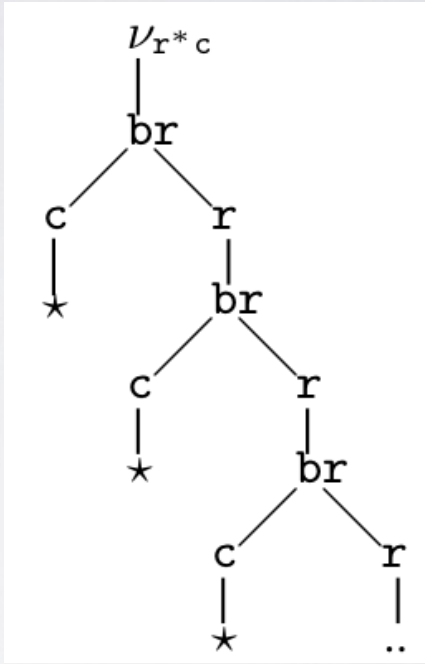


Such trees are interesting because

- They reflect a part of the semantics of a program.
- They have decidable MSOL theory.
- Interesting properties can be expressed in MSOL:
  - All elements in the result are in the range of f

Resource usage for functional programs [Kobayashi'09]

```
let rec g x = if b then close(x)
              else read(x); g(x) in
let r = open_in "foo" in g(r)
```



One can verify if usage patterns are correct.

While-programs

$$x := e \mid \texttt{if } x = 0 \texttt{ then } I_1 \texttt{ else } I_2 \mid \texttt{while } x > 0 \texttt{ do } I$$

variables range over $\mathbb{N}$ and e are arithmetic expressions

- While-programs are Turing powerful.
- Does this mean that all other programming concepts are obsolete?
- Schemes give a way to show that they are not:
  - There is a recursive scheme whose tree cannot be generated by a scheme of a while program.

Recursion ≡ stacks

Recursion $\equiv$ stacks

$$F \equiv \lambda x.\ \textbf{if}\ x = 0\ \textbf{then}\ 1\ \textbf{else}\ F(x-1) \cdot x.$$

Recursion $\equiv$ stacks

$$F \equiv \lambda x.\ \textbf{if}\ x = 0\ \textbf{then}\ 1\ \textbf{else}\ F(x-1) \cdot x.$$

Thm [Courcelle PhD]:
1-st order recursive schemes $\equiv$ deterministic pushdown automata.

Recursion $\equiv$ stacks

$$F \equiv \lambda x.\ \textbf{if}\ x = 0\ \textbf{then}\ 1\ \textbf{else}\ F(x-1) \cdot x.$$

Thm [Courcelle PhD]:
1-st order recursive schemes $\equiv$ deterministic pushdown automata.

Thm [Senizergues]:
Equivalence of 1-st order schemes (in terms of trees they generate) is decidable.

Thm [Courcelle]:
MSOL theory of trees generated by 1-st order schemes is decidable.

What about higher-order schemes?

Second-order scheme

$Map \equiv \lambda f.\lambda x.\ \textbf{if}\ x = nil\ \textbf{then}\ \text{nil}\ \textbf{else}\ f(hd(x)) \cdot Map(f, tl(x))$

Thm [Knapik, Niwiński, Urzyczyn]:
Higher-order safe schemes $\equiv$ higher-order pushdown automata

Theorem [Hague, Murawski, Ong & Serre]: n-th order schemes $\equiv$ unfoldings of n-th order collapse pushdown automata.

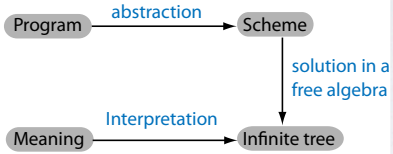
Thm [Parys]:
Safety is a true restriction

Here:

On MSO theories of trees generated by higher-order schemes
(These are also the tress generated by programs of finitary PCF).

Schemes

+ Ianov'58 "The logical schemas of algorithms"
+ Park PhD'68 Recursive schemes
+ Scott, Elgot



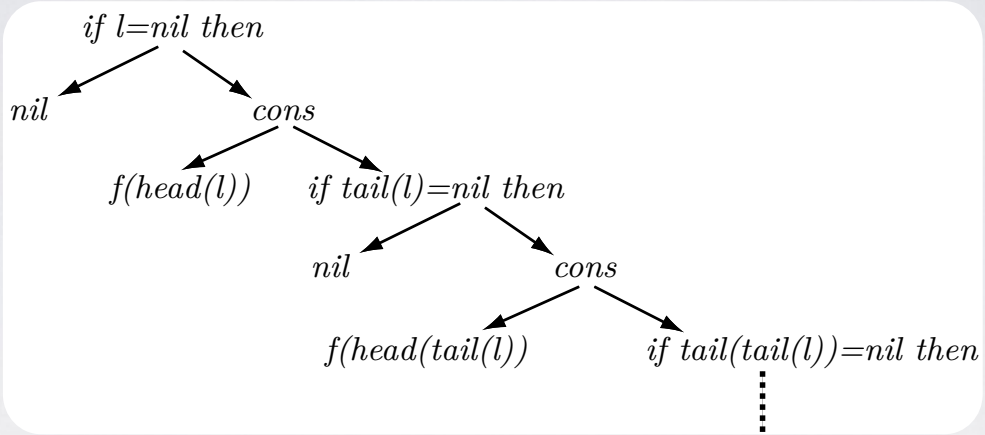
+ Milner'73 Plotkin'77 PCF

Languages, Higher-order pushdowns

+ Aho'68 indexed languages
+ Maslov'74 '76 higher-order indexed languages and higher order pushdown automata.

+ Courcelle'76 for trees: 1-st order schemes=CFL
+ Engelfriet Schmidt'77 IO/OI
+ Damm'82 for languages: rec schemes= higher-order pusdowns
+ Kanpik Niwinski Urzyczyn'02 Safe schemes = higher-order pusdown
+ Senizergues'97 Equivalence of 1st order schemes is decidable
  +Statman'04 Equivalence of PCF terms is undecidable
  +Loader'01: Lambda-definability is undecidable
+ Ong'06: Decidability of MSOL theory

Two main algorithmic problems



Deciding equality of schemes:

Do two schemes generate the same trees?

Deciding MSOL theory for schemes:

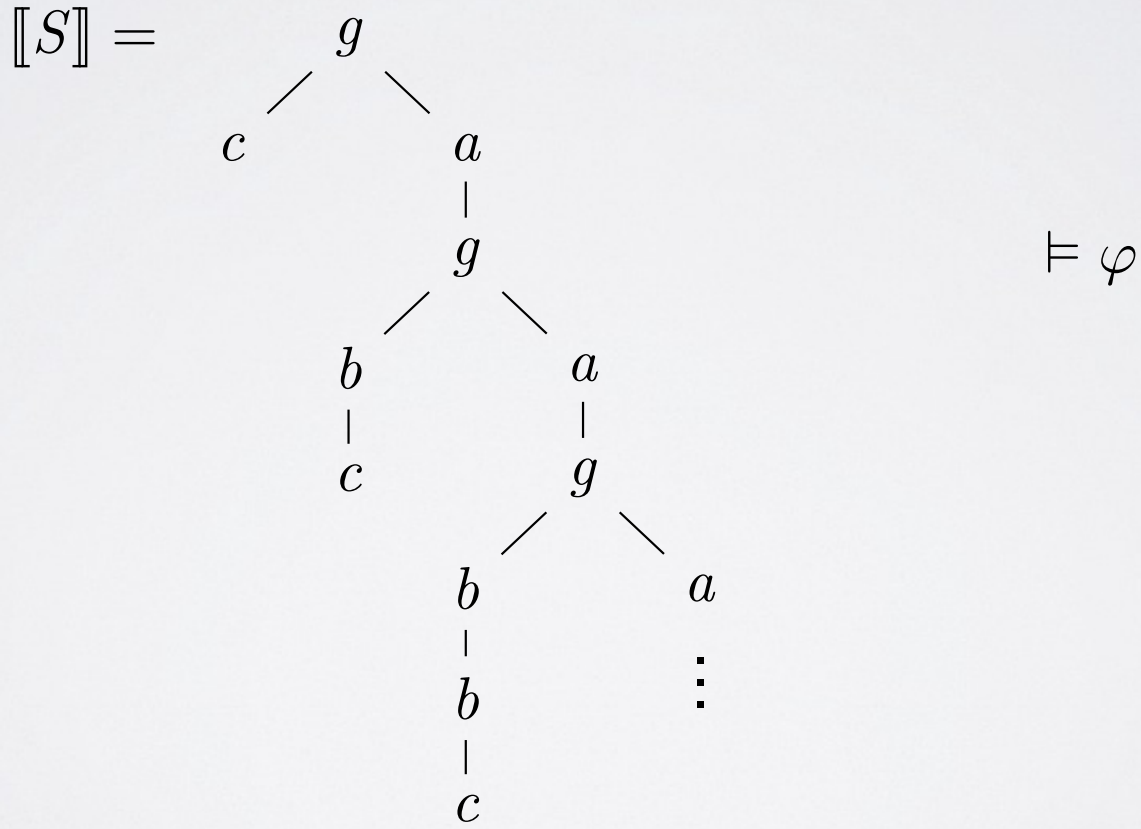
Does a given MSOL formula hold in a tree generated by a scheme?

Ad equality: Decidable for schemes of order 1 [Senizergues]

Ad MSOL: Decidable [Ong]

The model-checking problem:
Given S and an MSOL formula φ decide if $[\![S]\!] \models \varphi$.

Theorem[Ong]:
This problem is decidable.



Motivation

- Finitary PCF is an important abstraction of functional languages.
- Finitary PCF $\equiv$ schemes $\equiv \lambda Y$-calculus.
- It has been studied by semantics and language communities since 60'ties.
- The "schematological" approach to semantics gives non-trivial insights and without (sometimes) sacrificing decidability.

Objective : Understanding trees generated by PCF programs

$$M \xrightarrow{eval} BT(M)$$

$$M \xrightarrow{eval} BT(M)$$

Transfer Theorem

For all φ exists $\widehat{\varphi}$ s.t.

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi$$

$$M \xrightarrow{eval} BT(M)$$

Transfer Theorem

For all $\Sigma, \mathcal{T}, \mathcal{X}$.
For all φ exists $\widehat{\varphi}$ s.t. for all $M \in Terms(\Sigma, \mathcal{T}, \mathcal{X})$:

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi$$

Example: Unfolding

$$\text{Graph} \xrightarrow{unfold} \text{Tree}$$

MSO-compatibility of unfolding

For all Σ.
For all φ exists $\widehat{\varphi}$ s.t. for all $G \in Graph(\Sigma)$:

$$G \vDash \widehat{\varphi} \quad \text{iff} \quad Unf(G) \vDash \varphi$$

Rem: This theorem implies Rabin's Theorem.

Example: Normalizable terms

Transfer Theorem

For all φ exists $\widehat{\varphi}$ s.t. for all $M \in Terms(\Sigma, \mathcal{T}, \mathcal{X})$:

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi$$

- Take $\varphi \equiv$ "finite tree"
- $BT(M) \vDash \varphi$ iff M has a normal form.

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad M \text{ has a normal form}$$

So $\{M \in Terms(\Sigma, \mathcal{T}, \mathcal{X}) : M$ has a normal form$\}$ is MSOL-definable.

$$M \xrightarrow{eval} BT(M)$$

λY-terms

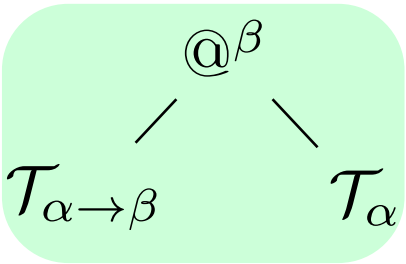
Types: 0 is a type, and $\alpha \to \beta$ is a type if α, β types.

Tree signature $\Sigma = \{a, b, \dots\}$ all constants of type $0 \to 0 \to 0$.

Terms

Ω^α, x^α, $\mathbf{x}^\alpha$, or c^α are in $\mathcal{T}_\alpha$.

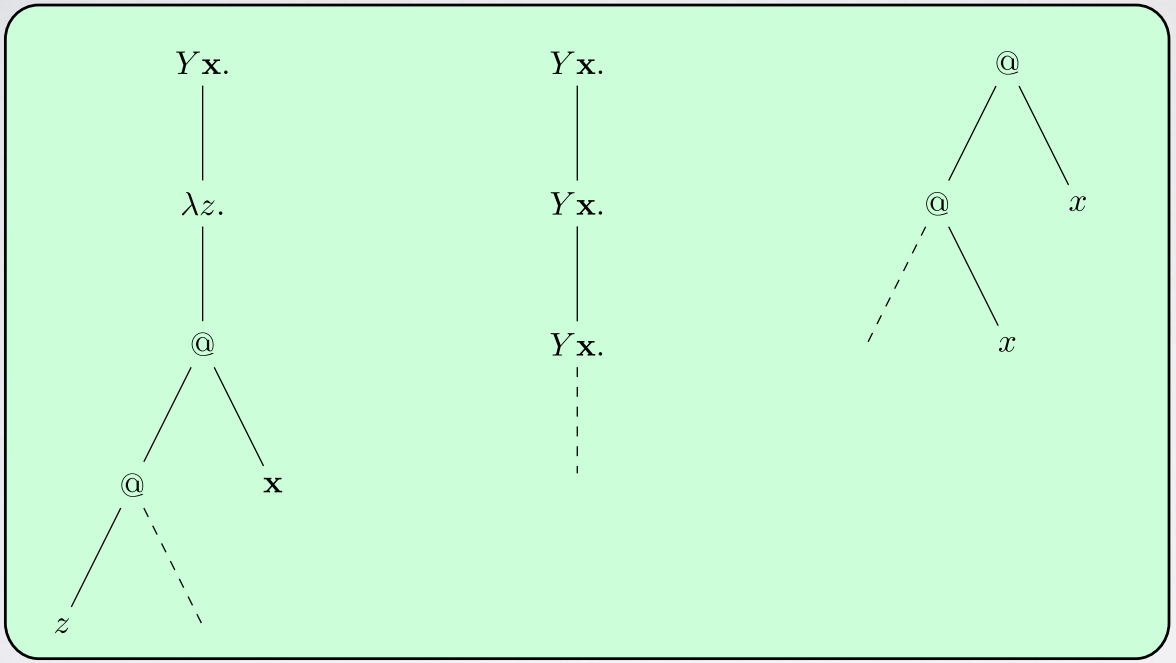
$\lambda^{\alpha\to\beta}$ over $\mathcal{T}_\beta$ is in $\mathcal{T}_{\alpha\to\beta}$.



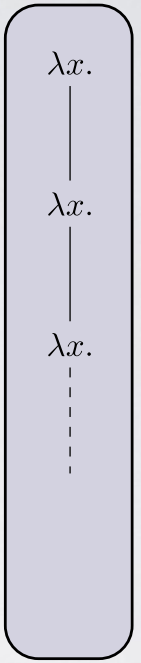
is in $\mathcal{T}_\beta$.

$Y^\alpha \mathbf{x}$ over $\mathcal{T}_\alpha$ is in $\mathcal{T}_\alpha$.

Example of terms



Malformed tree



$$M \xrightarrow{eval} BT(M)$$

Evaluation and $BT(M)$

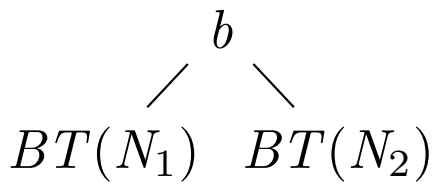
- β-reduction $(\lambda x.M)N \to_\beta M[x := N]$.
- δ-reduction $Y\mathbf{x}.M \to_\delta M[\mathbf{x} := Y\mathbf{x}.M]$.

- Head redex: $(\lambda x.P)P_0 \dots P_n$ or $(Y\mathbf{x}.P)P_1 \dots P_n$
- Weak head normal form: $hN_1 \dots N_k$ with h a variable or a constant different than Ω.

Böhm tree

Suppose that $M : 0$ over a tree signature.

- if $M \to_h^* bN_1N_2$ with $b \neq \Omega$ then $BT(M) =$

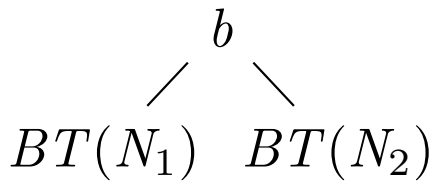


- otherwise $BT(M) = \Omega^0$.

Böhm tree

Suppose that $M : 0$ over a tree signature.

- if $M \to_h^* bN_1N_2$ with $b \neq \Omega$ then $BT(M) =$



- otherwise $BT(M) = \Omega^0$.

Example 1

- $Y\mathbf{x}.a\mathbf{x}$
- Every tree generated by a recursive scheme is $BT(M)$ for some M.

Example 2 (QBF)

- tt $= \lambda xy.\ x$, ff $= \lambda xy.\ y$, They are of type $0 \to 0 \to 0$.
- and $= \lambda b_1\, b_2\, xy.\ b_1(b_2\, xy)y$, or $= \lambda b_1\, b_2\, xy.\ b_1\, x(b_2\, xy)$,
- neg $= \lambda bxy.\ byx$
- All $= \lambda f.$ and$(f$ tt$)(f$ ff$)$, Exists $= \lambda f.$ or$(f$ tt$)(f$ ff$)$.

QBF to terms

Every QBF formula α can be translated to a term M_α:

$$\forall x.\exists y.\ x \wedge \neg y \quad \mapsto \quad \text{All}(\lambda x.\ \text{Exists}(\lambda y.\ \text{and}\ x\ (\text{neg}\ y)))$$

Fact For every QBF sentence α:

$$\alpha \text{ is true} \quad \text{iff} \quad M_\alpha \text{ evaluates to tt.}$$

$$M \xrightarrow{eval} BT(M)$$

Transfer Theorem

For all $\Sigma, \mathcal{T}, \mathcal{X}$.
For all φ exists $\widehat{\varphi}$ s.t. for all $M \in Terms(\Sigma, \mathcal{T}, \mathcal{X})$:

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi$$

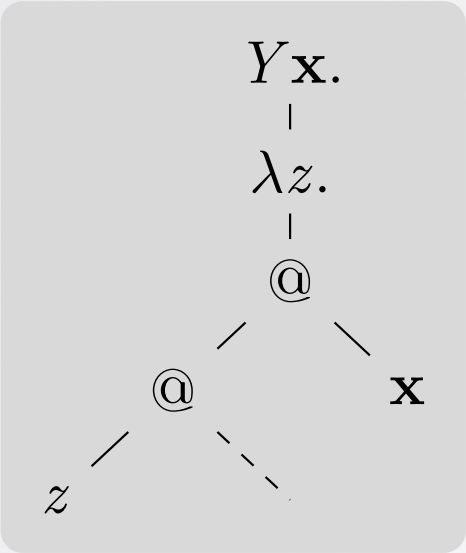
- Σ is a tree signature
- $\mathcal{T}$ is a finite set of terms
- $\mathcal{X}$ is a finite set of λ-variables
- $Terms(\Sigma, \mathcal{T}, \mathcal{X})$: terms over Σ with
  - all subterms having type in $\mathcal{T}$,
  - all λ-variables from $\mathcal{X}$.

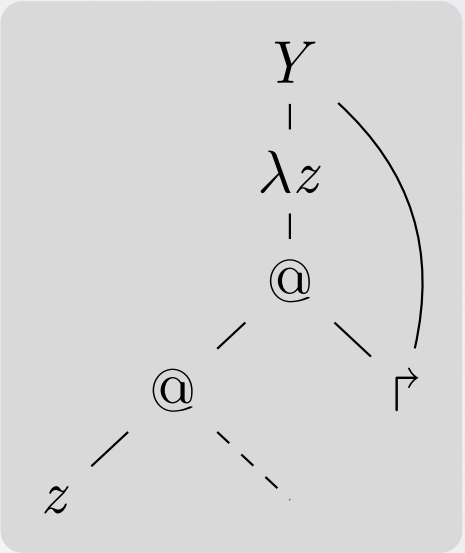
Note: no limitation on Y variables.

What it means $M \models \hat{\varphi}$?

M is represented as a graph $Graph(M)$ over the alphabet

$$\begin{aligned} Talph(\Sigma, \mathcal{T}, \mathcal{X}) \quad = \quad & \Sigma \cup \{@^\alpha, Y^\alpha, \curvearrowright^\alpha : \alpha \in \mathcal{T}\} \cup \mathcal{X} \cup \\ & \{\lambda^{\alpha \to \beta} x^\alpha : \alpha \in \mathcal{T} \wedge \alpha \to \beta \in \mathcal{T} \wedge x^\alpha \in \mathcal{X}\} \, . \end{aligned}$$





Transfer Thm: For all φ exists $\widehat{\varphi}$ s.t. for all $M \in Terms(\Sigma, \mathcal{T}, \mathcal{X})$:

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi$$

Consequences of the transfer theorem

Ong's Theorem

It is decidable if for a given finite term M and MSOL formula φ, $BT(M) \models \varphi$ holds.

Proof: Just test $M \models \widehat{\varphi}$.

The set of normalizing terms is MSOL definable

For a fixed $\mathcal{T}$ and $\mathcal{X}$ there is a formula defining the set of terms $M \in Terms(\Sigma, \mathcal{T}, \mathcal{X})$ having a normal form.

Proof: Take φ defining the set of finite trees and consider $\widehat{\varphi}$.

Digression: why limiting λ-variables

QBF to terms

Every QBF formula α can be translated to a term M_α:

$$\forall x.\exists y.\ x \wedge \neg y \quad \mapsto \quad All(\lambda x.\ Exists(\lambda y.\ and\ x\ (neg\ y)))$$

α is true iff $BT(M_\alpha)$ is the term *true*

Take φ saying that the tree consists only of the root labeled $true$.
Consider $\widehat{\varphi}$.

$$M_\alpha \vDash \widehat{\varphi} \qquad \text{iff} \qquad \alpha \text{ is true.}$$

If we could construct $\widehat{\varphi}$ without limiting $\mathcal{X}$ then we get collapse of the polynomial hierarchy.

Matching with restricted no of variables

For a fixed $\mathcal{X}$. Given M and K (without fixpoints) decide if there is a substitution σ such that

$$M\sigma =_\beta K$$

Substitution Σ can use only terms from $Terms(\Sigma, \mathcal{T}, \mathcal{X})$.

Proof:

- Let $shape(N)$ be MSOL formula defining the set of terms in $Terms(\Sigma, \mathcal{T}, \mathcal{X})$ that can be obtained from N by substitutions.
- Let $\varphi \equiv shape(K)$.
- There is desired σ iff the formula $shape(M) \wedge \widehat{\varphi}$ is satisfiable.

If there is a solution then there is a finite one.

Synthesis from modules

Given finite λY-terms $M_1, \ldots, M_k$ and φ can one construct a λY term K from these terms such that $BT(M) \models \varphi$.

Proof:

- The candidate term K can be described as having the form $(\lambda x_1 \ldots x_k.\ N)M_1, \ldots, M_k$ for some term N without constants and λ-abstractions.
- Let ψ be a formula defining terms of this form.
- There is a solution iff the formula $\psi \wedge \widehat{\varphi}$ is satisfiable.

Every model of $\psi \wedge \widehat{\varphi}$ gives a solution.

If there is a solution then there is a regular one, hence a finite one thanks to the presence of Y.

Transfer Thm: For all φ exists $\widehat{\varphi}$ s.t. for all $M \in Terms(\Sigma, \mathcal{T}, \mathcal{X})$:

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi$$

A sketch of the proof

φ	$BT(M) \models \varphi$		$M \models F^{-1}(\gamma_{win})$	M
$\downarrow$	iff		iff	$\downarrow F$
$\mathcal{A}$	$BT(M) \in L(\mathcal{A})$		$G(\mathcal{A}, M) \models \gamma_{win}$	$G(\mathcal{A}, M)$
	iff		iff	
	Eve wins in $\mathcal{K}(\mathcal{A}, M)$	iff	Eve wins in $G(\mathcal{A}, M)$	

M is in a canonical form if

no subterm $Y\mathbf{x}.\ N$ of M has free λ-variables.

$$Y\mathbf{x}.\ N \quad \mapsto \quad (Y\mathbf{y}.\lambda x_1 \ldots \lambda x_n.N[\mathbf{x} := \mathbf{y}x_1 \ldots x_n])x_1 \ldots x_n$$

Krivine machine

- Closure $C ::= (N, \rho)$
- Environment $\rho ::= \emptyset \mid \rho[x \mapsto C]$

Initial term

$$M = (\,Y\mathbf{x}^{0\to 0}.\lambda y^0.b(\mathbf{x}y)y)(\,Y\mathbf{z}^0.c\mathbf{z}\mathbf{z})$$

A closure

$$C = (b(\mathbf{x}y^0), [y^0 \leftarrow (\,Y\mathbf{z}^0.c\mathbf{z}\mathbf{z}, \emptyset)])$$

Expansion of the closure

$$E(C) = b(\mathbf{x}y^0)[y := E(\,Y\mathbf{z}^0.c\mathbf{z}\mathbf{z}, \emptyset)]$$

Krivine machine (2)

A configuration of a Krivine machine is a triple (N, ρ, S) where:

- N is a term (a subterm of M);
- ρ is an environment defined for all free variables of N;
- S is a stack $C_1 \dots C_k$, where k and the types of the closures are determined by the type of N: the type of C_i is α_i where the type of N is $\alpha_1 \to \dots \to \alpha_k \to 0$.

A configuration (N, ρ, S) represents an infinitary term:

$$E((N, \rho, S)) = E(N, \rho)E(C_1) \dots E(C_n)$$

Example: $(b(\mathbf{x}y^0),\ [y^0 \leftarrow Y\mathbf{z^0}.c\mathbf{zz}],\ (y^0, [y^0 \leftarrow Y\mathbf{z}^0.c\mathbf{zz}]))$

Fix a canonical term M. Every Y-variable bound at most once in M. So $term(\mathbf{x})$ is the subterm $Y\mathbf{x}.N$ of M. (It has not free λ-vars).

Krivine machine

$$
\begin{aligned}
(\lambda x.N, \rho, (K, \rho')S) &\to (N, \rho[x \mapsto (K, \rho')], S) \\
(Y\mathbf{x}.N, \rho, S) &\to (N, \rho, S) \\
(NK, \rho, S) &\to (N, \rho, (K, \rho)S) \\
(x, \rho, S) &\to (N, \rho', S) \quad \text{where } (N, \rho') = \rho(x) \\
(\mathbf{x}, \rho, S) &\to (term(\mathbf{x}), \emptyset, S)
\end{aligned}
$$

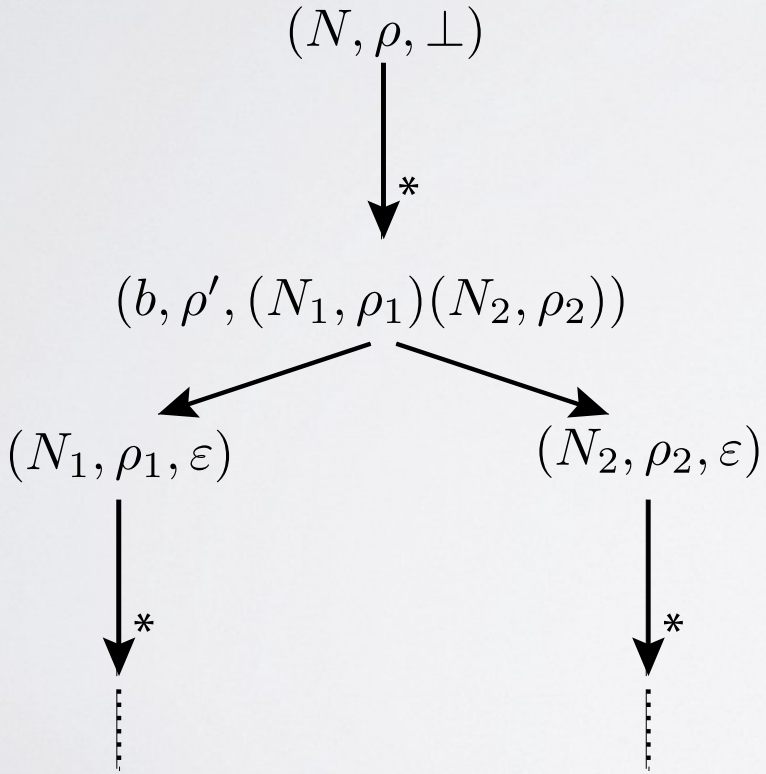
Lemma

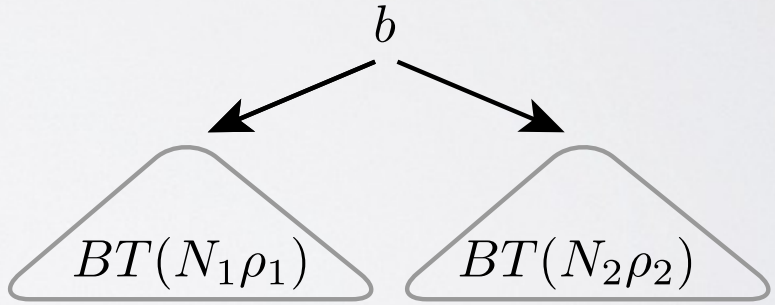
Term $E(N, \rho, \bot)$ has a head normal form iff Krivine machine reduces $(N, \rho, \bot)$ to a (b, ρ, S) for some constant $b \neq \Omega$.

Computing Böhm tree

Lemma

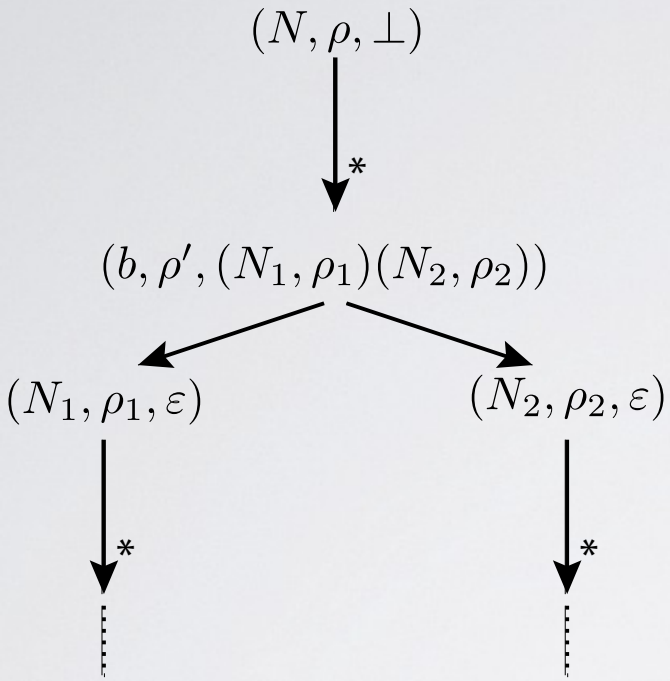
Term $E(N, \rho, \bot)$ has a head normal form iff Krivine machine reduces $(N, \rho, \bot)$ to a (b, ρ, S) for some constant $b \neq \Omega$.

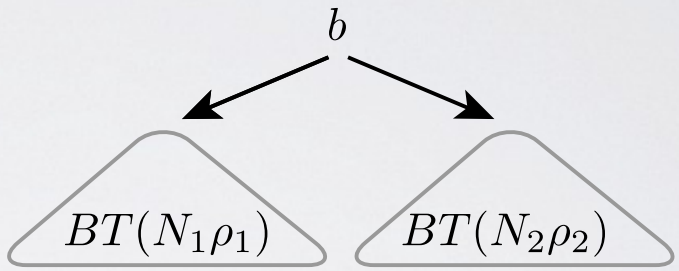




$Ktree(N, \rho, \bot) = BT(N\rho)$

Computing Böhm tree





$Ktree(N, \rho, \bot) = BT(N\rho)$

Theorem

For every concrete canonical and closed λY-term M of type 0:

$$BT(M) = KT(M).$$

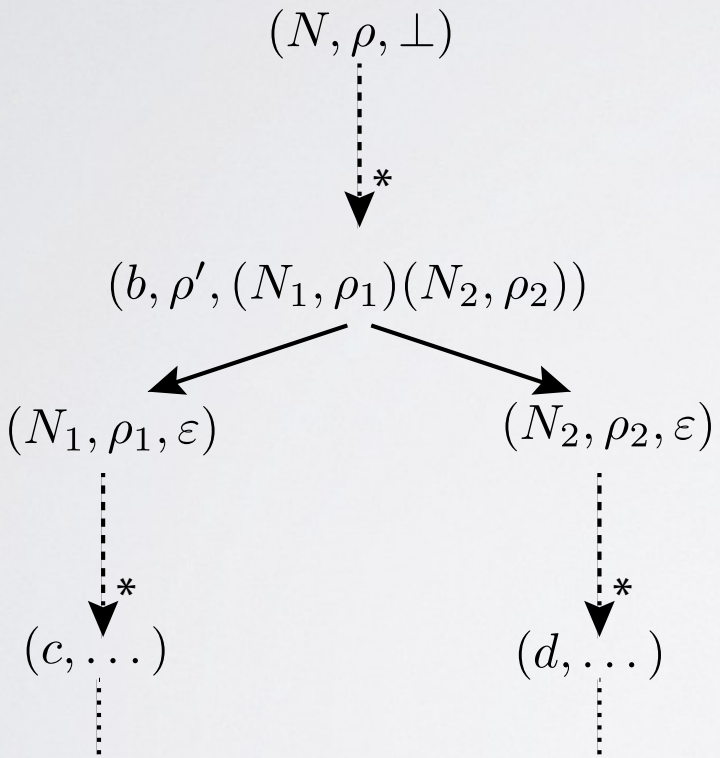
All the terms appearing in configurations of the Krivine machine during the computation of $KT(M)$ are subterms of M.

A sketch of the proof

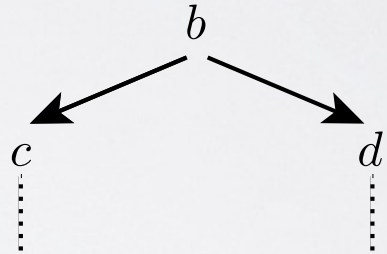
φ	$BT(M) \models \varphi$	$M \models F^{-1}(\gamma_{win})$	M
$\downarrow$	iff	iff	$\downarrow F$
$\mathcal{A}$	$BT(M) \in L(\mathcal{A})$	$G(\mathcal{A}, M) \models \gamma_{win}$	$G(\mathcal{A}, M)$
	iff	iff	
	Eve wins in $\mathcal{K}(\mathcal{A}, M)$	iff Eve wins in $G(\mathcal{A}, M)$	

Defining $\mathcal{K}(\mathcal{A}, M)$

Krivine machine comptuation

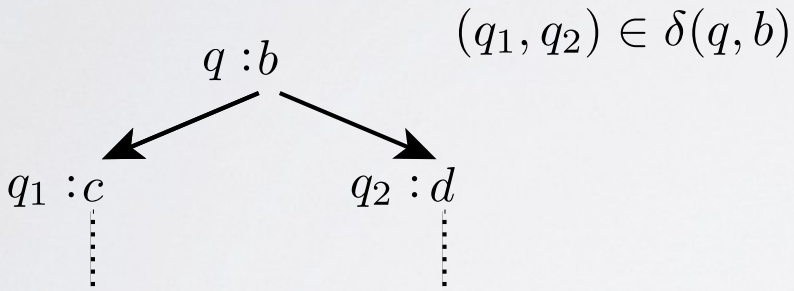


The resulting Bohm tree

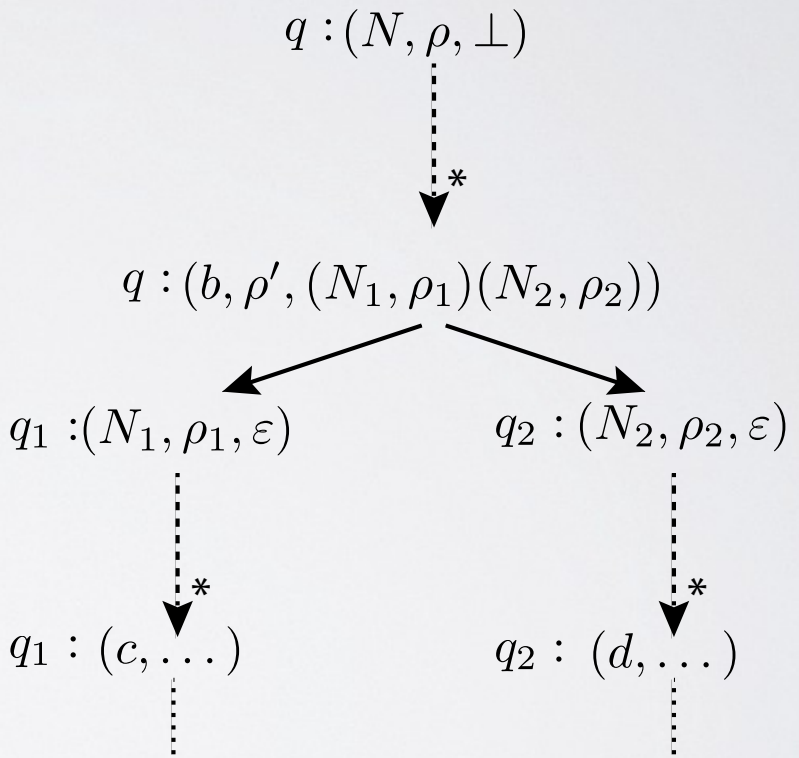


Defining $\mathcal{K}(\mathcal{A}, M)$

Run of the automaton on the Bohm tree

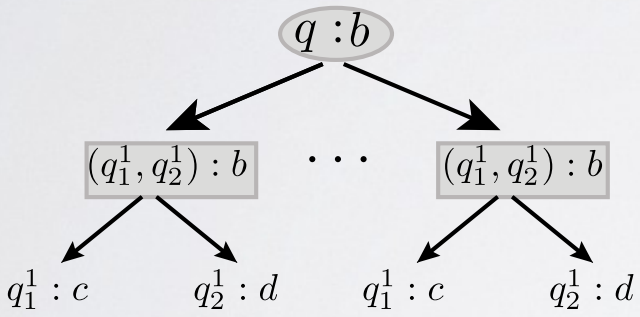


Run of the automaton on Krivine machine comptuation



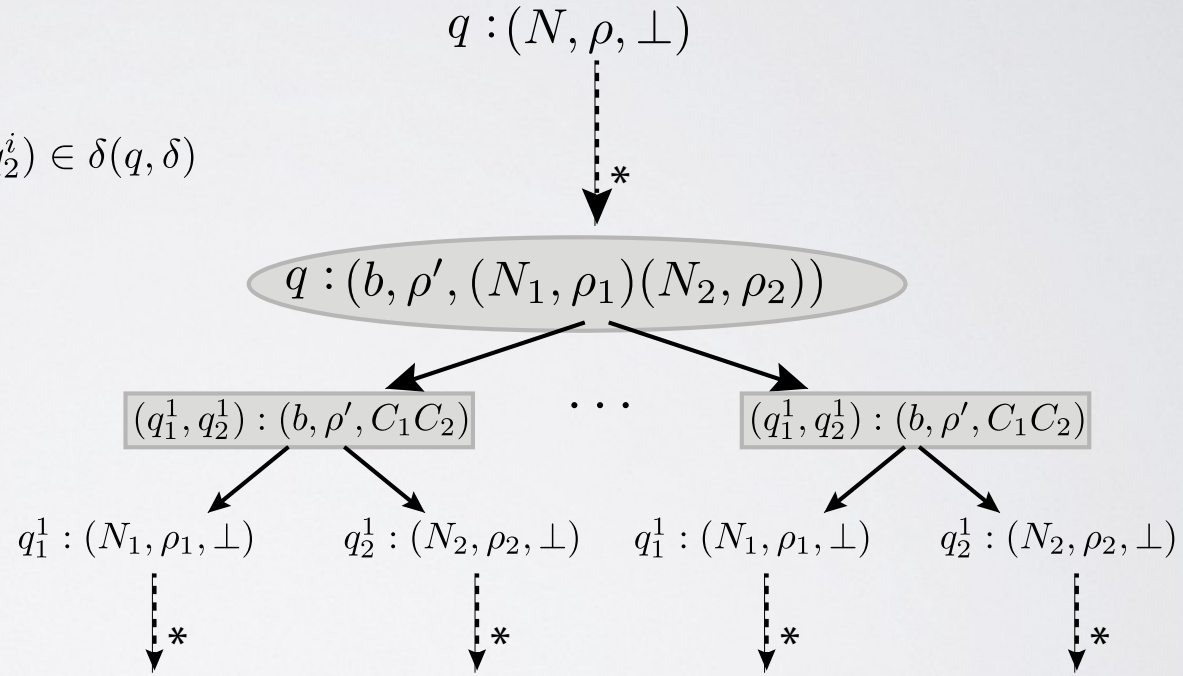
Defining $\mathcal{K}(\mathcal{A}, M)$

Acceptance of the automaton in terms of a game on the Bohm tree



for all $(q_1^i, q_2^i) \in \delta(q, \delta)$

Acceptance of the automaton in terms of a game on Krivine machine comptuation



Definition of $\mathcal{K}(\mathcal{A}, M)$

- The root is $q^0 : (M, \emptyset, \bot)$
- A node $q : (a, \rho, S)$ has a successor $(q_0, q_1) : (a, \rho, S)$ for every $(q_0, q_1) \in \delta(q, a)$.
- A node $(q_0, q_1) : (a, \rho, (v_0, N_0, \rho_0)(v_1, N_1, \rho_1))$ successors $q_0 : (N_0, \rho_0, \bot)$ and $q_1 : (N_1, \rho_1, \bot)$.

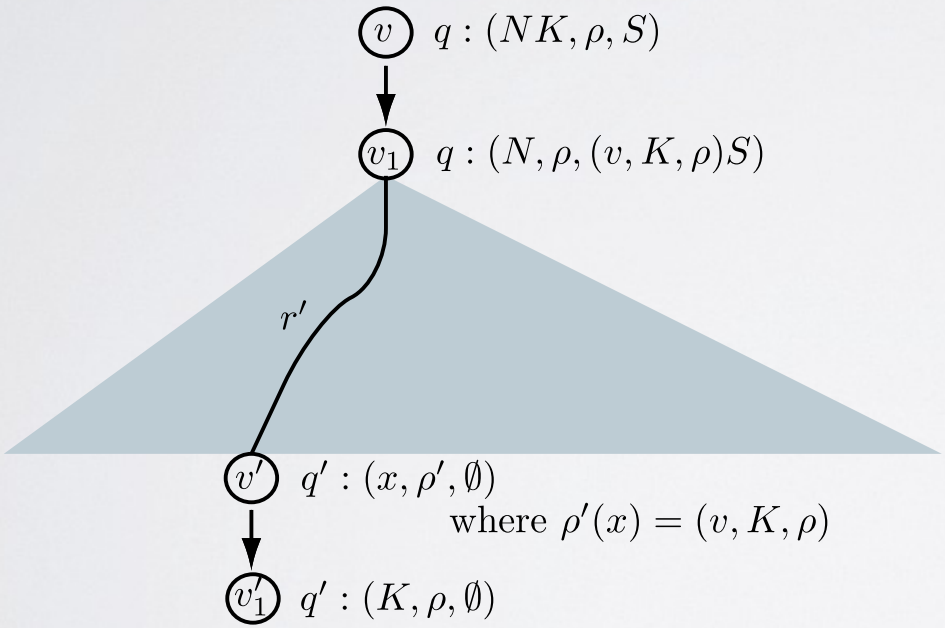
Definition of $\mathcal{K}(\mathcal{A}, M)$

- The root is $q^0 : (M, \emptyset, \bot)$
- A node $q : (a, \rho, S)$ has a successor $(q_0, q_1) : (a, \rho, S)$ for every $(q_0, q_1) \in \delta(q, a)$.
- A node $(q_0, q_1) : (a, \rho, (v_0, N_0, \rho_0)(v_1, N_1, \rho_1))$ successors $q_0 : (N_0, \rho_0, \bot)$ and $q_1 : (N_1, \rho_1, \bot)$.
- A node $q : (\lambda x.N, \rho, CS)$ has a successor $q : (N, \rho[x \mapsto C], S)$.
- A node $q : (Y\mathbf{x}.N, \rho, S)$ has a successor $q : (N, \rho, S)$.
- A node $q : (\mathbf{x}, \rho, S)$, for $\mathbf{x}$ a recursive variable, has a successor $q : (term(\mathbf{x}), \emptyset, S)$.

Definition of $\mathcal{K}(\mathcal{A}, M)$

- The root is $q^0 : (M, \emptyset, \bot)$
- A node $q : (a, \rho, S)$ has a successor $(q_0, q_1) : (a, \rho, S)$ for every $(q_0, q_1) \in \delta(q, a)$.
- A node $(q_0, q_1) : (a, \rho, (v_0, N_0, \rho_0)(v_1, N_1, \rho_1))$ successors $q_0 : (N_0, \rho_0, \bot)$ and $q_1 : (N_1, \rho_1, \bot)$.
- A node $q : (\lambda x.N, \rho, CS)$ has a successor $q : (N, \rho[x \mapsto C], S)$.
- A node $q : (Y\mathbf{x}.N, \rho, S)$ has a successor $q : (N, \rho, S)$.
- A node $q : (\mathbf{x}, \rho, S)$, for $\mathbf{x}$ a recursive variable, has a successor $q : (term(\mathbf{x}), \emptyset, S)$.
- A node v labeled $q : (NK, \rho, S)$ has a unique successor labeled $q : (N, \rho, (v, K, \rho)S)$. v-closure is *created*.
- A node v labeled $q : (x, \rho, S)$, for x a λ-variable and $\rho(x) = (v', N, \rho')$, has a unique successor labeled $q : (N, \rho', S)$. Node v *uses* a v'-closure.

- A node v labeled $q : (NK, \rho, S)$ has a unique successor labeled $q : (N, \rho, (v, K, \rho)S)$. v-closure is *created*.
- A node v labeled $q : (x, \rho, S)$, for x a λ-variable and $\rho(x) = (v', N, \rho')$, has a unique successor labeled $q : (N, \rho', S)$. Node v *uses* a v'-closure.

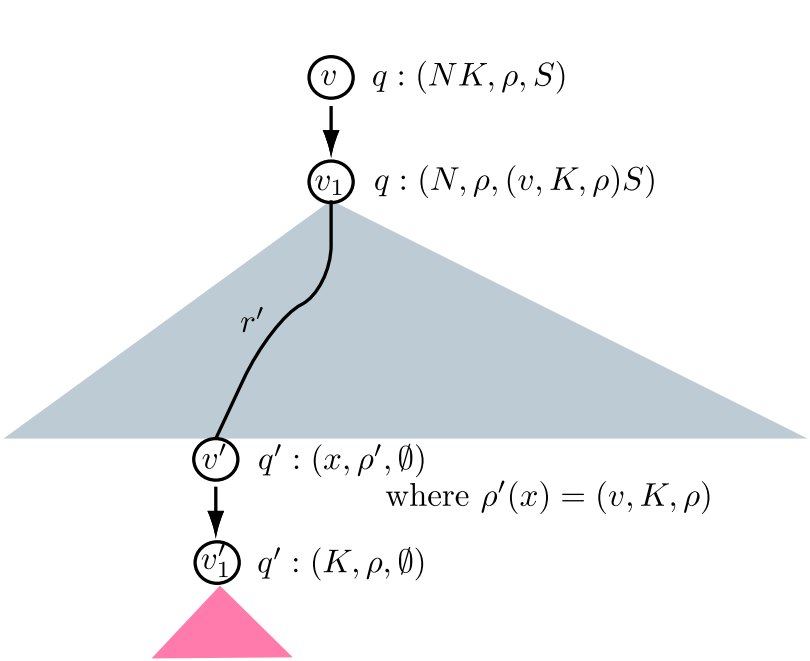


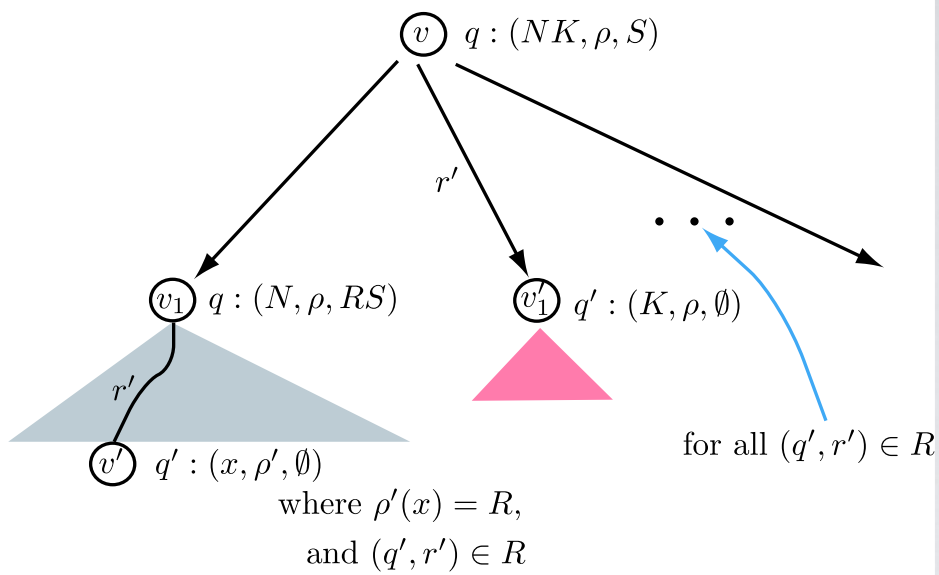
Thm: Eve wins in $\mathcal{K}(\mathcal{A}, M)$ iff $\mathcal{A}$ accepts $BT(M)$.

A sketch of the proof

φ	$BT(M) \models \varphi$		$M \models F^{-1}(\gamma_{win})$	M
$\downarrow$	iff		iff	$\downarrow F$
$\mathcal{A}$	$BT(M) \in L(\mathcal{A})$		$G(\mathcal{A}, M) \models \gamma_{win}$	$G(\mathcal{A}, M)$
	iff		iff	
	Eve wins in $\mathcal{K}(\mathcal{A}, M)$	iff	Eve wins in $G(\mathcal{A}, M)$	

From $\mathcal{K}(\mathcal{A}, M)$ to $G(\mathcal{A}, M)$





- Residual of type 0 is from $\mathcal{P}(Q \times [d])$.
- Residual of type $0 \to 0$ is from $\mathcal{P}(Q \times [d]) \to \mathcal{P}(Q \times [d])$.

$G(\mathcal{A}, M)$

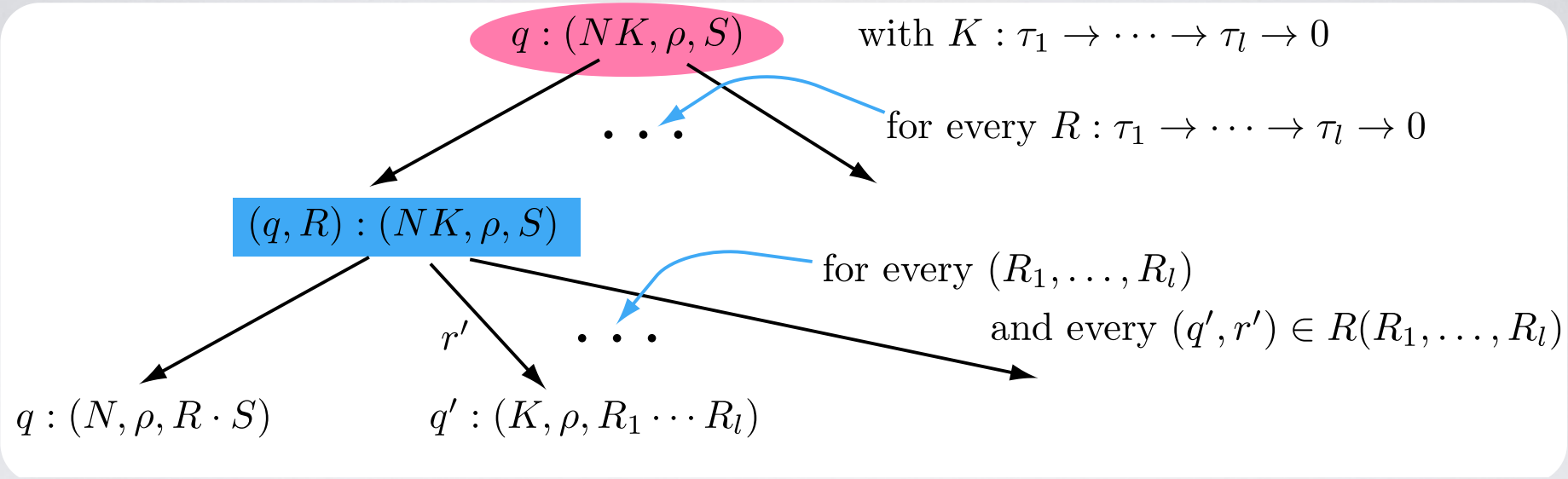
$$q : (\lambda x.N, \rho, R \cdot S) \to q : (N, \rho[x \mapsto R], S)$$

$$q : (a, \rho, R_0 R_1) \to (q_0, q_1) : (a, \rho, R_0 R_1) \qquad \text{for } (q_0, q_1) \in \delta(q, a)$$

$$q : (Y\mathbf{x}.N, \rho, S) \to q : (N, \rho, S)$$

$$q : (\mathbf{x}, \rho, S) \to q : (term(\mathbf{x}), \rho, S) \qquad \mathbf{x} \text{ a recursion variable}$$

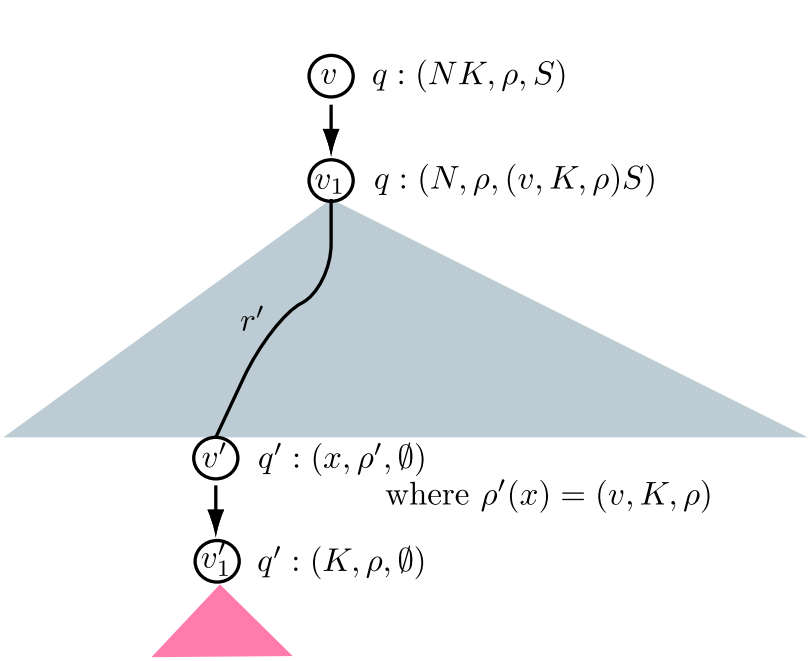
$G(\mathcal{A}, M)$

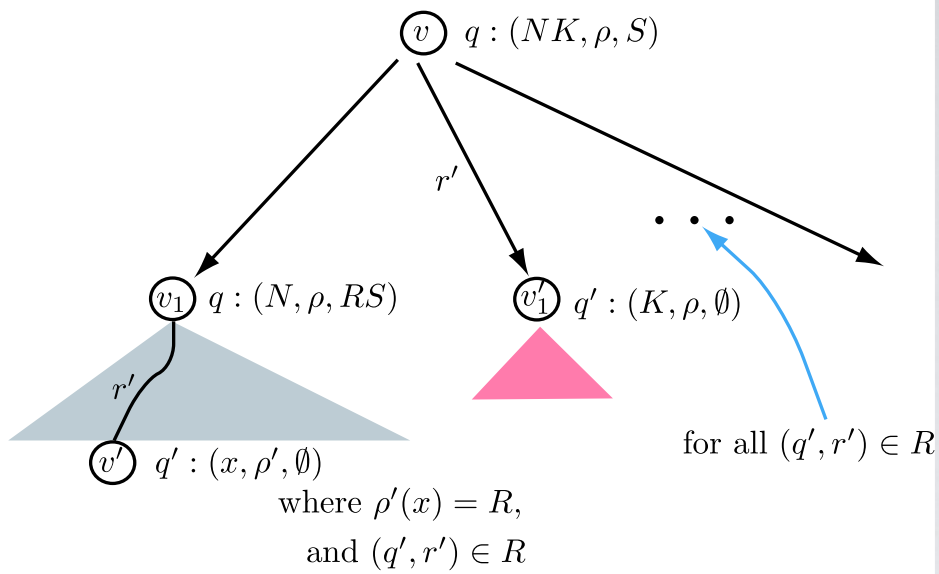


Eve wins in a position:

- $q : (x, \rho, S)$
  if $(q, rk(q)) \in R_x(R_1, \ldots, R_k)$; where $\rho(x) = R_x$ and $S = R_1 \cdots R_k$.
- $(q_0, q_1) : (a, \rho, R_0 R_1)$
  if $(q_0, rk(q_0)) \in R_0 \restriction_{rk(q_0)}$ and $(q_1, rk(q_1)) \in R_1 \restriction_{rk(q_1)}$.

Properties of $G(\mathcal{A}, M)$





Thm: Eve wins in $G(\mathcal{A}, M)$ iff Eve wins in $\mathcal{K}(\mathcal{A}, M)$.

Obs: For every N there are finitely many nodes in $G(\mathcal{A}, M)$ containing N.

A sketch of the proof

φ	$BT(M) \models \varphi$		$M \models F^{-1}(\gamma_{win})$	M
$\downarrow$	iff		iff	$\downarrow F$
$\mathcal{A}$	$BT(M) \in L(\mathcal{A})$		$G(\mathcal{A}, M) \models \gamma_{win}$	$G(\mathcal{A}, M)$
	iff		iff	
	Eve wins in $\mathcal{K}(\mathcal{A}, M)$	iff	Eve wins in $G(\mathcal{A}, M)$	

Prop: $G(\mathcal{A}, M)$ is definable from in $Graph(M)$ by means of MSOL-transduction.

$$M \xrightarrow{eval} BT(M)$$

Transfer Theorem

For all φ exists $\widehat{\varphi}$ s.t. for all $M \in Terms(\Sigma, \mathcal{T}, \mathcal{X})$:

$$M \vDash \widehat{\varphi} \quad \text{iff} \quad BT(M) \vDash \varphi$$