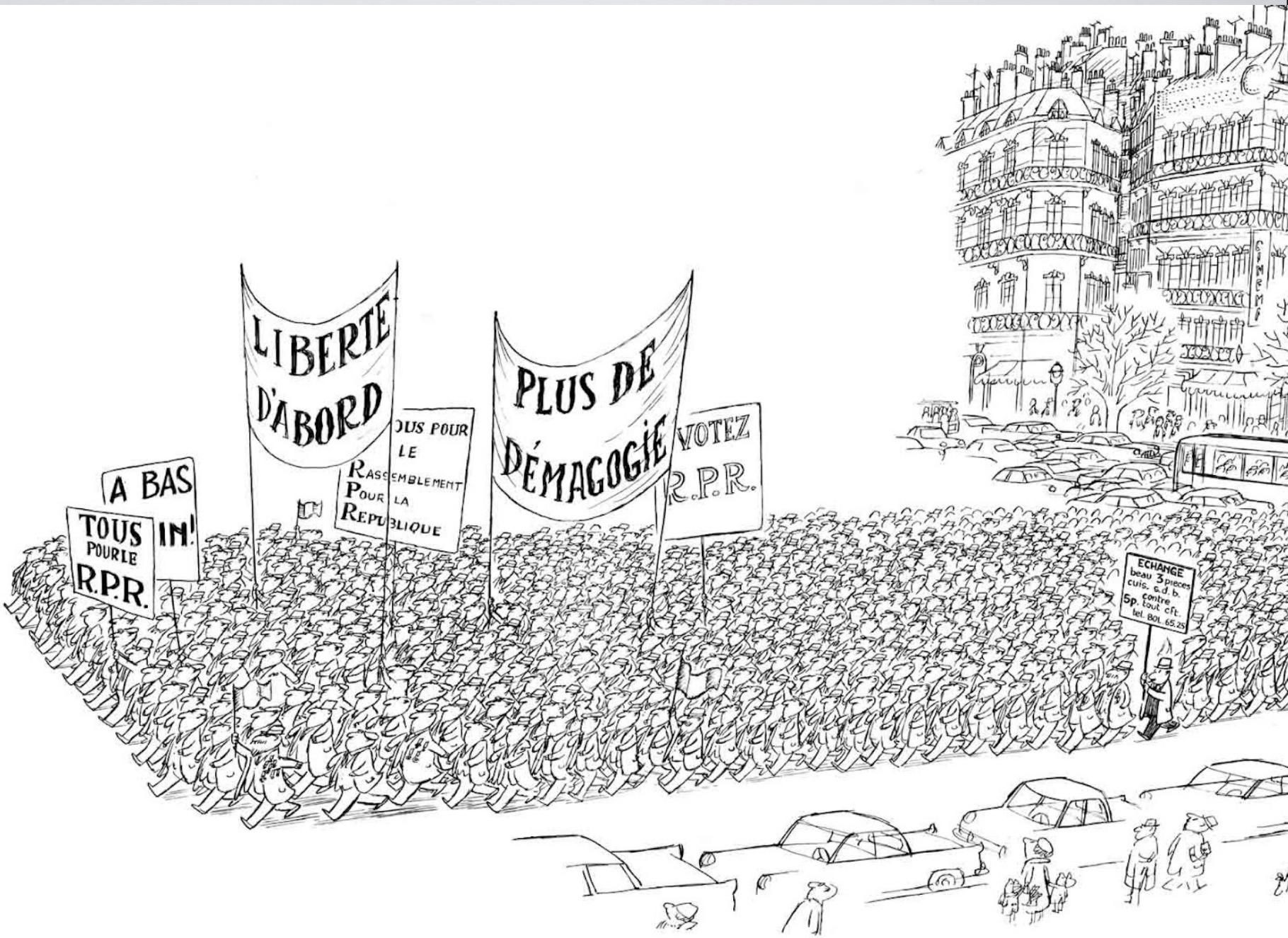
Recursive Schemes, Krivine Machines, and Monadic Logic

Igor Walukiewicz

Bordeaux University

Joint work with Sylvain Salvati


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.

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

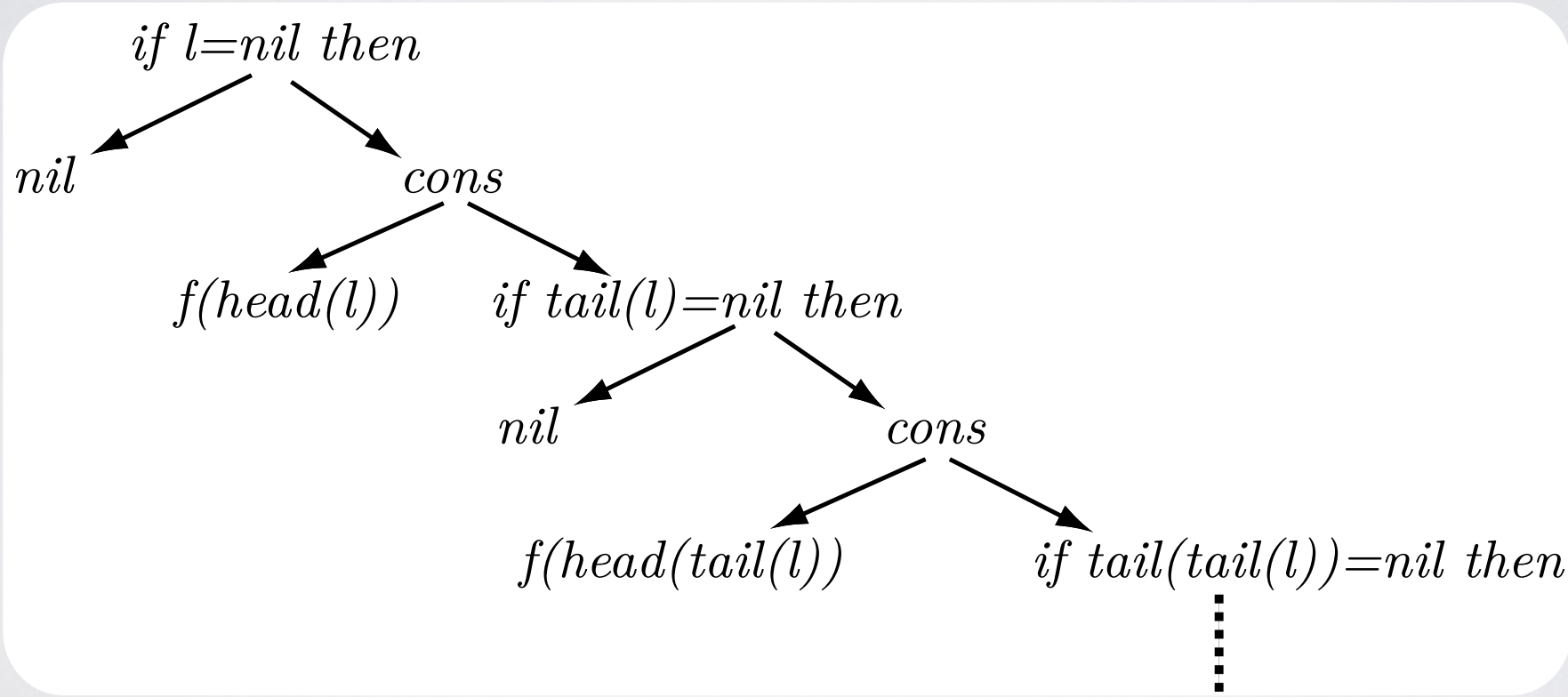
- **[Statman'04]:** $\beta\delta$-equality on terms is undecidable.
- **[Loader'96]:** There is no recursive fully-abstract model

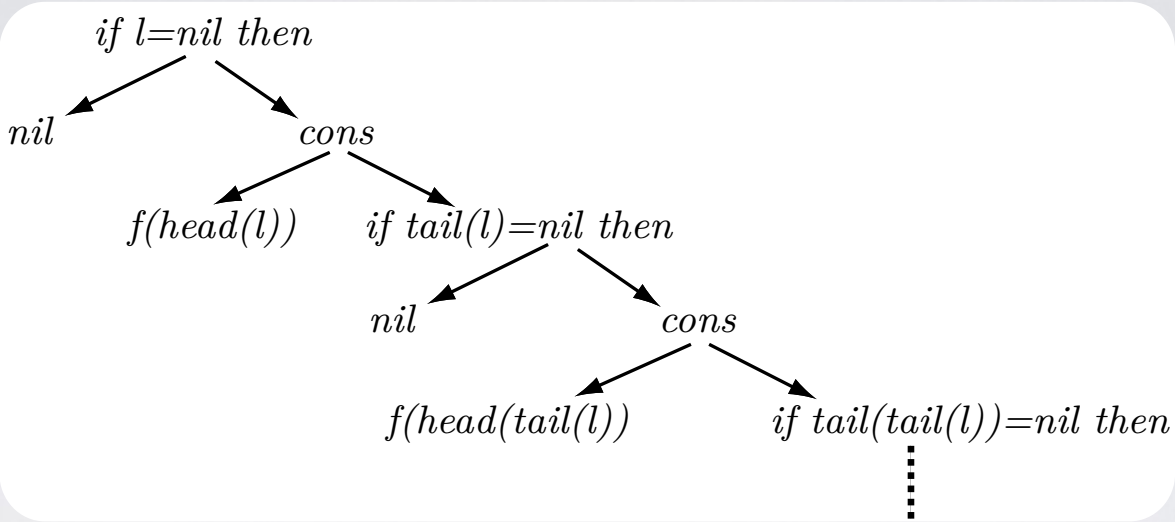
Finitary PCF $\equiv$ λY-calculus
simply-typed λ calculus with fixpoint operators.

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

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

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



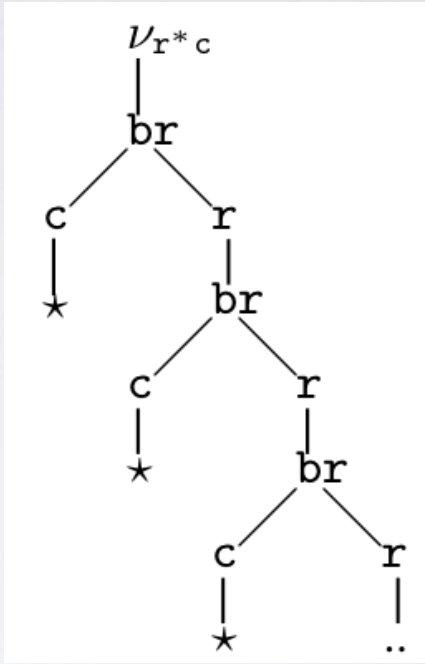


Such trees are interesting because

- 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 ≡ stacks

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

Recursion $\equiv$ stacks

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

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.

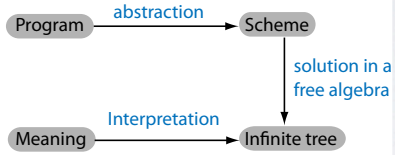
Thm [Parys]:
Safety is a true restriction

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



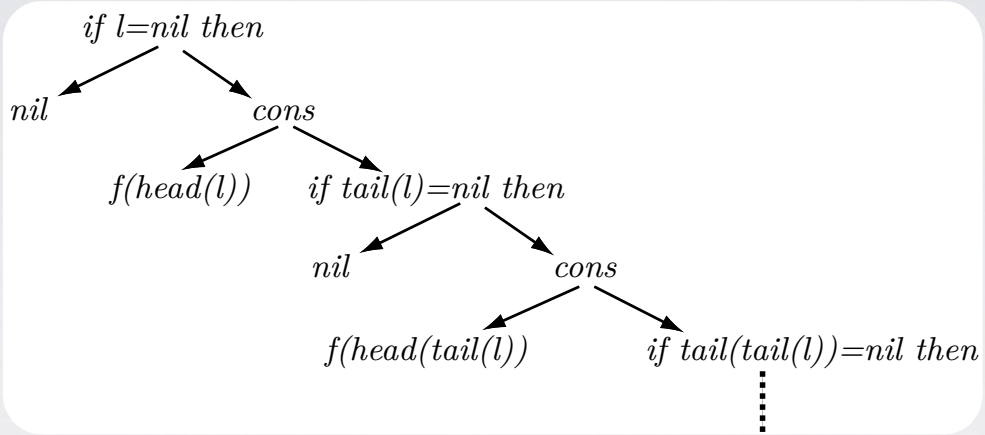
+ Milner'73 Plotkin'77 PCF

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:

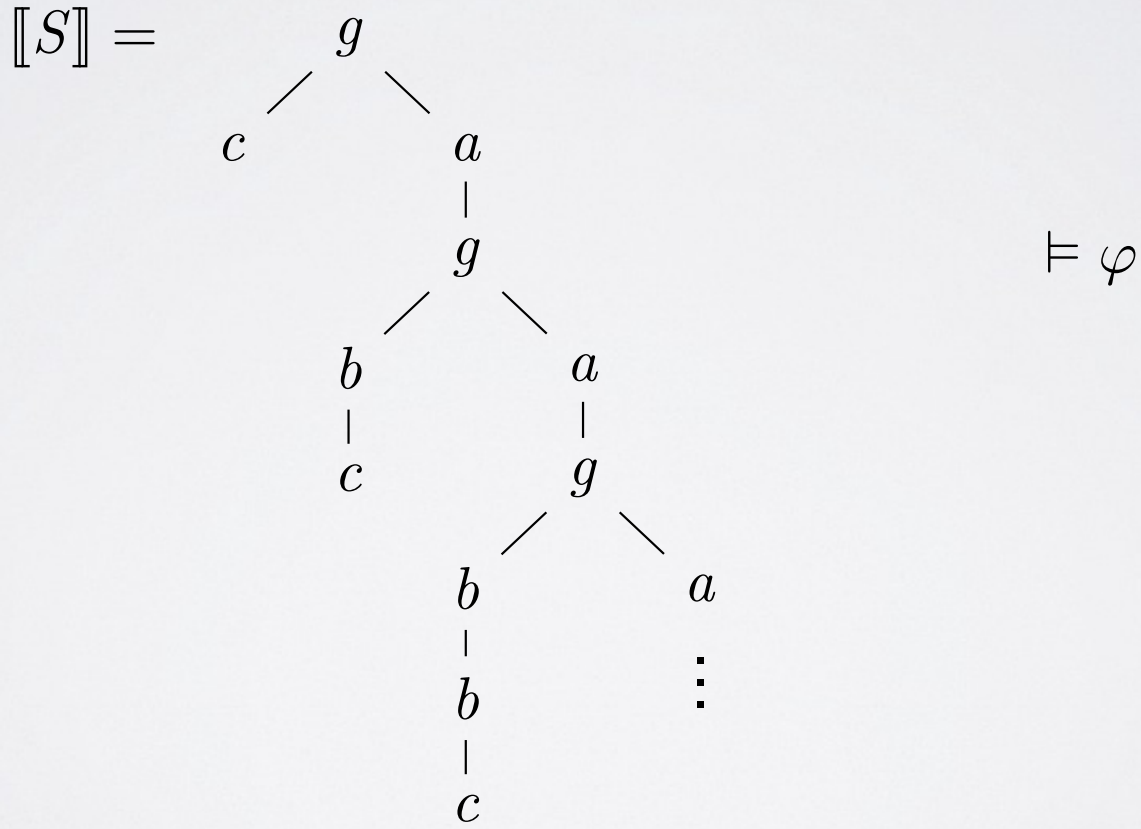
Does a given MSOL formula hold in a tree generated by a scheme?

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

Preparation

- λY-terms.
- Evaluation.
- Böhm trees.
- MSOL and automata.

$$M \xrightarrow{eval} BT(M) \stackrel{?}{\in} L(\mathcal{A})$$

Simply typed λ-calculus

Types:

- 0 is a type;
- $\alpha \to \beta$ is a type if α, β are types.

Eg. $(0 \to 0) \to 0$

Typed constants:

c^α for a type α.

Tree signature: All constants of types $0 \to \cdots \to 0 \to 0$.

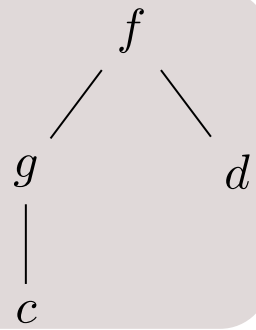
Typed terms:

c^α,
x^α,
$(M^{\alpha\to\beta} N^\alpha)^\beta$,
$(\lambda x^\alpha . M^\beta)^{\alpha\to\beta}$.

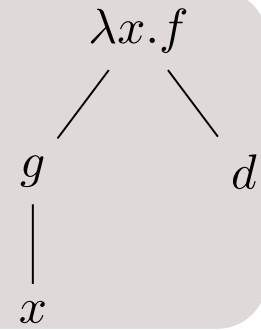
- Types: $0 \mid \alpha \to \beta$
- Constants: c^{α}
- Terms: $c^{\alpha}, \quad x^{\alpha}, \quad (M^{\alpha\to\beta} N^{\alpha})^{\beta}, \quad (\lambda x^{\alpha}.M^{\beta})^{\alpha\to\beta}.$

Example: $c, d : 0, \quad g : 0 \to 0, \quad f : 0 \to 0 \to 0$

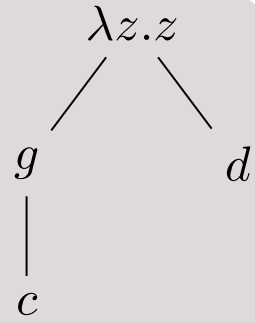
$f(gc)d : 0$



$\lambda x.f(gx)d : 0 \to 0$



$\lambda z.z(gc)d : (0 \to 0 \to 0) \to 0$



β-reduction: $(\lambda x.M)N =_\beta M[N/x]$

- $(\lambda x.f(gx)d)c \to_\beta f(gc)d$
- $(\lambda z.z(gc)d)(\lambda xy.y) \to_\beta (\lambda xy.y)(gc)d \to_\beta d$

Substitution is as in logic: one should avoid variable capture

$(\lambda h.\lambda x.g(hx))(fx) \to_\beta \lambda y.g(fxy)$

and not $\lambda x.g(fxx)$

$$f : 0 \to 0 \to 0, \quad g, h : 0 \to 0$$

Result of the computation $\equiv$ normal form

- $(\lambda x.f(gx)d)c \rightarrow_\beta f(gc)d$
- $(\lambda z.z(gc)d)(\lambda xy.y) \rightarrow_\beta (\lambda xy.y)(gc)d \rightarrow_\beta d$
- $(\lambda h.\lambda x.g(hx))(fx) \rightarrow_\beta \lambda y.g(fxy)$

Example (QBF)

- tt $= \lambda xy.\ x$, ff $= \lambda xy.\ y$, They are of type $0 \to 0 \to 0$.
- and $= \lambda b_1 b_2.\ \lambda xy.\ b_1 (b_2 xy) y$, or $= \lambda b_1 b_2.\ \lambda xy.\ b_1 x (b_2 xy)$,
- neg $= \lambda b.\ \lambda xy.\ byx$
- 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 formula α:

α is true iff M_α evaluates to tt.

Let us reduce: or (neg tt) tt

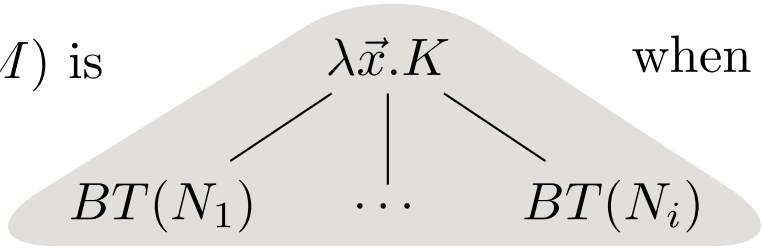
or (neg tt) tt		
$\lambda b_1 b_2.\ \lambda xy.\ b_1 x (b_2 xy)$		(neg tt), tt
$\lambda xy.$ (neg tt) x (tt x y)		

(neg tt) x (tt x y)		
$\lambda b.\lambda xy.\ byx$		tt, x, (tt x y)
tt (tt x y) x		
$\lambda xy.x$		(tt x y), x
(tt x y)		
$\lambda xy.x$		x, y
x		

We obtain: or (neg tt) tt $\rightarrow^*$ $\lambda xy.x \equiv$ tt

A **Böhm tree of a term** M**:**

$BT(M)$ is

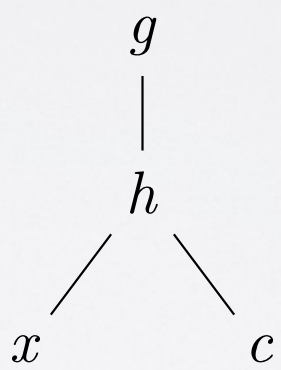


when $M \to_\beta^* \lambda\vec{x}.KN_1 \ldots N_i$

constant or variable

head normal form

Böhm tree of $(\lambda y.\ g\ (hxy))\ c$ is



Important: If $M : 0$ over tree signature then $BT(M)$ is a ranked tree, the only possible head normal form of M is $aN_1 \ldots N_k$.

λY-calculus

We add constants $Y^{(\alpha\to\alpha)\to\alpha}$ and Ω^α, for every type α.

New reduction rule $YM \to_\delta M(YM)$.

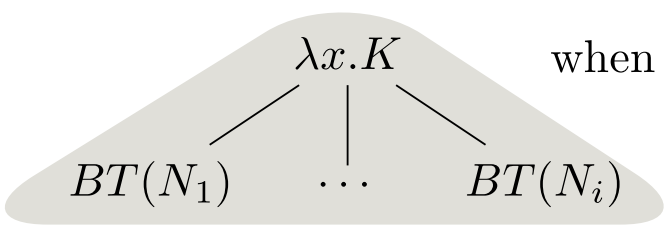
Example: YM with $M = (\lambda x.ax)$

$$\begin{aligned} YM &\to_\delta M(YM) \equiv (\lambda x.ax)(YM) \\ &\to_\beta a(YM) \\ &\to_\delta a(M(YM)) \\ &\to_\beta a(a(YM)) \to \dots \end{aligned}$$

What is the result of the computation? $BT(YM) = a^\omega$.

A **Böhm tree of a λY-term** M is:

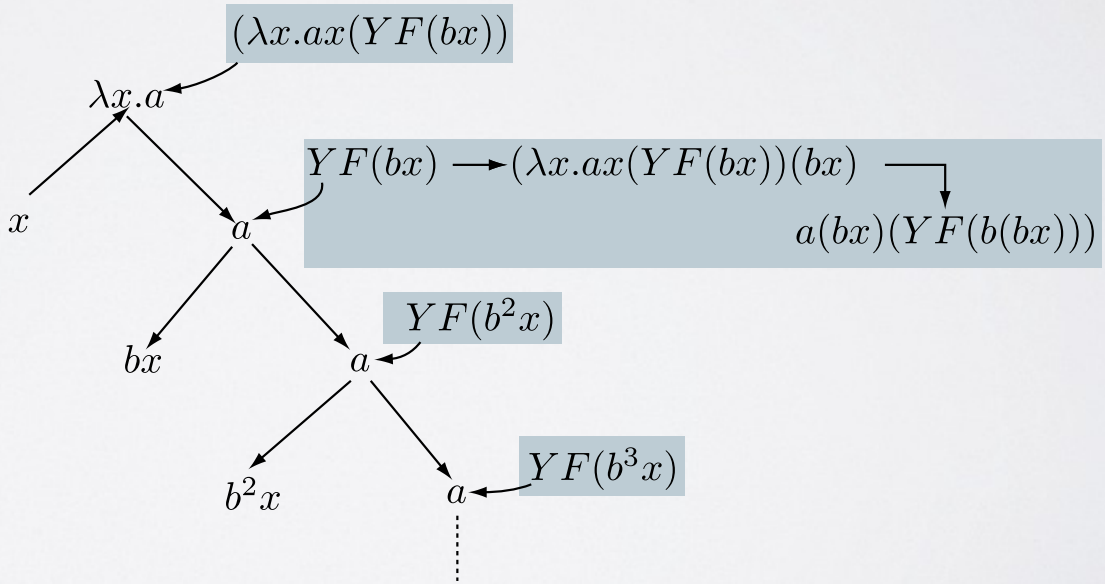
$BT(M)$ is



when $M \to_\beta^* \lambda\vec{x}.KN_1 \dots N_i$

Ω^α otherwise

$Y(\lambda F.\lambda x.ax(F(bx))) \ : \ 0 \to 0$



For closed terms of type 0 over tree signatures, Böhm tree is a tree.

Digression: Recursion schemes $\equiv \lambda Y$-calculus

$$\begin{aligned} F_1 &= \lambda \vec{x}.M_1 \\ &\vdots \\ F_n &= \lambda \vec{x}.M_n \end{aligned}$$

$$\begin{aligned} T_1 &= Y(\lambda F_1.M_1) \\ T_2 &= Y(\lambda F_2.M_2)[T_1/F_1]) \\ &\vdots \\ T_n &= Y(\lambda F_n.(\ldots((M_n[T_1/F_1])[T_2/F_2])\ldots)[T_{n-1}/F_{n-1}]) \end{aligned}$$

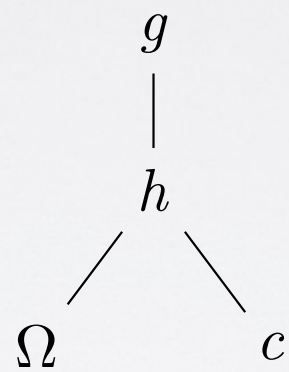
Fact

The tree generated from F_n is $BT(T_n)$.
There is also a translation from λY-terms to schemes.

Specifying properties of Böhm trees

Proviso: Σ has only constants of types 0 or $0 \to 0 \to 0$
(plus constants Ω^α, $Y^{(\alpha\to\alpha)\to\alpha}$).

Recall: For tree signature: if M is a closed term of type 0 then $BT(M)$ is a ranked tree.



Monadic second order logic:

$$\exists X.\, \forall y \in X.\, \exists z \in X.\, y < z \wedge a(z)$$

Tree automata:

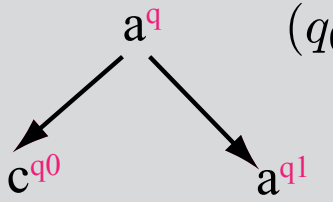
Proviso:

$\Sigma = \Sigma_0 \cup \Sigma_2$ with Σ_0 constants of type 0 and Σ_2 of type $0 \to 0 \to 0$.

Tree automaton:

$$\mathcal{A} = \langle Q, \Sigma \cup \{\Omega\}, q^0 \in Q, \delta_1 : Q \times \Sigma_0 \to \{false, true\}, \delta_2 : Q \times \Sigma_2 \to \mathcal{P}(Q^2) \rangle$$

Run of $\mathcal{A}$:



$(q_0, q_1) \in \delta(q, a)$

c^q	$\delta(q, c) = tt$
Ω^q	$q \in Q$

Trivial acceptance condition: every run is accepting.

Parity acceptance condition: max rank on every path is even.

First camp

- λ-terms $\xrightarrow{\beta-red}$ Böhm trees (normal form)
- λY-terms $\xrightarrow{\beta\delta-red}$ Böhm trees with Ω.
- Tree automata running on Böhm trees.

Models

Models

- The meaning of a term is its Böhm tree
- But we can also evaluate terms in models

$$\text{if } BT(M) = BT(N) \text{ then } [\![M]\!] = [\![N]\!]$$

- Types: $0 \mid \alpha \to \beta$
- Constants: c^α
- Terms: $c^\alpha, \quad x^\alpha, \quad (M^{\alpha\to\beta} N^\alpha)^\beta, \quad (\lambda x^\alpha . M^\beta)^{\alpha\to\beta}.$

MODEL: $\mathcal{D} = \langle \{D^\alpha\}_{\alpha\in\mathcal{T}}, \rho \rangle$

- D^0 is a complete lattice;
- $D^{\alpha\to\beta}$ monotone functions from D^α to D^β ordered coordinatewise;
- $\rho(\Omega^\alpha)$ is the greatest element of D^α;
- $\rho(\, Y^{(\alpha\to\alpha)\to\alpha})$ is a mapping assigning to a function $f \in D^{\alpha\to\alpha}$ its fixpoint.

- GFP model: when Y assigns greatest fixpoints.
- Finitary model: when every D^α is finite.

Interpretation of a term $M : \alpha$ in a model $\mathcal{D}$ is an element $[\![M]\!]_{\mathcal{D}} \in D^{\alpha}$.

- $[\![c]\!]^{v}_{\mathcal{D}} = \rho(c)$
- $[\![x^{\alpha}]\!]^{v}_{\mathcal{D}} = v(x^{\alpha})$
- $[\![MN]\!]^{v}_{\mathcal{D}} = [\![M]\!]^{v}_{\mathcal{D}}[\![N]\!]^{v}_{\mathcal{D}}$
- $[\![\lambda x^{\alpha}.M]\!]^{v}_{\mathcal{D}}$ is a function mapping an element $d \in D^{\alpha}$ to $[\![M]\!]^{v[d/x^{\alpha}]}_{\mathcal{D}}$. (this is a monotone function).

Fact:
For every model $\mathcal{D}$: if $M =_{\beta,\delta} N$ then $[\![M]\!]^{\mathcal{D}} = [\![N]\!]^{\mathcal{D}}$.

β-REDUCTION $(\lambda x.M)N \to_{\beta} M[N/x]$

δ-REDUCTION $Y(M) \to_{\delta} M(YM)$.

Example

Take $D^0 = \{0, 1\}$.

Then $D^{0\to 0\to 0}$ is $\{0, 1\} \to \{0, 1\} \to \{0, 1\}$.

$[\![\lambda xy.\ x]\!] = \pi_1 \in D^{0\to 0\to 0}$ is the projection on the first component.

$[\![\lambda xy.\ y]\!] = \pi_2 \in D^{0\to 0\to 0}$.

For every QBF sentence α: $[\![M_\alpha]\!] = \pi_1$ iff α is true.

Fact For every QBF formula α:

$$\alpha \text{ is true} \quad \text{iff} \quad M_\alpha \text{ reduces to } \lambda xy.\ x$$

Digression

Thm [Statman's Weak Completeness Theorem '82]:
For every λ-term M there is a finitary model $\mathcal{D}_M$ such that for every λ-term K:

$$[\![M]\!]^{\mathcal{D}_M} = [\![K]\!]^{\mathcal{D}_N} \quad \text{iff} \quad M =_\beta K \ .$$

Thm [Loader's λ-definability theorem '96]:
For every nontrivial finitary model $\mathcal{D}$. It is not decidable if a given element d of the model is a denotation of a term.

Interpretation of a term $M : \alpha$ in a model $\mathcal{D}$ is an element $[\![M]\!]_{\mathcal{D}} \in D^{\alpha}$.

- $[\![c]\!]^{\upsilon}_{\mathcal{D}} = \rho(c)$
- $[\![x^{\alpha}]\!]^{\upsilon}_{\mathcal{D}} = \upsilon(x^{\alpha})$
- $[\![MN]\!]^{\upsilon}_{\mathcal{D}} = [\![M]\!]^{\upsilon}_{\mathcal{D}}[\![N]\!]^{\upsilon}_{\mathcal{D}}$
- $[\![\lambda x^{\alpha}.M]\!]^{\upsilon}_{\mathcal{D}}$ is a function mapping an element $d \in D^{\alpha}$ to $[\![M]\!]^{\upsilon[d/x^{\alpha}]}_{\mathcal{D}}$.

Fact:
For every model $\mathcal{D}$: if $M =_{\beta,\delta} N$ then $[\![M]\!]^{\mathcal{D}} = [\![N]\!]^{\mathcal{D}}$.

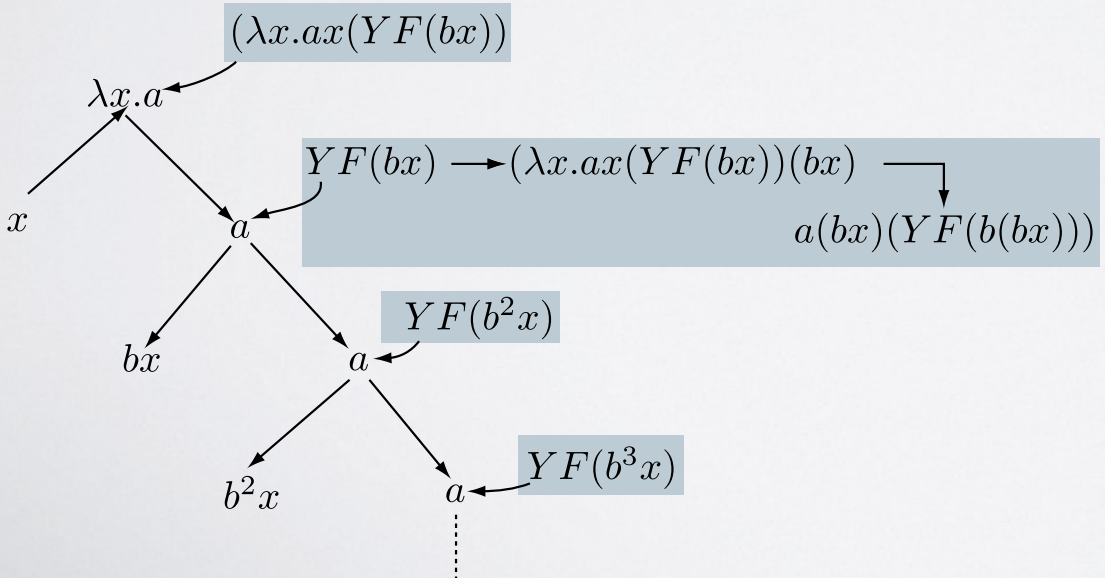
Theorem [Barendregt]: For every finitary GFP-model $\mathcal{D}$:
if $BT(M) \equiv BT(N)$ then $[\![M]\!]^{\mathcal{D}} = [\![N]\!]^{\mathcal{D}}$.

Approximate Böhm tree

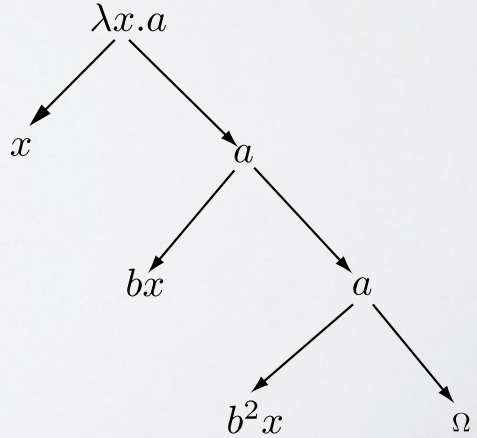
$ABT(M)$ is defined by

$$ABT(M) \text{ is } \begin{cases} \lambda x.K \;(BT(N_1)\ \cdots\ BT(N_i)) & \text{when } M \equiv \lambda\vec{x}.KN_1 \ldots N_i \\ \Omega^\alpha & \text{otherwise} \end{cases}$$

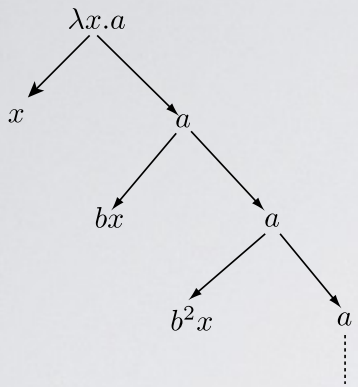
$Y(\lambda F.\lambda x.ax(F(bx)))\ :\ 0 \to 0$

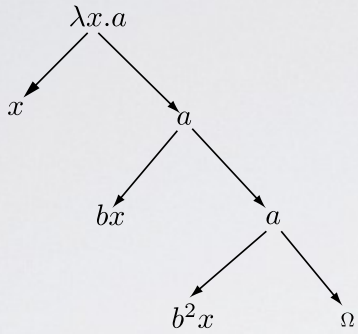


$\lambda x.a\ x\ (a\ (bx)\ (YF\ (b^2x)))$



Meanings of Böhm trees





Lemma

$BT(M) = \bigsqcup\{ABT(N) : N =_{\beta,\delta} M\}$;
here we are taking syntactic limit over trees.

Semantics

$$[\![BT(M)]\!]^{\mathcal{D}} = \bigwedge\{[\![ABT(N)]\!]^{\mathcal{D}} : N =_{\beta,\delta} M\}$$

Theorem [?]

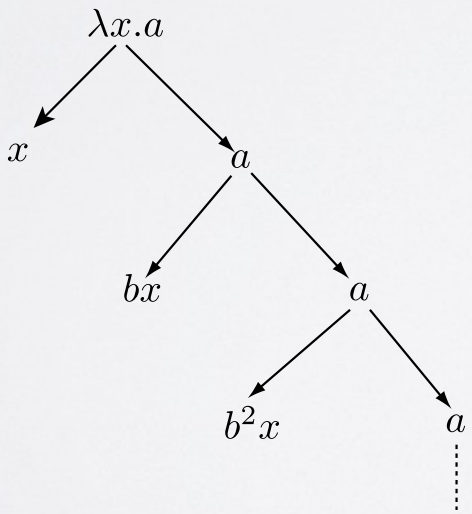
If $\mathcal{D}$ is a finitary GFP model then: $[\![M]\!]^{\mathcal{D}} = [\![BT(M)]\!]^{\mathcal{D}}$.

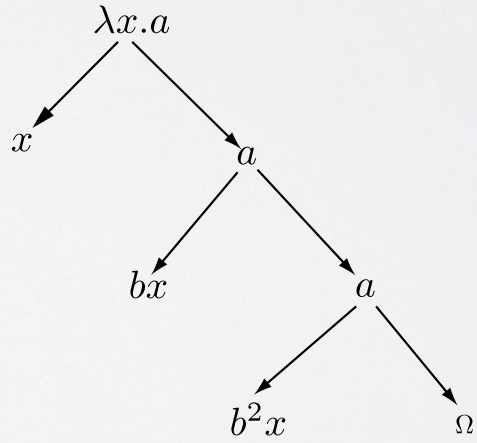
Theorem

If $\mathcal{D}$ is a finitary GFP model then: $[\![M]\!]^{\mathcal{D}} = [\![BT(M)]\!]^{\mathcal{D}}$.

Proof $[\![BT(M)]\!] \geq [\![M]\!]$**:**

- $[\![BT(M)]\!]^{\mathcal{D}} = \bigwedge\{[\![ABT(N)]\!]^{\mathcal{D}} : N =_{\beta,\delta} M\}$.
- $[\![ABT(N)]\!]^{\mathcal{D}} \geq [\![N]\!]^{\mathcal{D}} = [\![M]\!]^{\mathcal{D}}$.





Theorem

If $\mathcal{D}$ is a finitary GFP model then: $[\![M]\!]^{\mathcal{D}} = [\![BT(M)]\!]^{\mathcal{D}}$.

Proof $[\![M]\!] \geq [\![BT(M)]\!]$**:**

- Let $N : \alpha \to \alpha$ without Y:
  Define *iterate*$^i(N)$ to be $N(\ldots(N\Omega^\alpha)\ldots)$.
- Define *iterate*$^i(M)$ as the result of repeatedly replacing all YN by *iterate*$^i(N)$.

Obs: If $\mathcal{D}$ is a finitary GFP model then there is i such that $[\![M]\!]^{\mathcal{D}} = [\![\textit{iterate}^i(M)]\!]^{\mathcal{D}}$.

$$[\![M]\!] = [\![\textit{iterate}^i(M)]\!] = [\![BT(\textit{iterate}^i(M))]\!] \geq [\![BT(M)]\!]$$

Almost there

We have

1. Models $\mathcal{D} = (\{D^\alpha\}_{\alpha \in \mathcal{T}}, \rho)$ interpreting fixpoint operators.
2. Models are capable of talking about Böhm trees:

$$[\![M]\!]^{\mathcal{D}} = [\![BT(M)]\!]^{\mathcal{D}}$$

We want

- A model $\mathcal{D}_{\mathcal{A}}$ such that $[\![M]\!]^{\mathcal{D}_{\mathcal{A}}}$ tells us if $BT(M)$ is accepted by $\mathcal{A}$.

$$\mathcal{A} = \langle Q, \Sigma \cup \{\Omega\}, q^0 \in Q,$$
$$\delta_1 : Q \times \Sigma_0 \to \{false, true\}, \delta_2 : Q \times \Sigma_2 \to \mathcal{P}(Q^2)\rangle$$

TAC (trivial acceptance condition) : all runs are accepting.

Model $\mathcal{D}_{\mathcal{A}}$:

- $D^0 = \mathcal{P}(Q)$.
- If $c : 0$ then $[\![c]\!] = \{q : \delta_1(q, c) = true\}$. $([\![\Omega]\!] = Q)$
- If $a : 0^2 \to 0$ then $[\![a]\!]$ is a function that for $(S_0, S_1) \in \mathcal{P}(Q)^2$ returns

$$\{q : \delta_2(q, a) \in S_0 \times S_1\}$$

THEOREM

For every closed term M of type 0:

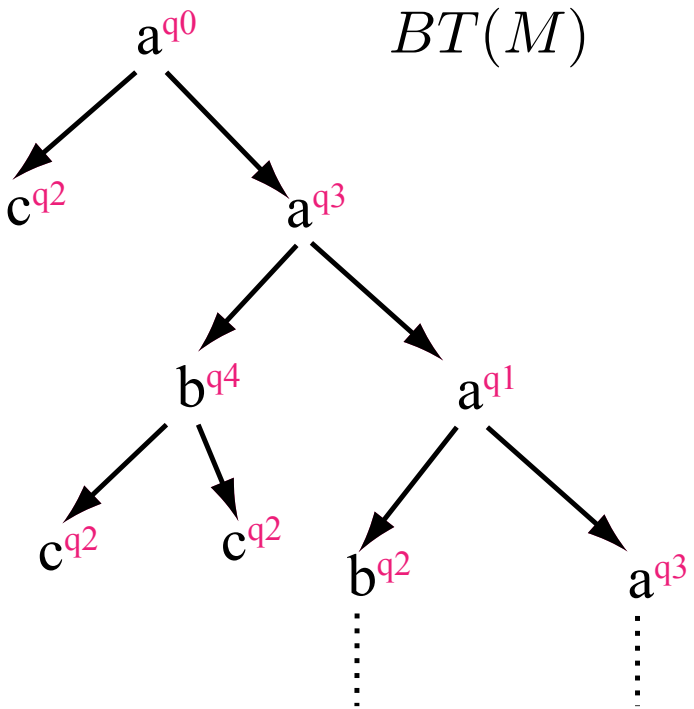
$$BT(M) \in L(\mathcal{A}) \quad \text{iff} \quad q_0 \in [\![M]\!]^{D_{\mathcal{A}}}$$

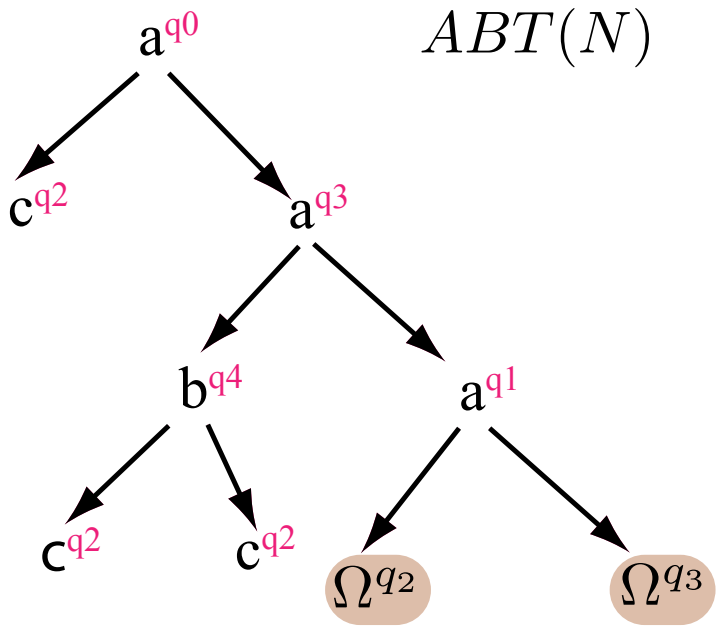
If $BT(M) \in L(\mathcal{A})$ then $q_0 \in [\![M]\!]^{\mathcal{D}_{\mathcal{A}}}$

Take a run of $\mathcal{A}$ on $BT(M)$ and show that $q^0 \in [\![BT(M)]\!]^{\mathcal{D}_{\mathcal{A}}} = [\![M]\!]^{\mathcal{D}_{\mathcal{A}}}$.

Recall that $[\![BT(M)]\!]^{\mathcal{D}} = \bigwedge\{[\![ABT(N)]\!]^{\mathcal{D}} : N =_{\beta,\delta} M\}$

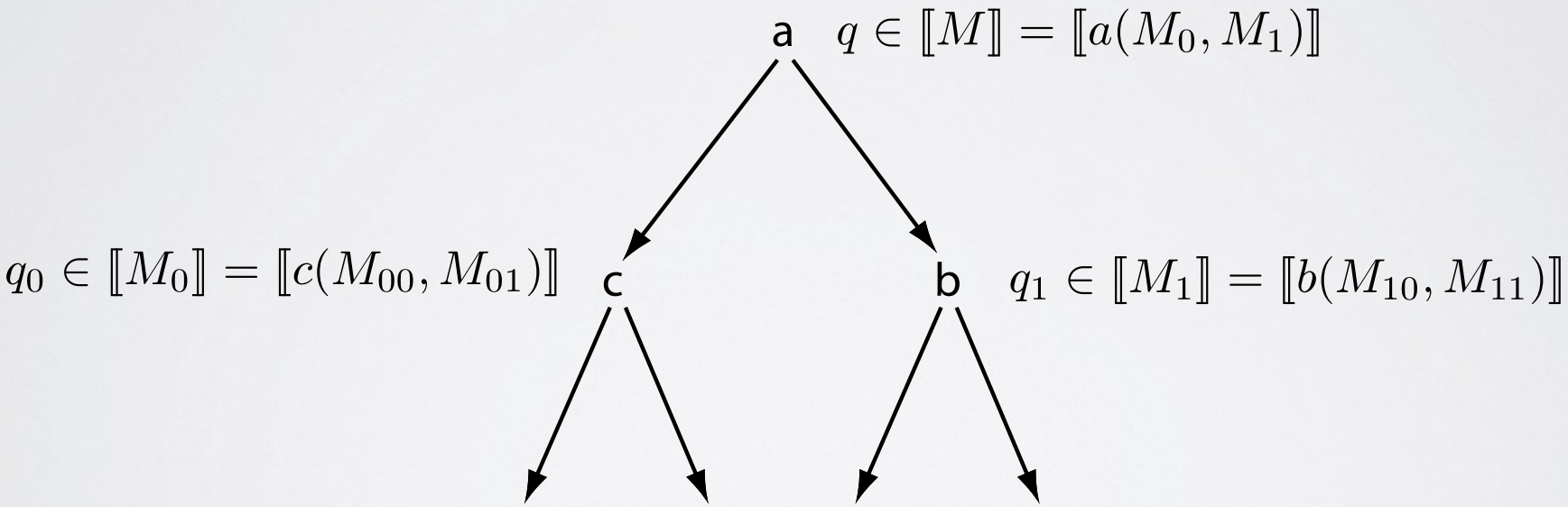
We show: $q^0 \in [\![ABT(N)]\!]$ for $N =_{\beta,\delta} M$.





If $q_0 \in [\![M]\!]$ then $BT(M) \in L(\mathcal{A})$

Property of the interpretation:
If $q \in [\![a(M_0, M_1)]\!]$ then there is $(q_0, q_1) \in \delta(q, a)$ such that: $q_0 \in [\![M_0]\!]$, and $q_1 \in [\![M_1]\!]$.



$\mathcal{A} = \langle Q, \Sigma \cup \{\Omega\}, q^0 \in Q, \delta_1 : Q \times \Sigma_0 \to \{false, true\}, \delta_2 : Q \times \Sigma_2 \to \mathcal{P}(Q^2)\rangle$

Model $\mathcal{D}_\mathcal{A}$:

- $D^0 = \mathcal{P}(Q)$.
- If $c : 0$ then $[\![c]\!] = \{q : \delta_1(q, c) = true\}$. $([\![\Omega]\!] = Q)$
- If $a : 0^2 \to 0$ then $[\![a]\!]$ is a function that for $(S_0, S_1) \in \mathcal{P}(Q)^2$ returns

$$\{q : \delta_2(q, a) \in S_0 \times S_1\}$$

THEOREM

For every closed term M of type 0:

$$BT(M) \in L(\mathcal{A}) \quad \text{iff} \quad q_0 \in [\![M]\!]^{D_\mathcal{A}}$$

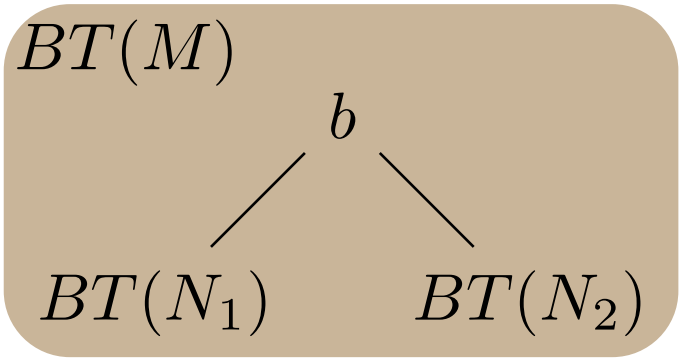
To decide $BT(M) \stackrel{?}{\in} L(\mathcal{A})$ it is enough to:

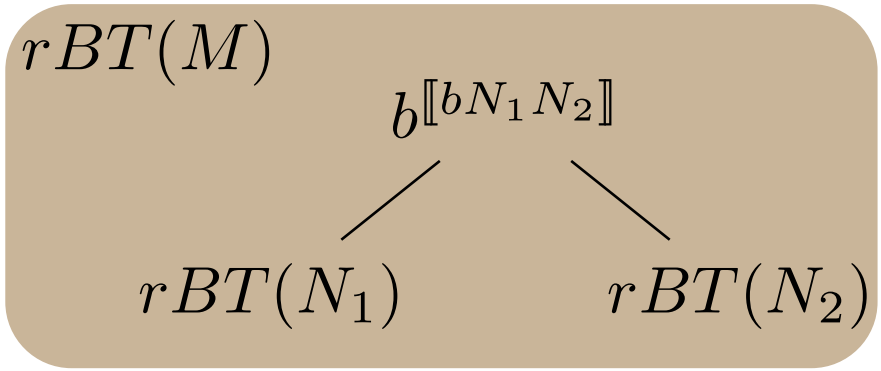
- Construct $\mathcal{D}_{\mathcal{A}}$,
- Calculate $[\![M]\!]^{\mathcal{D}_{\mathcal{A}}}$.

This works only for TAC conditions. (Simple models $\equiv$ TAC conditions)

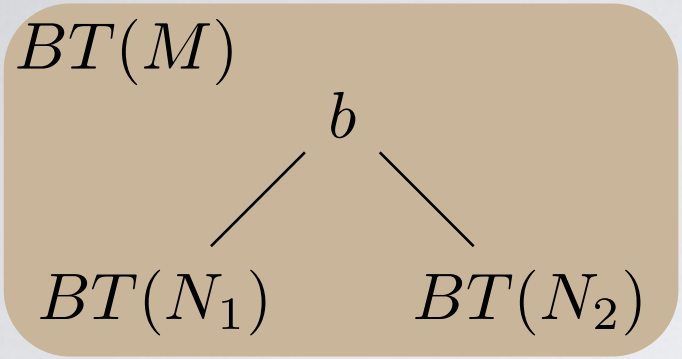
We can do Ω-aware TAC, but the climb is rather steep.

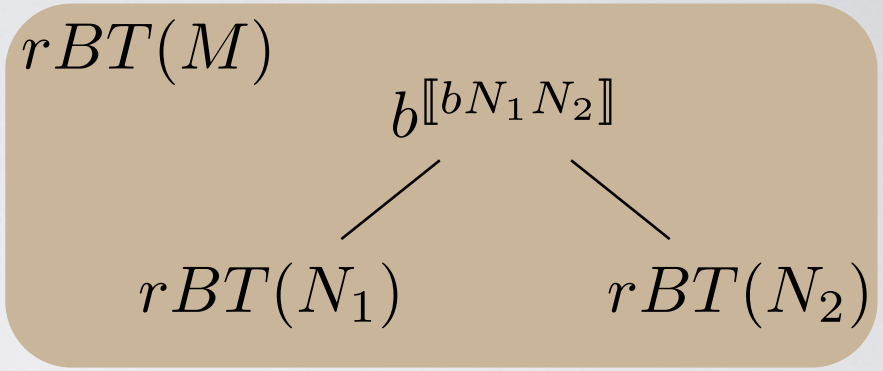
Reflective Böhm tree wrt. a model $\mathcal{D}$:





Thm [Broadbent, Carayol, Ong, Serre]: For every finitary model $\mathcal{D}$ and λY-term M there is a λY-term N such that $BT(N) = rBT_{\mathcal{D}}(M)$.





$(\alpha \to \beta)^\bullet = \alpha^\bullet \to [\alpha] \to \beta^\bullet$ and $\alpha^\bullet = \alpha$ when α is atomic.

$$
\begin{aligned}
[MN, \upsilon] &= [M, \upsilon]\,[N, \upsilon]\,[\![N]\!]^\upsilon \\
[x^\alpha, \upsilon] &= x^{\alpha^\bullet} \\
\left[\, Y^{(\alpha\to\alpha)\to\alpha} M, \upsilon \right] &= Y^{(\alpha^\bullet\to\alpha^\bullet)\to\alpha^\bullet}(\lambda x^{\alpha^\bullet}.\,[M, \upsilon]\,x^{\alpha^\bullet}\,[\![\,YM]\!]^\upsilon) \\
[\lambda x^\alpha.M, \upsilon] &= \lambda x^{\alpha^\bullet} \lambda y^{[\alpha]}.\ \mathsf{case}\ y^{[\alpha]}\{d \to [M, \upsilon[d/x^\alpha]]\}_{d\in\mathcal{S}_\alpha} \\
[a, \upsilon] &= \lambda x_1^0 \lambda y_1^{[0]} \lambda x_2^0 \lambda y_2^{[0]}.\mathsf{case}\ y_1^{[0]}\{d_1 \to \\
&\qquad \mathsf{case}\ y_2^{[0]}\{d_2 \to a^{\rho(a)\,d_1\,d_2} x_1\,x_2\}_{d_2\in\mathcal{S}_0}\}_{d_1\in\mathcal{S}_0}
\end{aligned}
$$

Krivine machines

Krivine machines

- The meaning of a term is its Böhm tree.
- It can be computed with a Krivine machine.
- So now instead of using semantics we use syntax.

Our objective is to decide, for a fixed $\mathcal{A}$,

$$\text{if for a given } M\text{:} \quad BT(M) \in L(\mathcal{A}).$$

We will:

1. use Krivine machine to compute $BT(M)$,
2. construct a game $\mathcal{K}(\mathcal{A}, M)$ on this computation,
3. reduce it to $G(\mathcal{A}, M)$ that will be a finite game.

1. Krivine machine calculating $BT(M)$

2. Acceptance in terms of a game $\mathcal{K}(\mathcal{A}, M)$

3. Reduction of $\mathcal{K}(\mathcal{A}, M)$ to $G(\mathcal{A}, M)$

Krivine machine

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

Term	Environment	Stack
$(\lambda z.\ z(gc)d)(\lambda xy.\ y)$	$\emptyset$	$\bot$

Krivine machine

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

Term	Environment	Stack
$(\lambda z.\ z(gc)d)(\lambda xy.\ y)$	$\emptyset$	$\bot$
$\lambda z.\ z(gc)d$	$\emptyset$	$(\lambda xy.\ y, \emptyset)$

Krivine machine

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

Term	Environment	Stack
$(\lambda z.\ z(gc)d)(\lambda xy.\ y)$	$\emptyset$	$\bot$
$\lambda z.\ z(gc)d$	$\emptyset$	$(\lambda xy.\ y, \emptyset)$
$z(gc)d$	$[z \mapsto (\lambda xy.\ y, \emptyset)]$	$\bot$

Krivine machine

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

Term	Environment	Stack
$(\lambda z.\ z(gc)d)(\lambda xy.\ y)$	$\emptyset$	$\bot$
$\lambda z.\ z(gc)d$	$\emptyset$	$(\lambda xy.\ y, \emptyset)$
$z(gc)d$	$[z \mapsto (\lambda xy.\ y, \emptyset)]$ Let $\rho \equiv [z \mapsto (\lambda xy.\ y, \emptyset)]$	$\bot$

Krivine machine

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

Term	Environment	Stack
$(\lambda z.\ z(gc)d)(\lambda xy.\ y)$	$\emptyset$	$\bot$
$\lambda z.\ z(gc)d$	$\emptyset$	$(\lambda xy.\ y, \emptyset)$
$z(gc)d$	$[z \mapsto (\lambda xy.\ y, \emptyset)]$ Let $\rho \equiv [z \mapsto (\lambda xy.\ y, \emptyset)]$	$\bot$
z	ρ	$(gc, \rho)\ (d, \rho)$

Krivine machine

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

Term	Environment	Stack
$(\lambda z.\ z(gc)d)(\lambda xy.\ y)$	$\emptyset$	$\bot$
$\lambda z.\ z(gc)d$	$\emptyset$	$(\lambda xy.\ y, \emptyset)$
$z(gc)d$	$[z \mapsto (\lambda xy.\ y, \emptyset)]$ Let $\rho \equiv [z \mapsto (\lambda xy.\ y, \emptyset)]$	$\bot$
z	ρ	$(gc, \rho)\ (d, \rho)$
$\lambda xy.\ y$	$\emptyset$	$(gc, \rho)\ (d, \rho)$

Krivine machine

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

Term	Environment	Stack
$(\lambda z.\ z(gc)d)(\lambda xy.\ y)$	$\emptyset$	$\bot$
$\lambda z.\ z(gc)d$	$\emptyset$	$(\lambda xy.\ y, \emptyset)$
$z(gc)d$	$[z \mapsto (\lambda xy.\ y, \emptyset)]$ Let $\rho \equiv [z \mapsto (\lambda xy.\ y, \emptyset)]$	$\bot$
z	ρ	$(gc, \rho)\ (d, \rho)$
$\lambda xy.\ y$	$\emptyset$	$(gc, \rho)\ (d, \rho)$
y	$[x \mapsto (gc, \rho)][y \mapsto (d, \rho)]$	$\bot$

Krivine machine

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

Term	Environment	Stack
$(\lambda z.\ z(gc)d)(\lambda xy.\ y)$	$\emptyset$	$\bot$
$\lambda z.\ z(gc)d$	$\emptyset$	$(\lambda xy.\ y, \emptyset)$
$z(gc)d$	$[z \mapsto (\lambda xy.\ y, \emptyset)]$ Let $\rho \equiv [z \mapsto (\lambda xy.\ y, \emptyset)]$	$\bot$
z	ρ	$(gc, \rho)\ (d, \rho)$
$\lambda xy.\ y$	$\emptyset$	$(gc, \rho)\ (d, \rho)$
y	$[x \mapsto (gc, \rho)][y \mapsto (d, \rho)]$	$\bot$
d	ρ	$\bot$

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 a term:

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

Example:

- $(z^{0\to 0\to 0},\ \rho,\ (gc, \rho)(d, \rho))$ with $\rho \equiv [z \mapsto (\lambda xy.\ y, \emptyset)]$ gives

$$(\lambda xy.\ y)\ (gc)\ d$$

Krivine machine

$$(\lambda x.N, \rho, (K, \rho')S) \to (N, \rho[x \mapsto (K, \rho')], S)$$

$$(\,YN, \rho, S) \to (N(\,YN), \rho, S)$$

$$(NK, \rho, S) \to (N, \rho, (K, \rho)S)$$

$$(x, \rho, S) \to (N, \rho', S) \quad \text{where } (N, \rho') = \rho(x)$$

Lemma: Term $E(N, \rho, \bot)$ has a head normal form iff Krivine machine reduces $(N, \rho, \bot)$ to a $(b(N_1, N_2), \rho', \bot)$ for some constant $b \neq \Omega$.

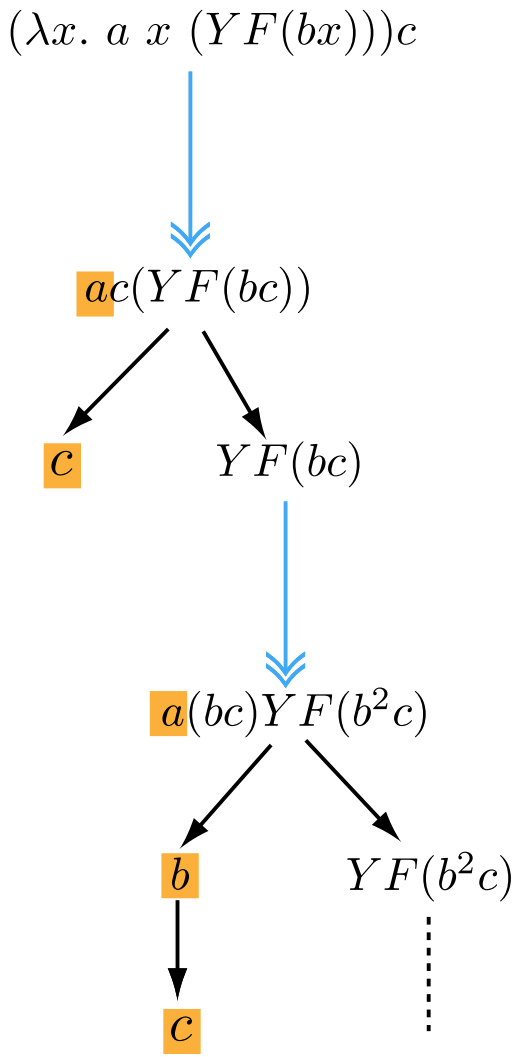
Lemma: All the terms appearing in configurations of the Krivine machine during the computation from $(M, \emptyset, \bot)$ are subterms of M.

$BT(M)$ with Krivine machines

The Böhm tree of

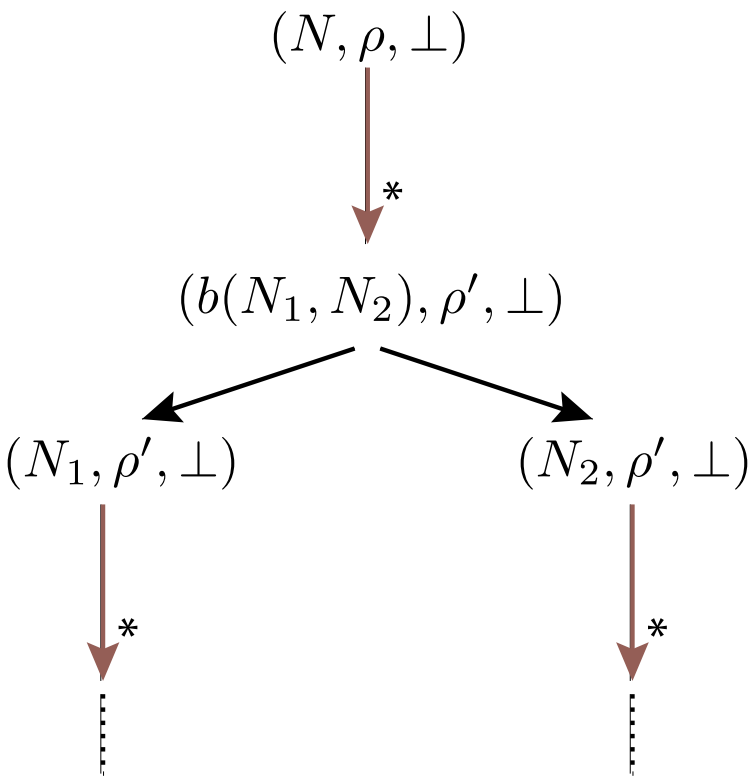
$$Y\left(\lambda F.\ \lambda x.\ a\ x\ \left(F(bx)\right)\right)\ c\ :0$$

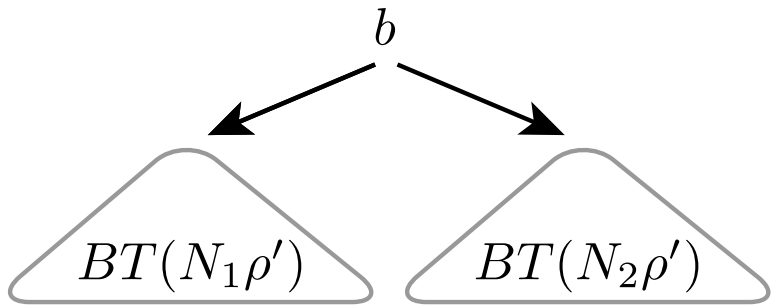
is



Computing Böhm tree

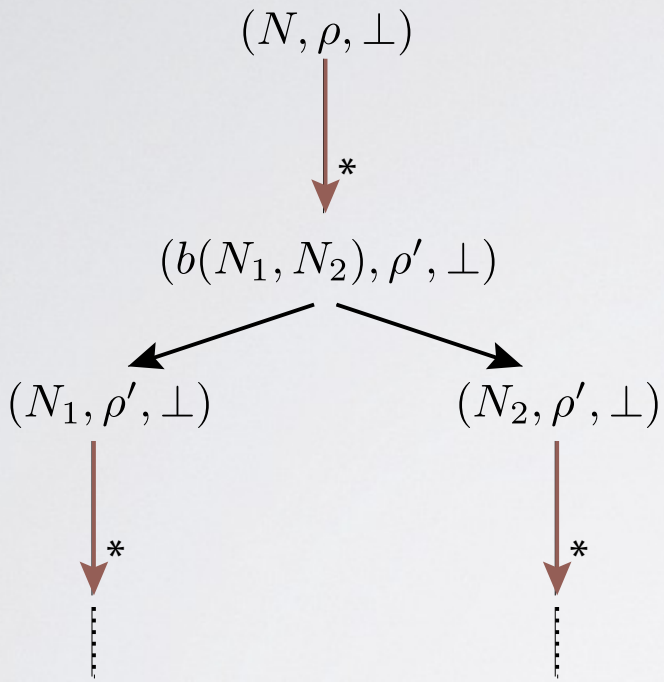
Lemma: Term $E(N, \rho, \bot)$ has a head normal form iff Krivine machine reduces $(N, \rho, \bot)$ to a $(b(N_1, N_2), \rho', \bot)$ for some constant $b \neq \Omega$.

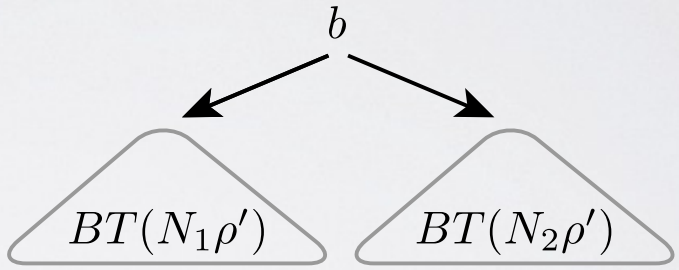




$Ktree(N, \rho, \bot) = BT(N\rho)$

Computing Böhm tree



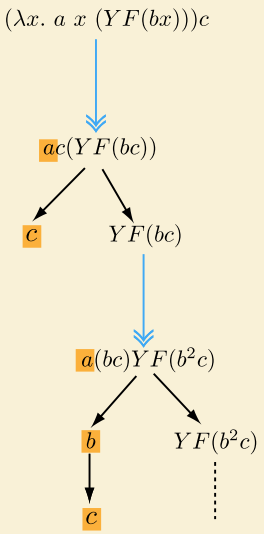


$Ktree(N, \rho, \perp) = BT(N\rho)$

Proposition: For every closed λY-term M of type 0:

$$BT(M) = Ktree(M, \emptyset, \perp).$$

1. Krivine machine calculating $BT(M)$

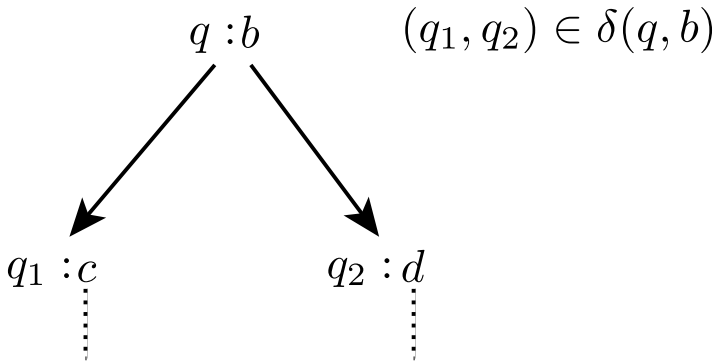


2. Acceptance in terms of a game $\mathcal{K}(\mathcal{A}, M)$

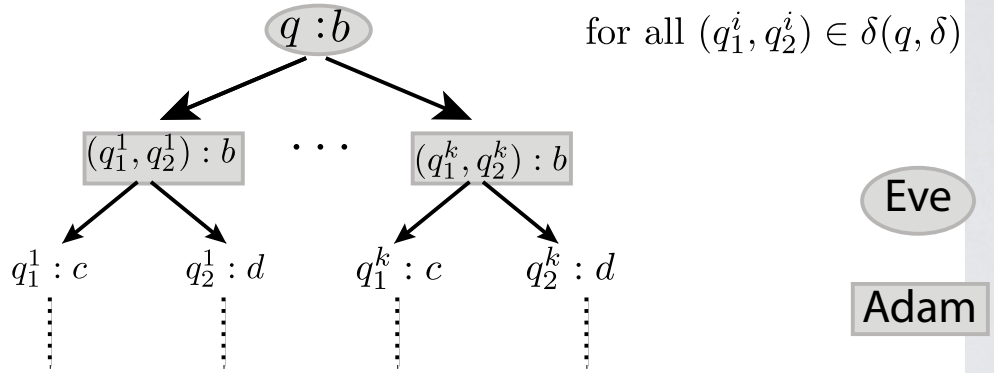
3. Reduction of $\mathcal{K}(\mathcal{A}, M)$ to $G(\mathcal{A}, M)$

Game for automaton acceptance

Run of $\mathcal{A}$ on t

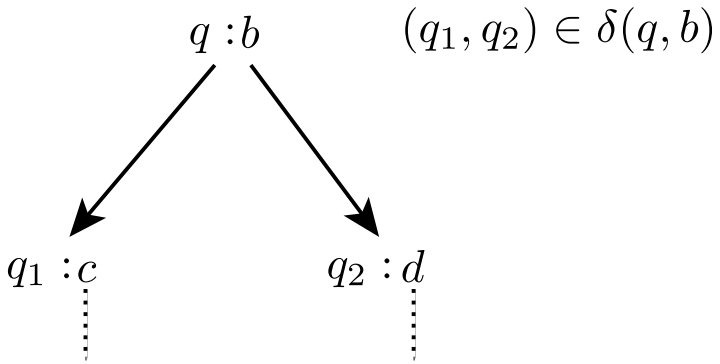


Acceptance game $G(\mathcal{A}, t)$

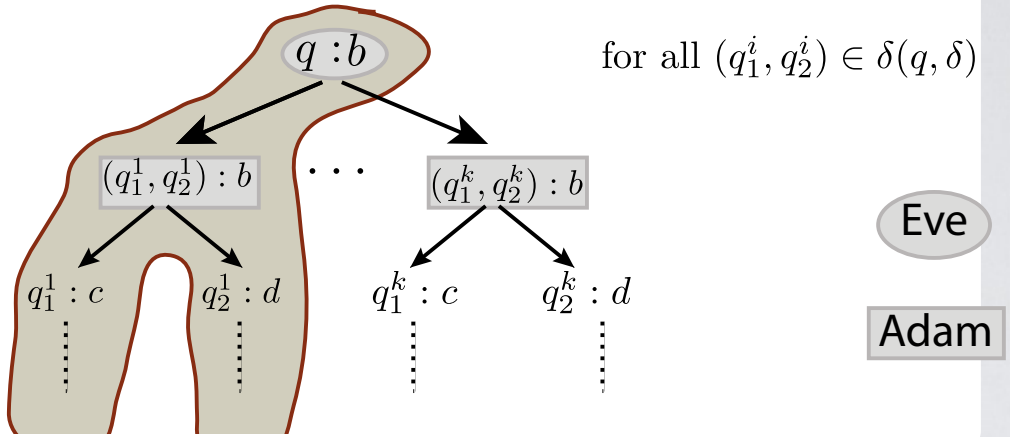


Game for automaton acceptance

Run of $\mathcal{A}$ on t



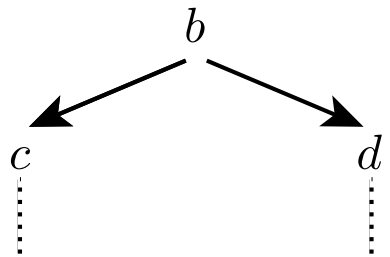
Acceptance game $G(\mathcal{A}, t)$



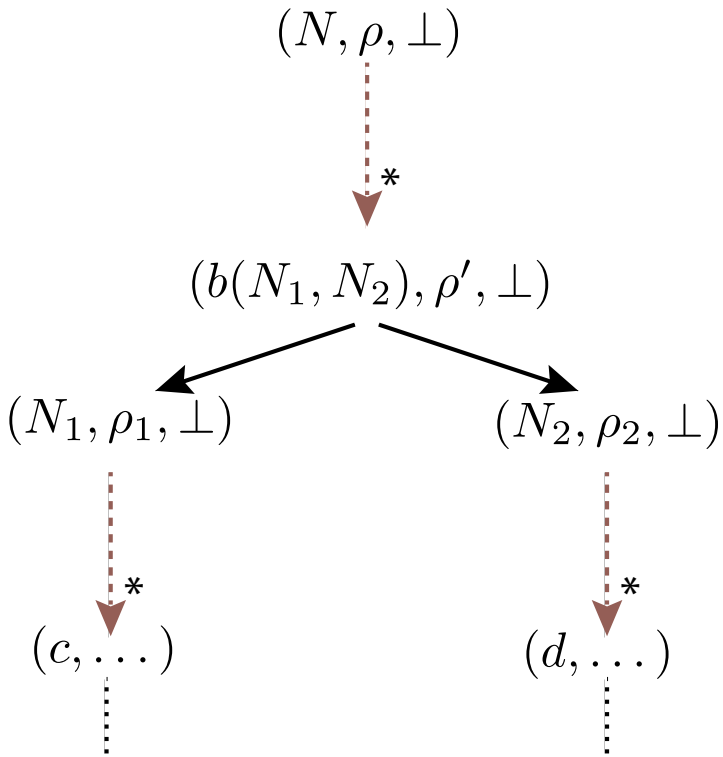
Eve has a strategy in $G(\mathcal{A}, t)$ iff t is accepted by $\mathcal{A}$.

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

Bohm tree

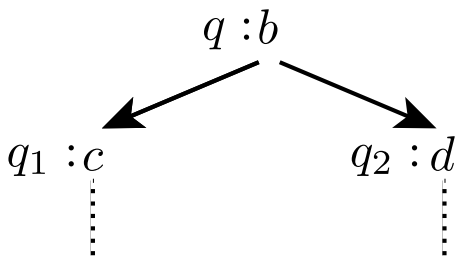


Krivine machine computing BT



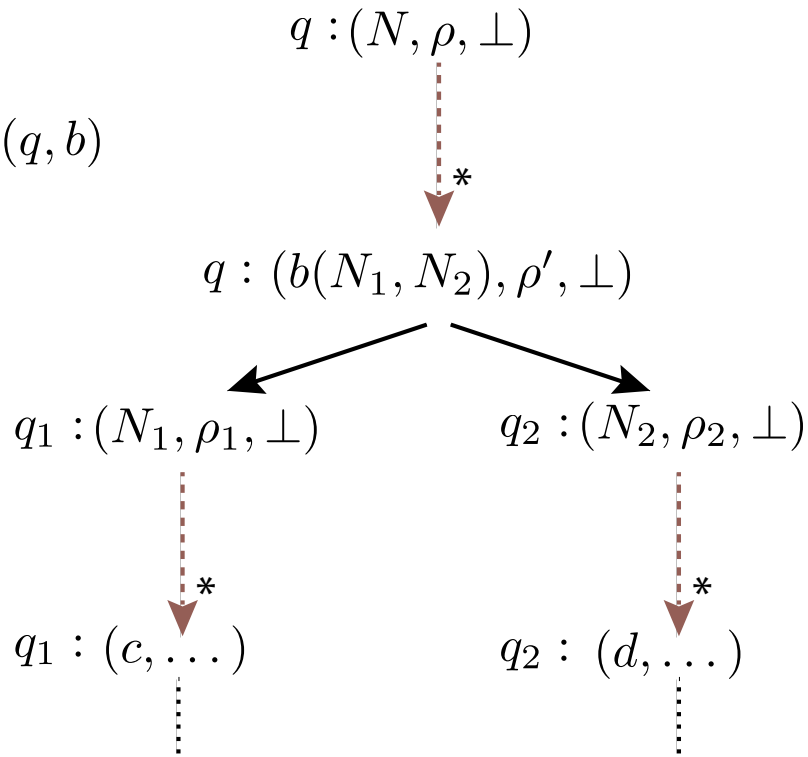
Defining $\mathcal{K}(\mathcal{A}, M)$

Run of the automaton on the Bohm tree



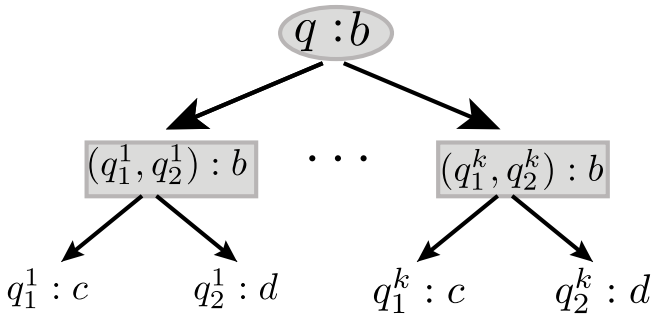
$(q_1, q_2) \in \delta(q, b)$

Run of the automaton on Krivine machine comptuation



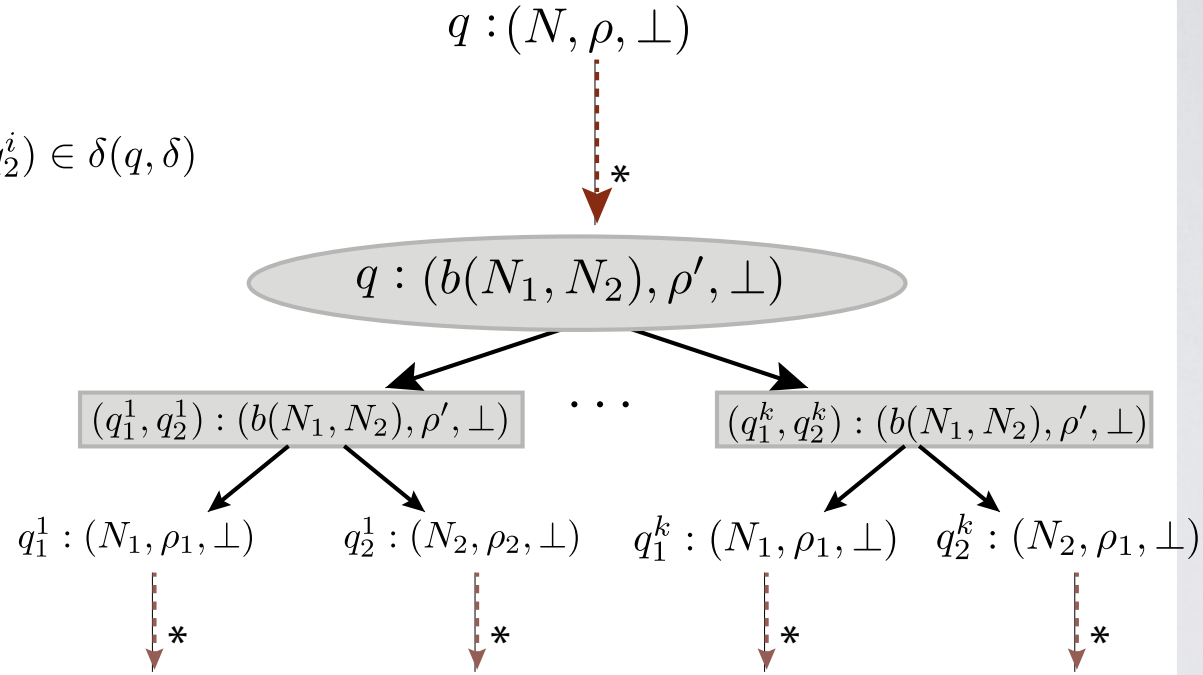
Defining $\mathcal{K}(\mathcal{A}, M)$

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

$q^0 : (M, \emptyset, \bot)$

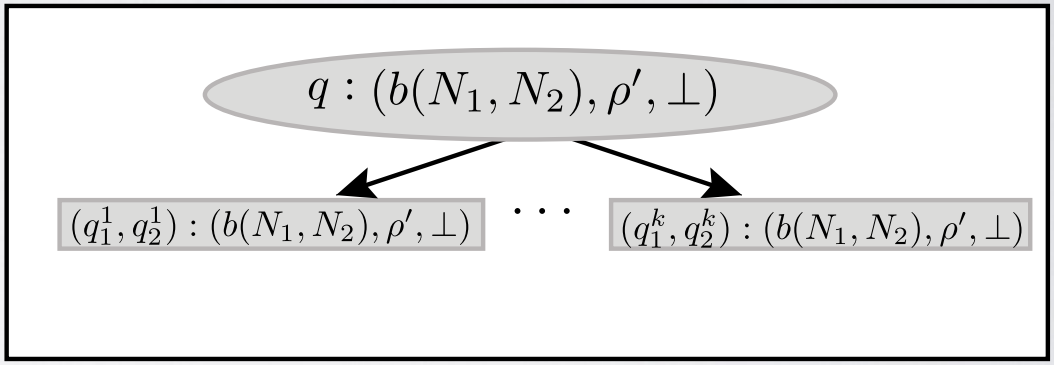
$q : (aN_0N_1, \rho, \bot)$

for $(q_1, q_2) \in \delta(q, b)$

$(q_0, q_1) : (aN_0N_1, \rho, \bot)$

$(q_0, q_1) : (aN_0N_1, \rho, \bot)$

$q_0 : (N_0, \rho, \bot)$ $\quad$ $q_1 : (N_1, \rho, \bot)$



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

$q^0 : (M, \emptyset, \bot)$

$q : (aN_0N_1, \rho, \bot)$

for $(q_1, q_2) \in \delta(q, b)$

$(q_0, q_1) : (aN_0N_1, \rho, \bot)$

$q : (\lambda x.N, \rho, CS)$

$q : (N, \rho[x \mapsto C], S)$

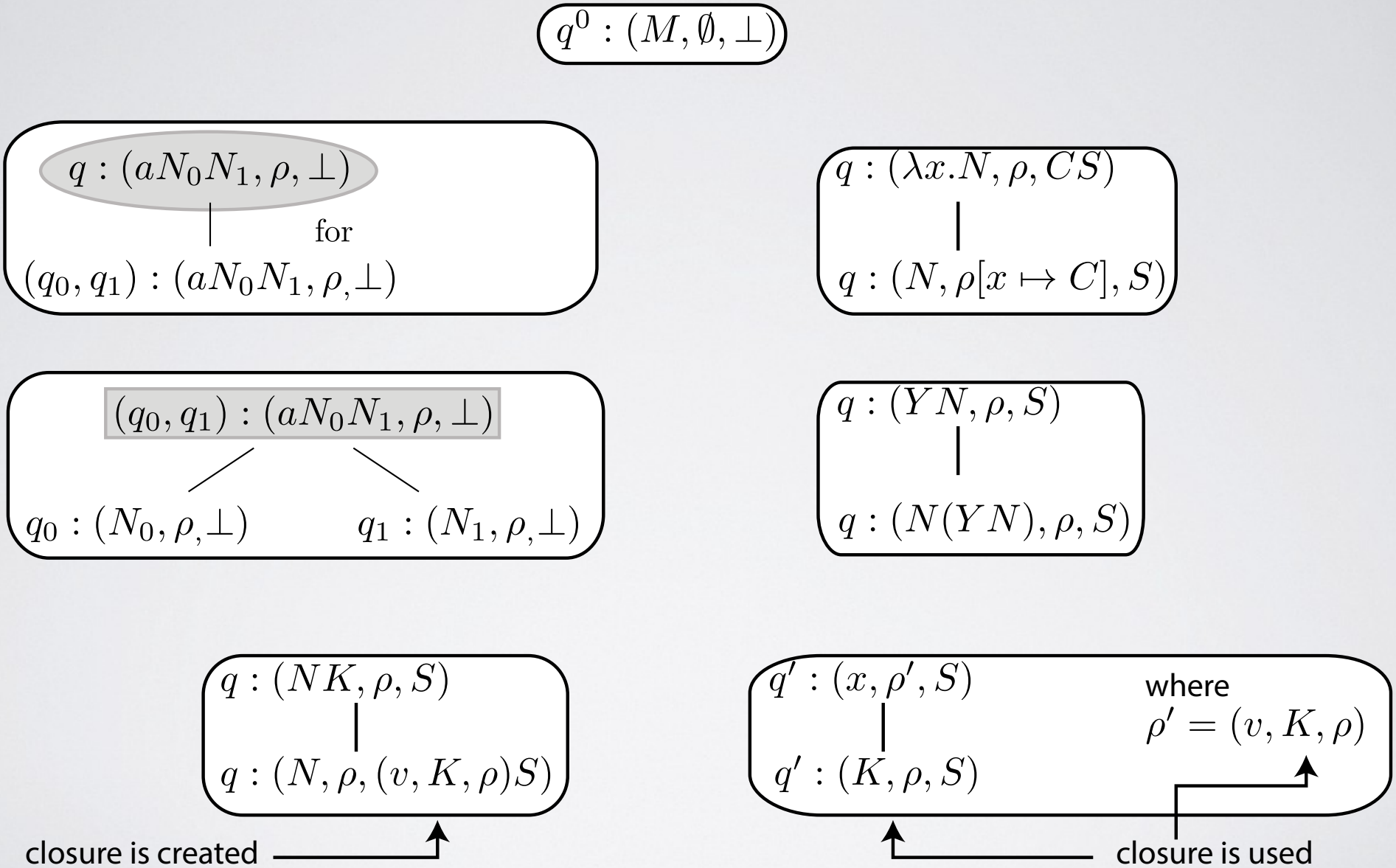
$(q_0, q_1) : (aN_0N_1, \rho, \bot)$

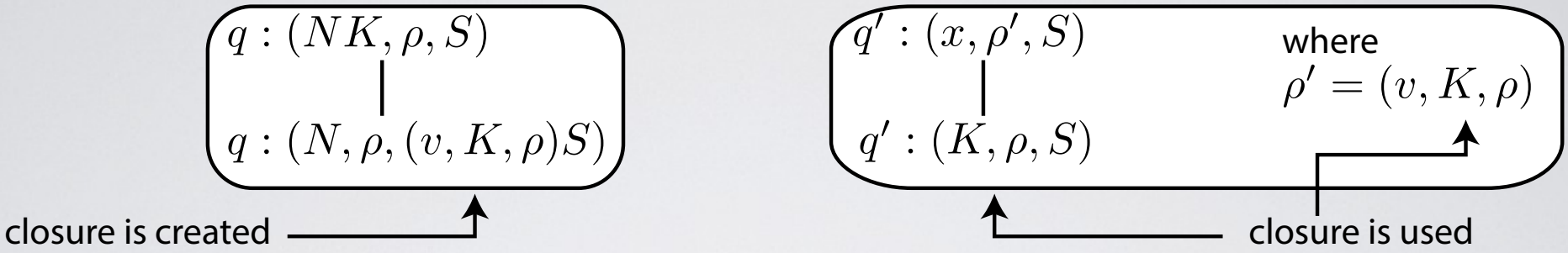
$q_0 : (N_0, \rho, \bot)$ $\quad$ $q_1 : (N_1, \rho, \bot)$

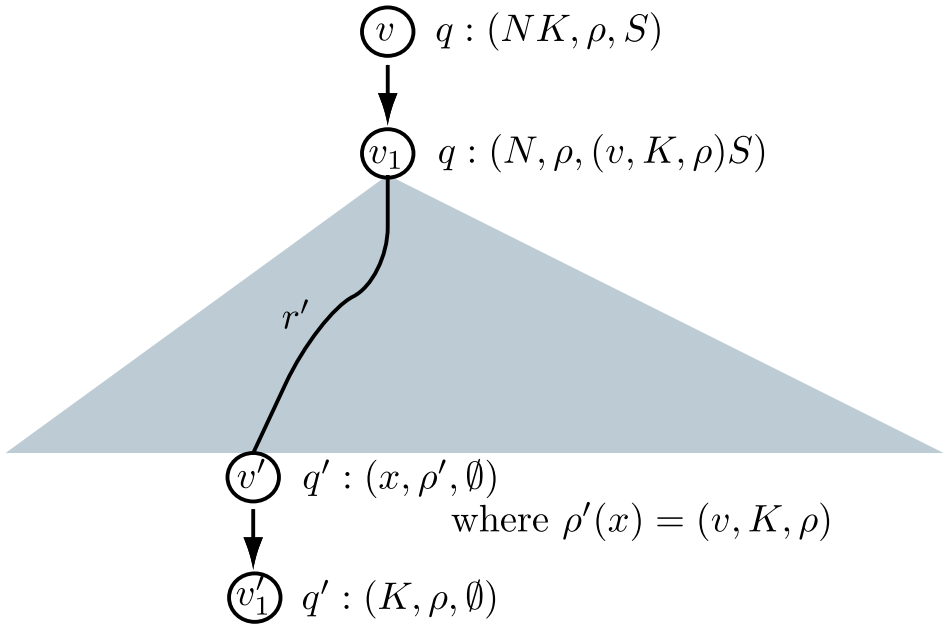
$q : (YN, \rho, S)$

$q : (N(YN), \rho, S)$

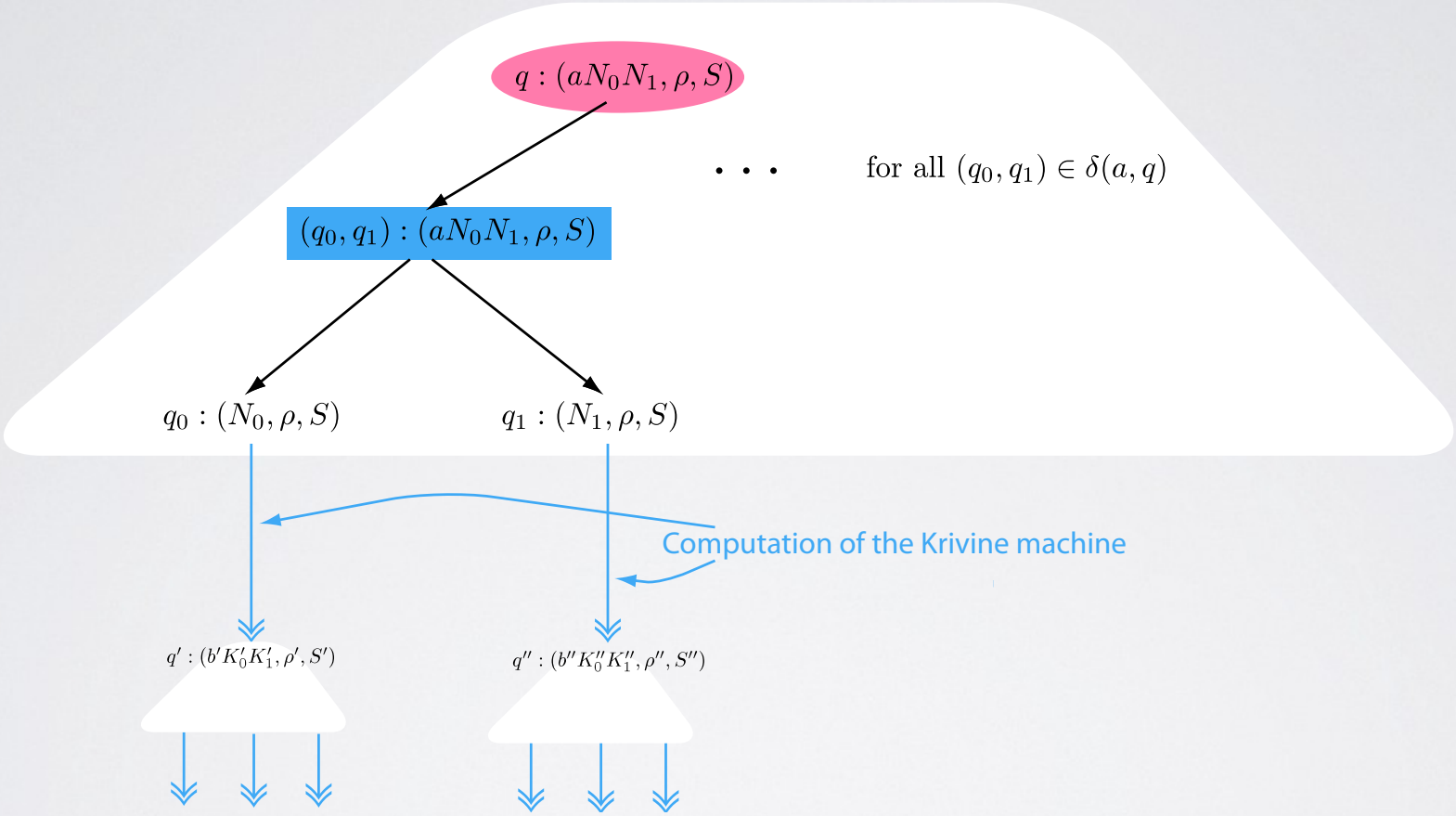
Definition of $\mathcal{K}(\mathcal{A}, M)$




$q : (NK, \rho, S)$
$q : (N, \rho, (v, K, \rho)S)$
closure is created
$q' : (x, \rho', S)$
$q' : (K, \rho, S)$
where
$\rho' = (v, K, \rho)$
closure is used

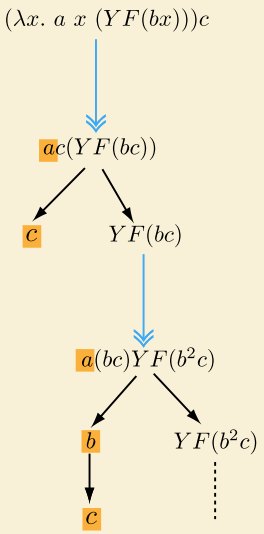

v
$q : (NK, \rho, S)$
v_1
$q : (N, \rho, (v, K, \rho)S)$
r'
v'
$q' : (x, \rho', \emptyset)$
where $\rho'(x) = (v, K, \rho)$
v'_1
$q' : (K, \rho, \emptyset)$

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

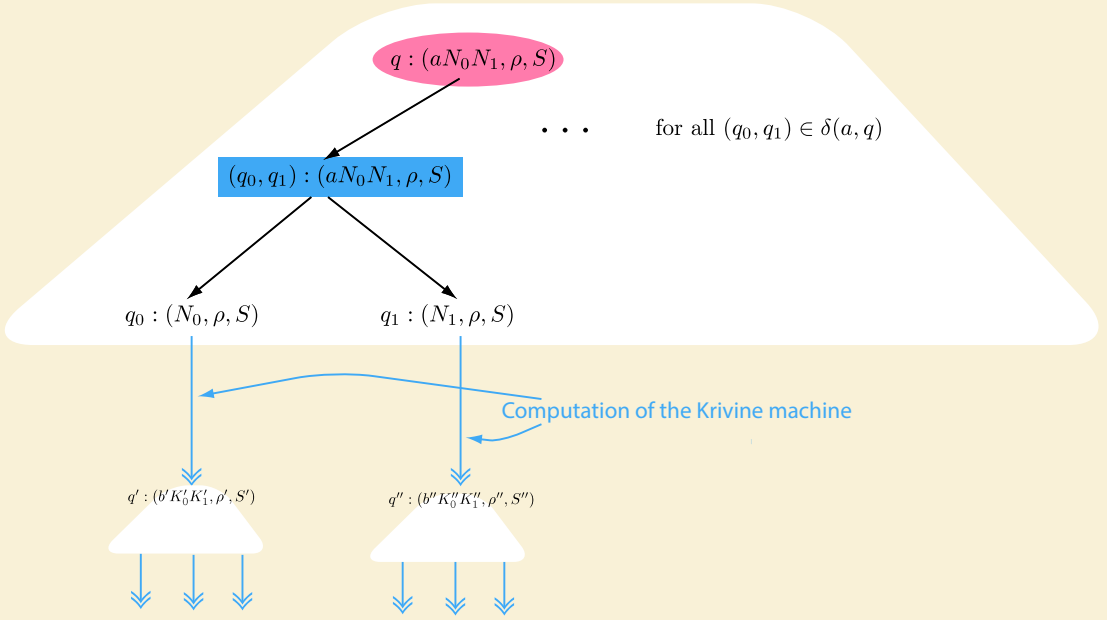


Proposition: For every closed λY-term M of type 0:
$BT(M) = Ktree(M, \emptyset, \bot)$.

1. Krivine machine calculating $BT(M)$



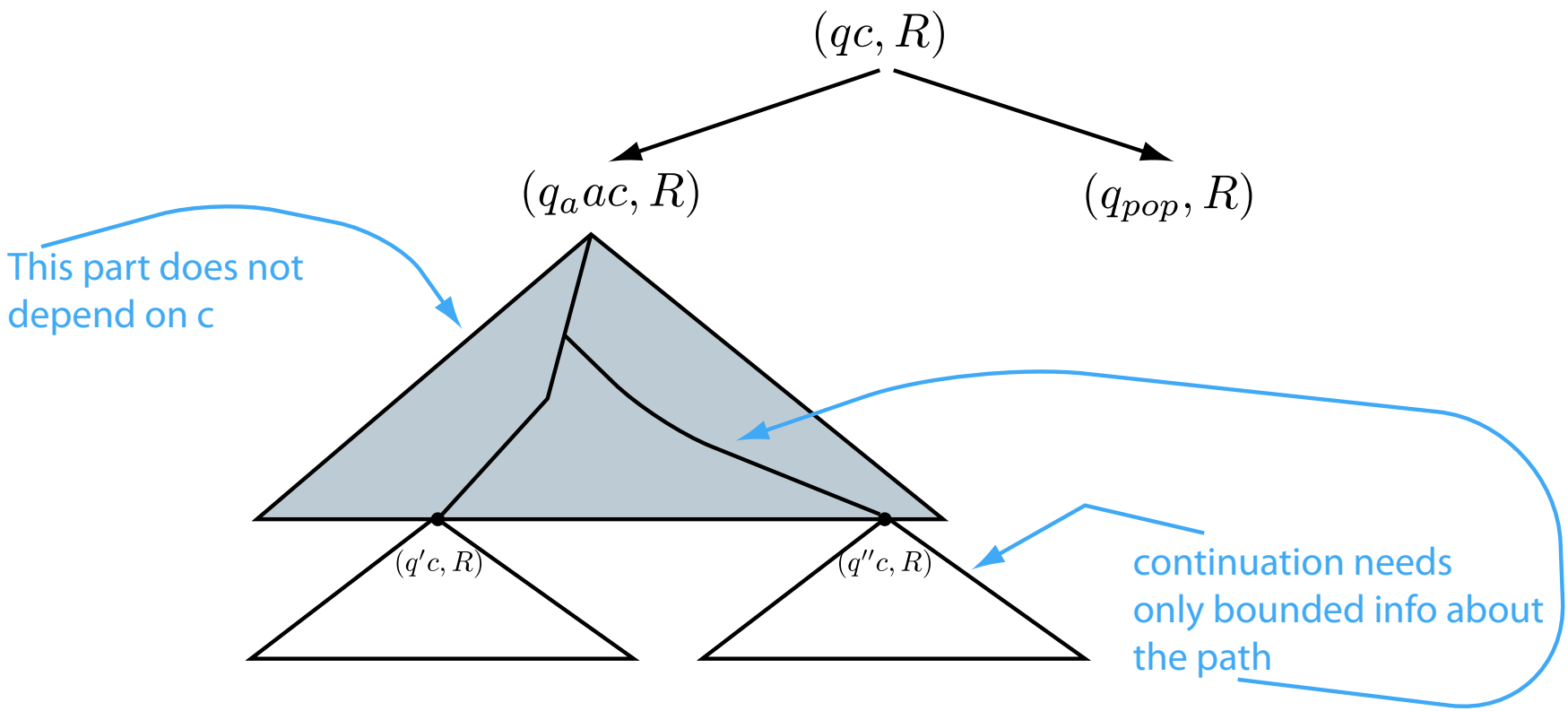
2. Acceptance in terms of a game $\mathcal{K}(\mathcal{A}, M)$



3. Reduction of $\mathcal{K}(\mathcal{A}, M)$ to $G(\mathcal{A}, M)$

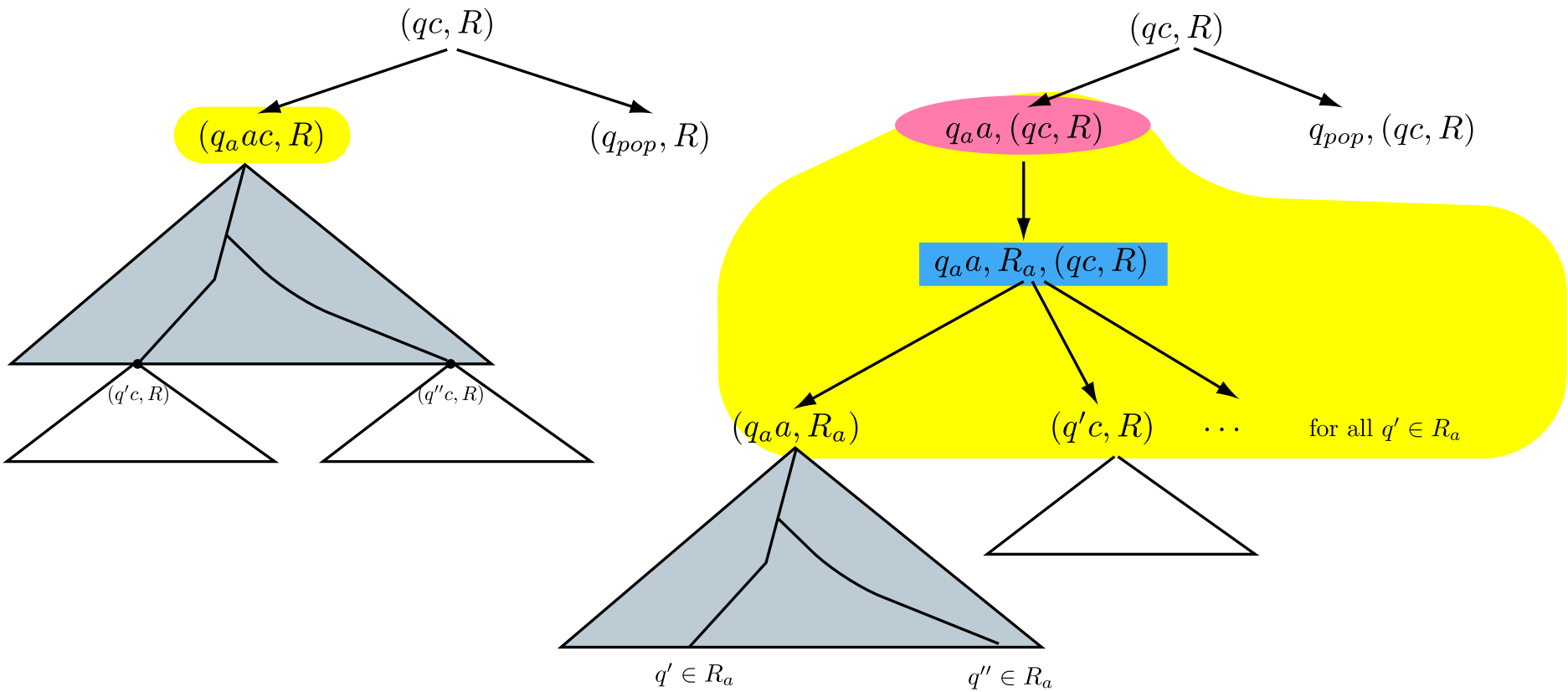
Decomposition property for a pushdown

$$qc \mapsto q_a ac \qquad qc \mapsto q_{pop}$$



Reduction to a finite game

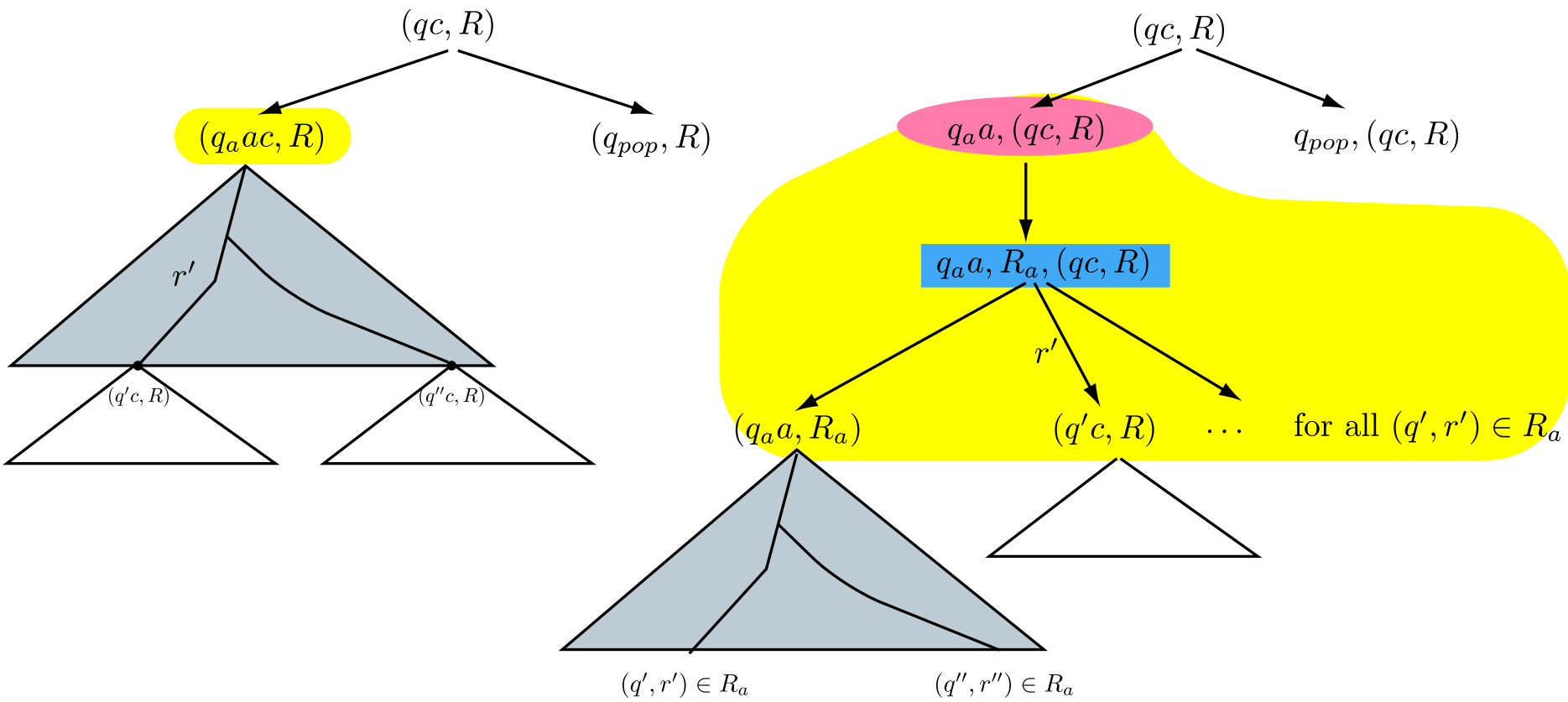
$$qc \mapsto q_a ac \qquad qc \mapsto q_{pop}$$



$$R_a \subseteq Q$$

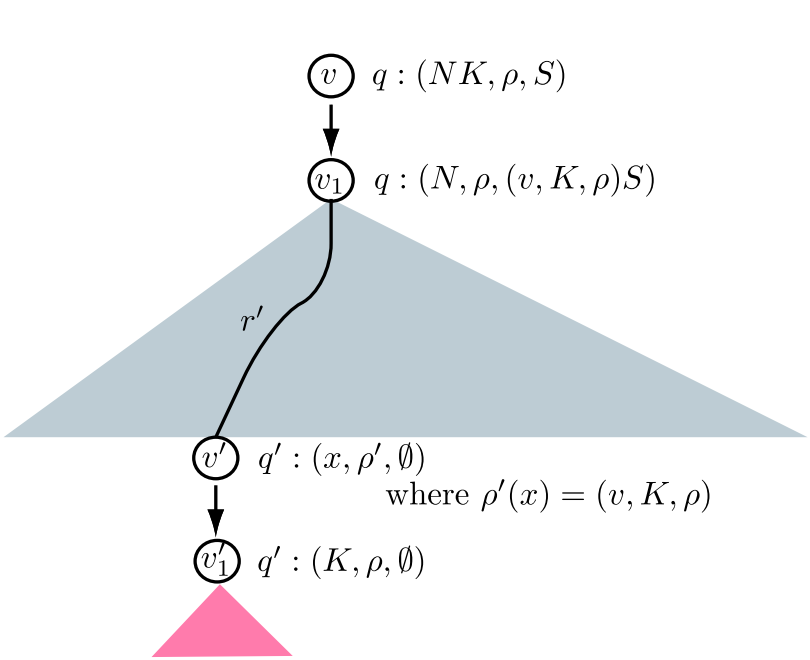
Reduction to a finite game (with ranks)

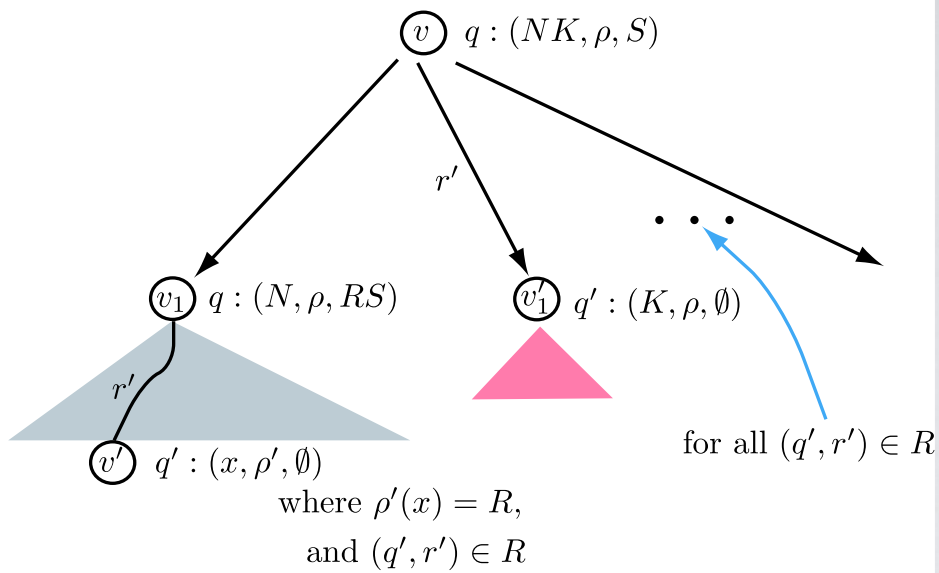
$$qc \mapsto q_a ac \qquad qc \mapsto q_{pop}$$



$$R_a \subseteq Q \times ranks$$

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 $\mathcal{P}(Q \times [d]) \to \mathcal{P}(Q \times [d])$.

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

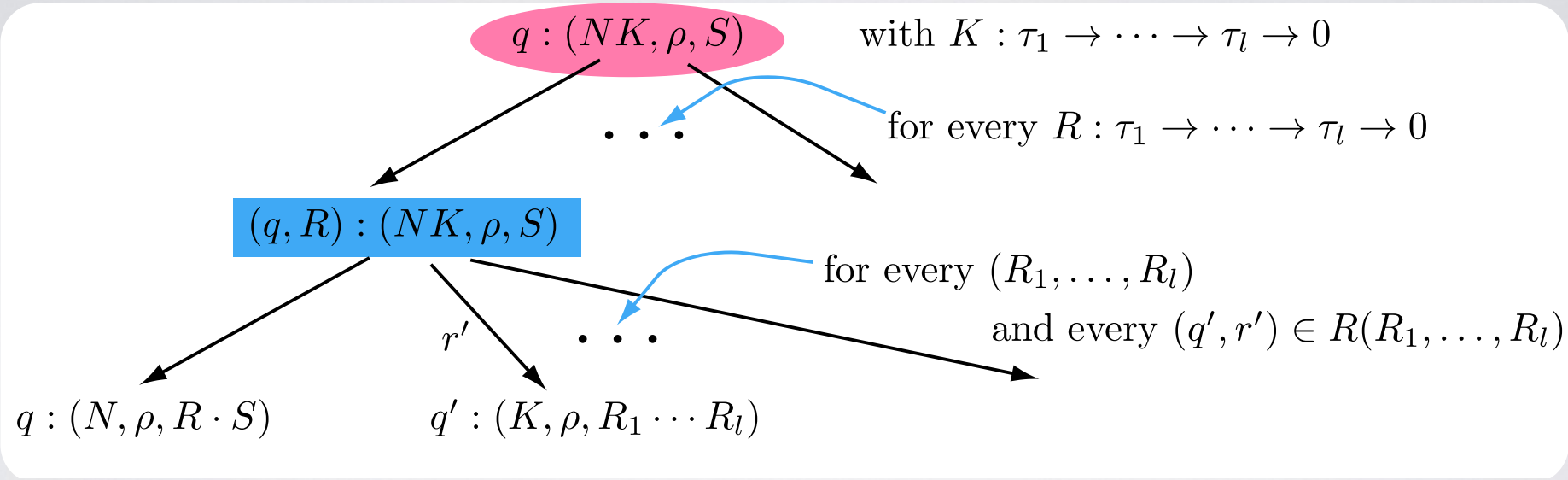
$$q : (\lambda x.N, \rho, R \cdot S) \to q : (N, \rho[x \mapsto R], S)$$

$$q : (a(N_0, N_1), \rho, \bot) \to (q_0, q_1) : (a(N_0, N_1), \rho, \bot) \quad \text{for } (q_0, q_1) \in \delta(q, a)$$

$$(q_0, q_1) : (a(N_0, N_1), \rho, \bot) \to q_i : (N_i, \rho \restriction_{rk(q_i)}, \bot) \qquad \text{for } i = 0, 1$$

$$q : (YN, \rho, S) \to q : (N(YN), \rho, S)$$

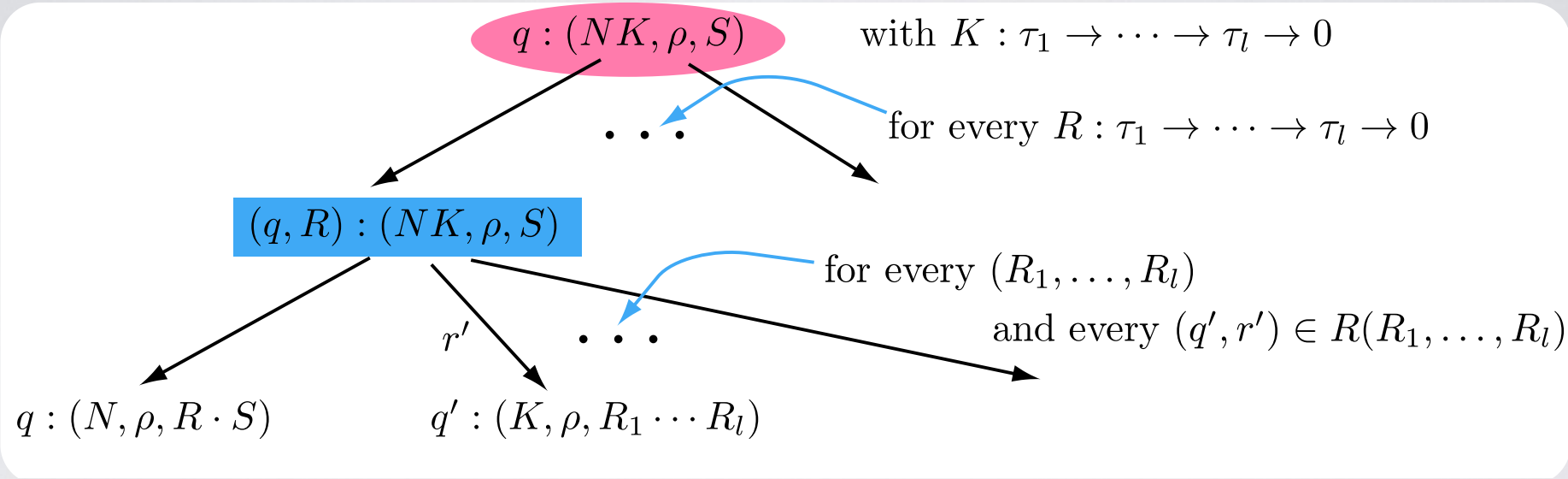
$G(\mathcal{A}, M)$



Eve wins in a position:

- $q : (x, \rho, S)$ if $(q, rk(q)) \in \rho(x)(S)$.

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

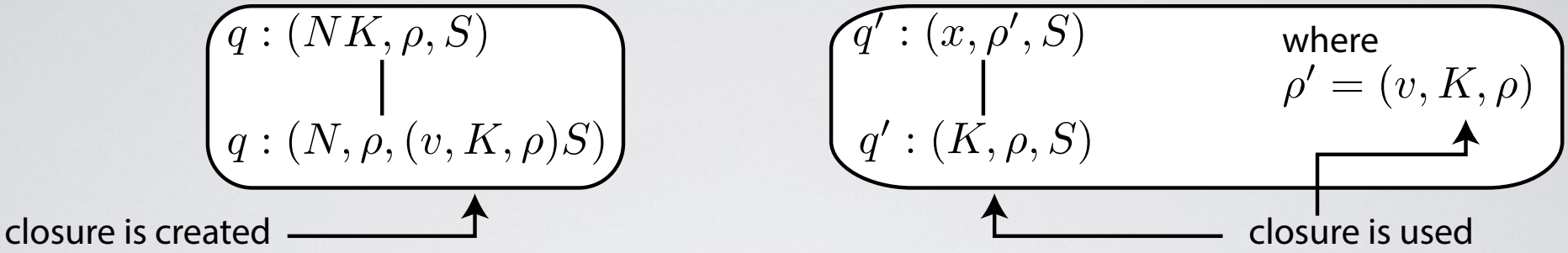


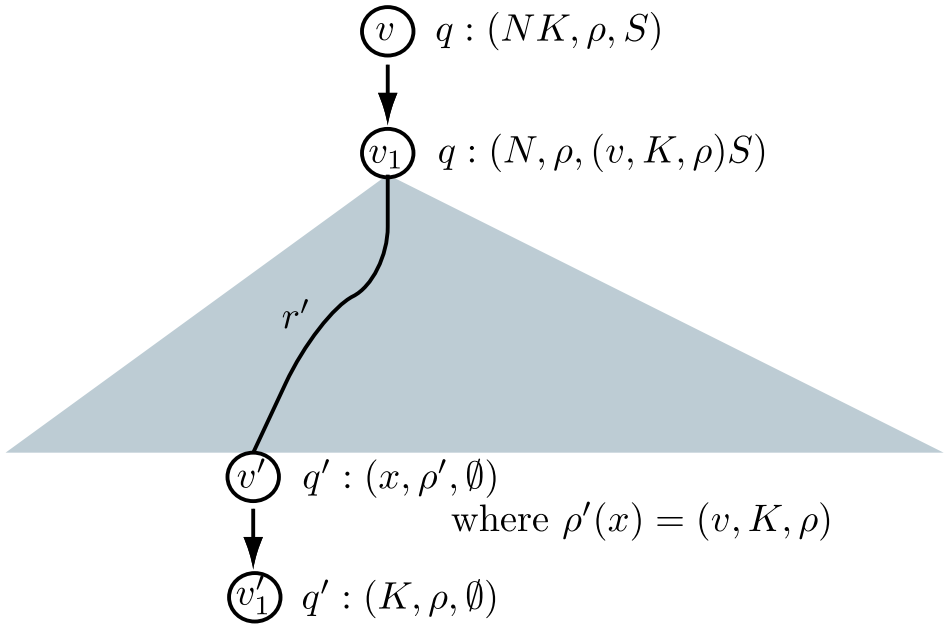
Obs:
For every N there are finitely many nodes in $G(\mathcal{A}, M)$ containing N.

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

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

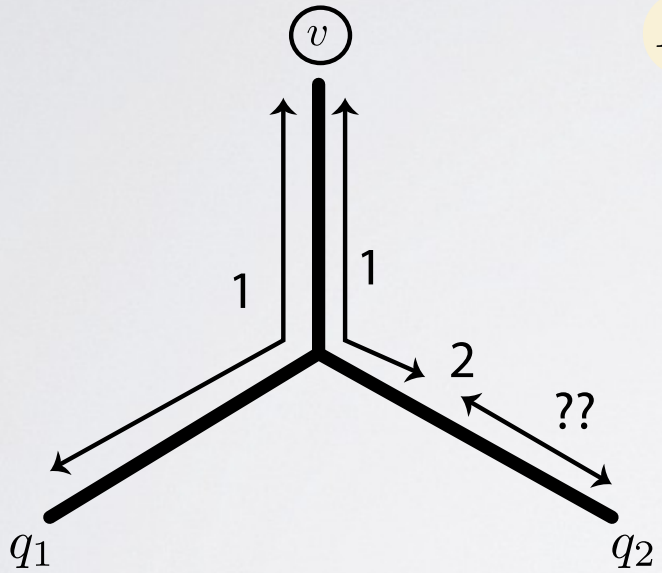
Proof:





Residual $R(v) = \{(q', r'), \dots\}$

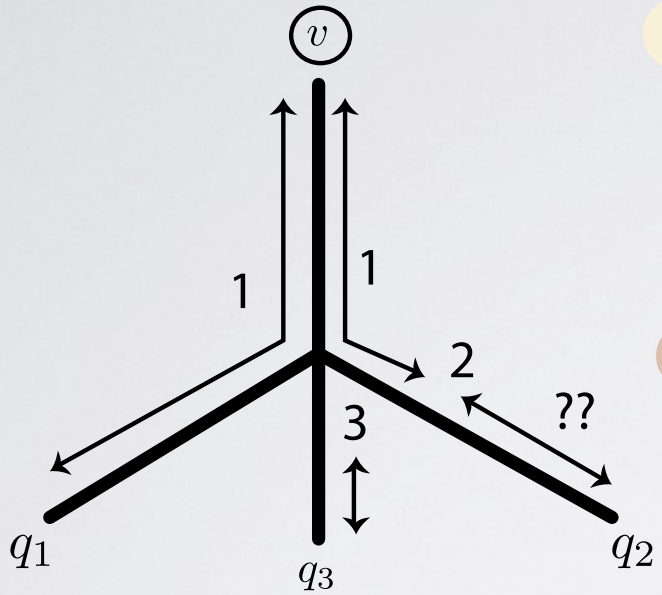
Adjusted residual $R(v)\downharpoonright_r$



$R(v) = \{(q_1, 1),\ (q_2, 2)\}$

$R(v)\downharpoonright_2 = \{(q_2, 1),\ (q_2, 2)\}$

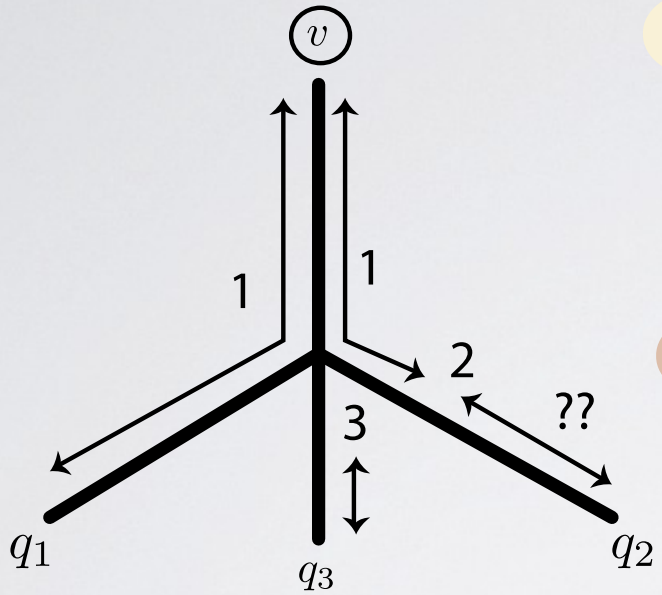
Adjusted residual $R(v)\downharpoonright_r$



$R(v) = \{(q_1, 1),\ (q_2, 2),\ (q_3, 3)\}$

$R(v)\downharpoonright_2 = \{(q_2, 1),\ (q_2, 2),\ (q_3, 3)\}$

Adjusted residual $R(v)\downarrow_r$

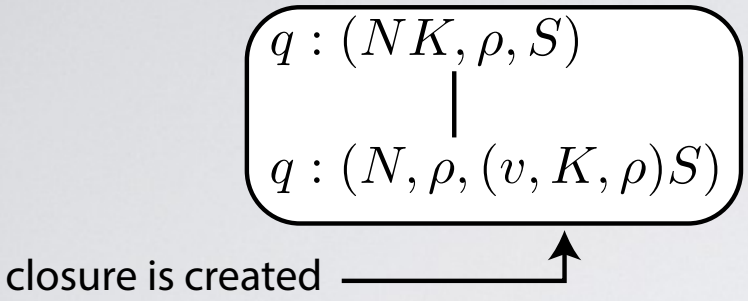


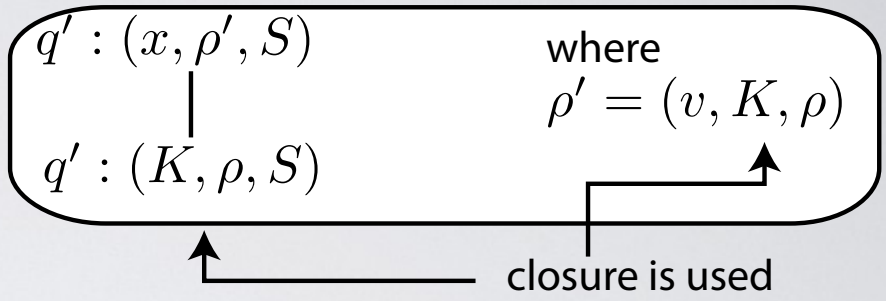
$R(v) = \{(q_1, 1),\ (q_2, 2),\ (q_3, 3)\}$

$R(v)\downarrow_2 = \{(q_2, 1),\ (q_2, 2),\ (q_3, 3)\}$

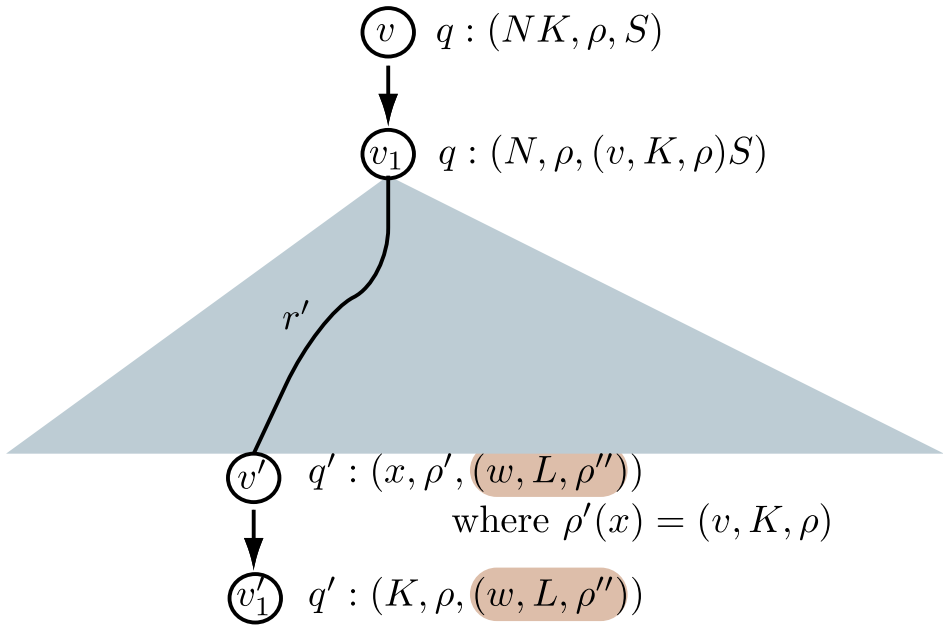
Notation : $res(v, v_1) = R(v)\downarrow_{\max(v,v_1)}$

- $res(C, v_1) = R(v)\downarrow_{\max(v,v_1)}$ where $C = (v, L, \rho)$,
- $res(\rho, v_1) = \rho_1$ such that $\rho_1(x) = res(\rho(x), v_1)$,
- $res(S, v_1) = R_1 \ldots R_k$ where $R_i = res(C_i, v_1)$, and $S = C_1 \ldots C_k$.





When $K : 0 \to 0$

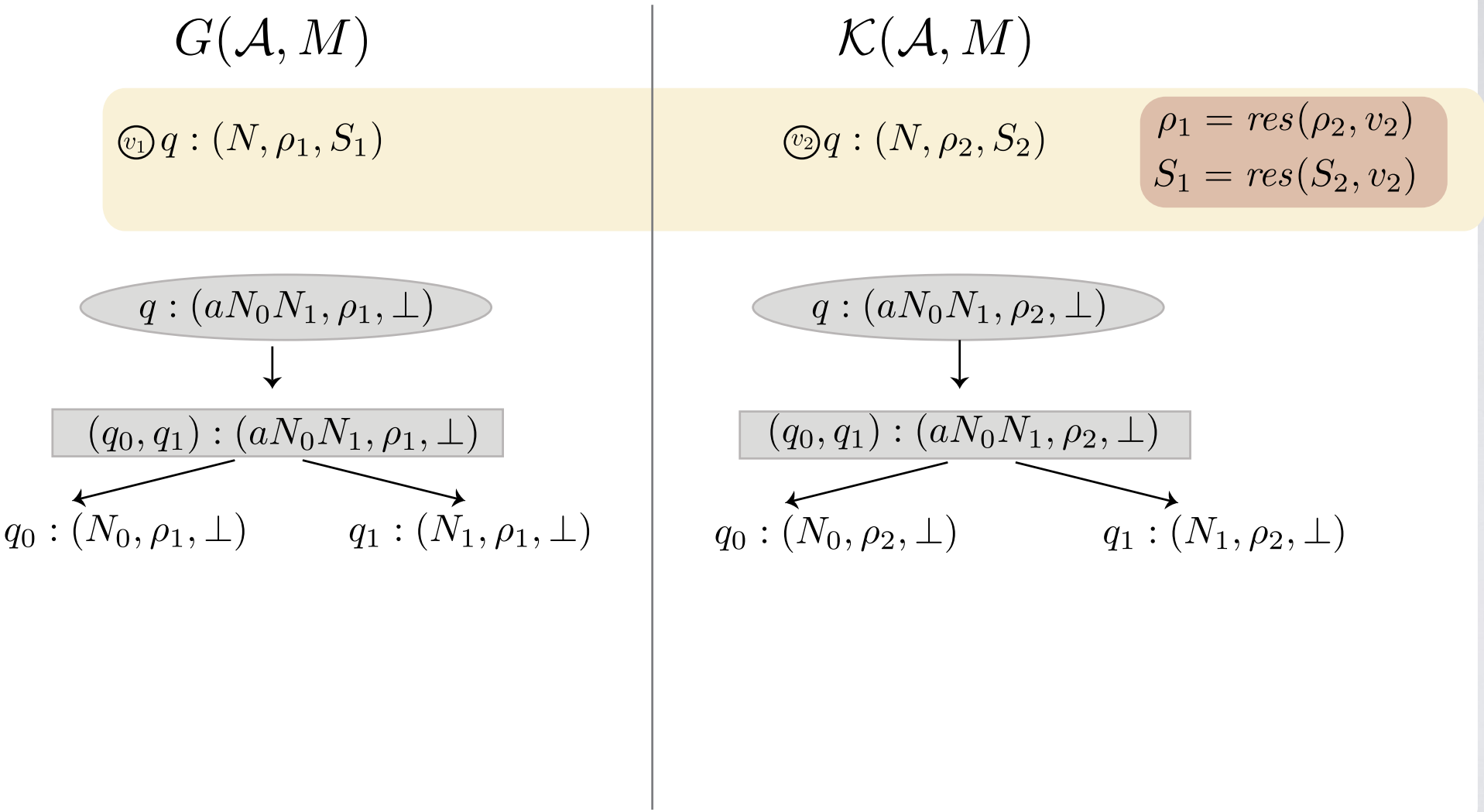


Residual $R(v)(R(w)\downarrow_{r''}) = \{(q', r'), \dots\}$ where $r'' = \max(w, v')$

Transferring Eve's strategy from $\mathcal{K}(\mathcal{A}, M)$ to $G(\mathcal{A}, M)$

$G(\mathcal{A}, M)$	$\mathcal{K}(\mathcal{A}, M)$	
$(v_1)\ q : (N, \rho_1, S_1)$	$(v_2)\ q : (N, \rho_2, S_2)$	$\rho_1 = res(\rho_2, v_2)$ $S_1 = res(S_2, v_2)$
$q : (\lambda x.N,\ \rho_1,\ RS_1)$	$q : (\lambda x.N,\ \rho_2,\ CS_2)$	$R = res(C, v_2)$
$\downarrow$	$\downarrow$	
$q : (N,\ \rho_1[x \mapsto R],\ S_1)$	$q : (N,\ \rho_2[x \mapsto C],\ S_2)$	

Transferring Eve's strategy from $\mathcal{K}(\mathcal{A}, M)$ to $G(\mathcal{A}, M)$



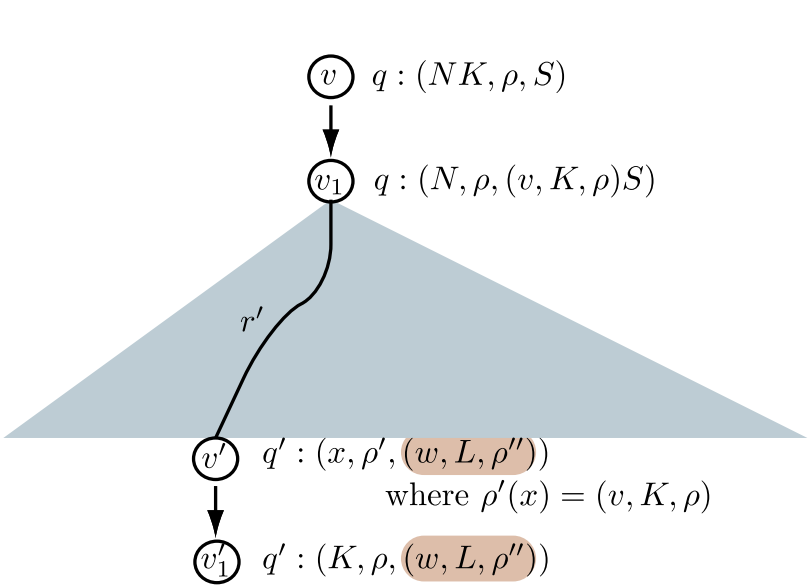
Transferring Eve's strategy from $\mathcal{K}(\mathcal{A}, M)$ to $G(\mathcal{A}, M)$

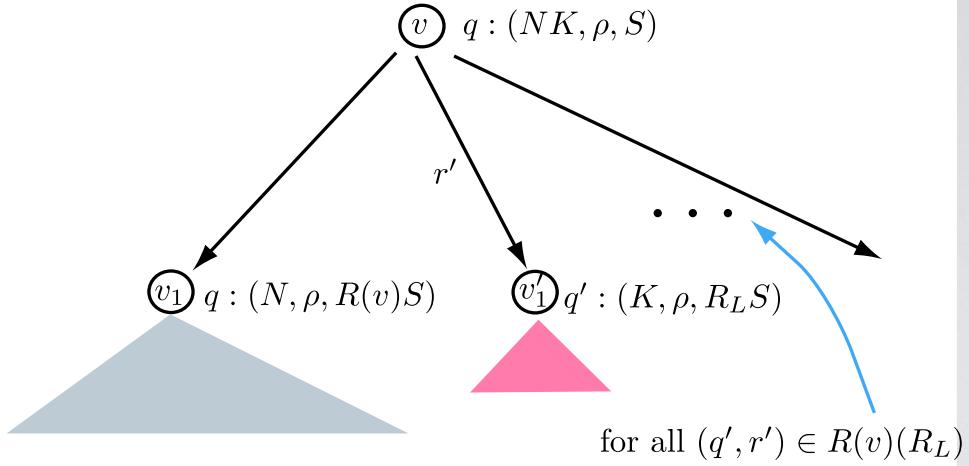
$G(\mathcal{A}, M)$	$\mathcal{K}(\mathcal{A}, M)$	
v_1 $q : (N, \rho_1, S_1)$	v_2 $q : (N, \rho_2, S_2)$	$\rho_1 = res(\rho_2, v_2)$ $S_1 = res(S_2, v_2)$
v_1 $q : (x,\ \rho_1,\ S_1)$	v_2 $q : (x,\ \rho_2,\ S_2)$	

We want to show $(q, rk(q)) \in \rho_1(x)(S_1)$.
Suppose $\rho_2(x) = (v, K, \rho)$

- $(q, \max(v, v_2)) \in R(v)(res(S_2, v_2))$ by def.
- $(q, \max(v, v_2)) \in R(v)(res(S_2, v_2))\restriction_{\max(v,v_2)}$ by prop of $\restriction$
- $(q, rk(q)) \in R(v)(res(S_2, v_2))\restriction_{\max(v,v_2)}$ by prop of $\restriction$
- $(q, rk(q)) \in \rho_1(x)(S_1)$ by def.

Decomposition property

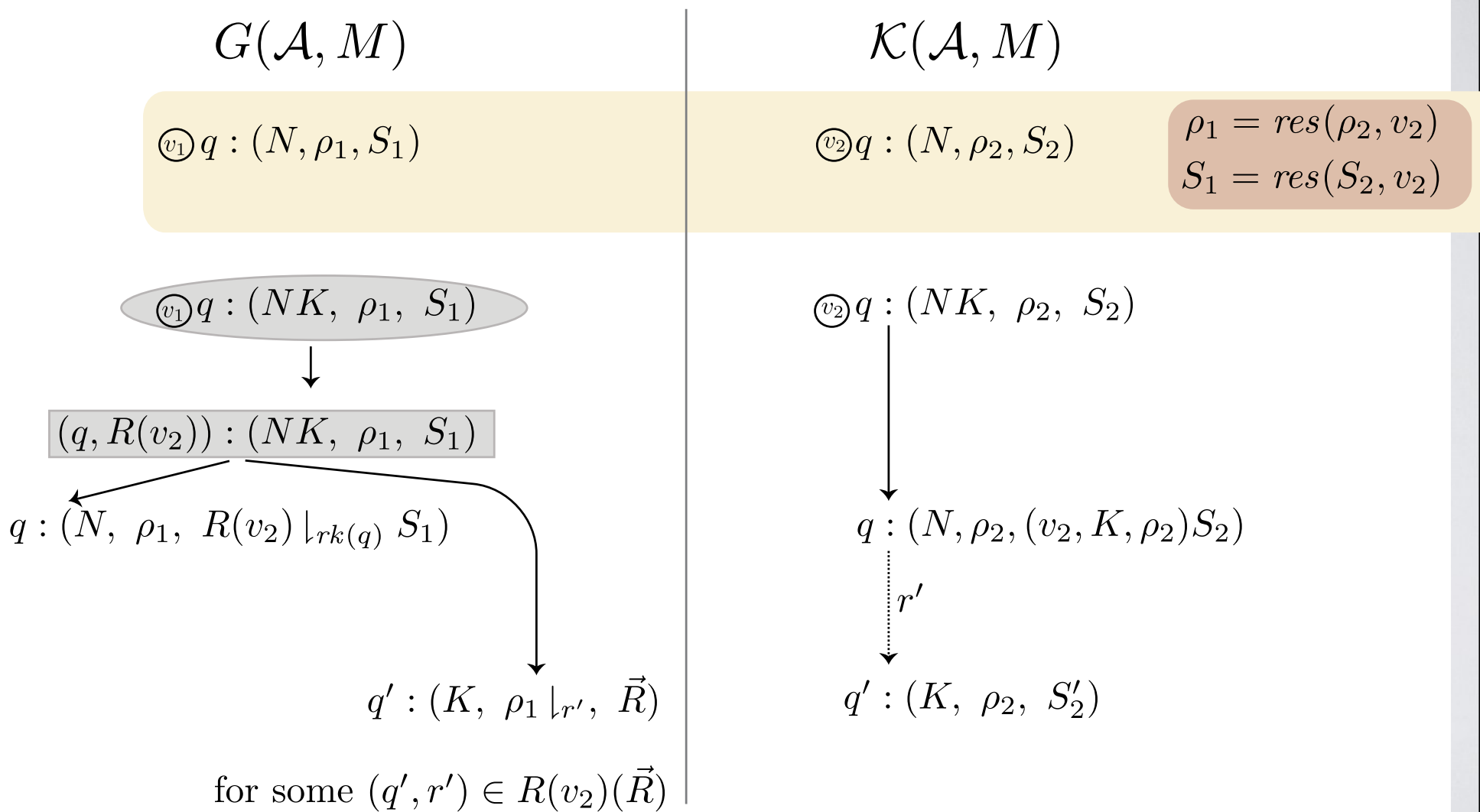




Closure $(v, K^{0\to 0}, \rho)$ is replaced by $R(v) : \mathcal{P}(Q \times [d]) \to \mathcal{P}(Q \times [d])$.
We put (q', r') in $R(v)(R_L)$.

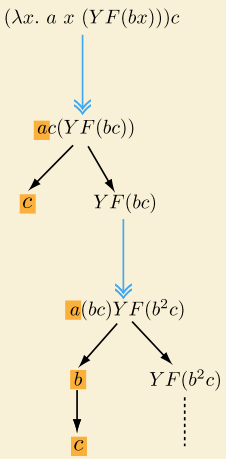
We use induction on types.

Transferring Eve's strategy from $\mathcal{K}(\mathcal{A}, M)$ to $G(\mathcal{A}, M)$

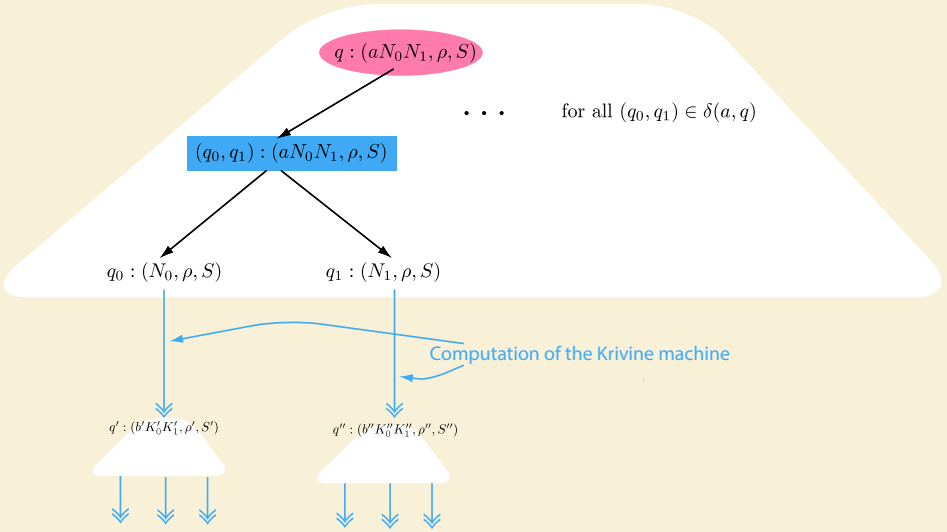



orange

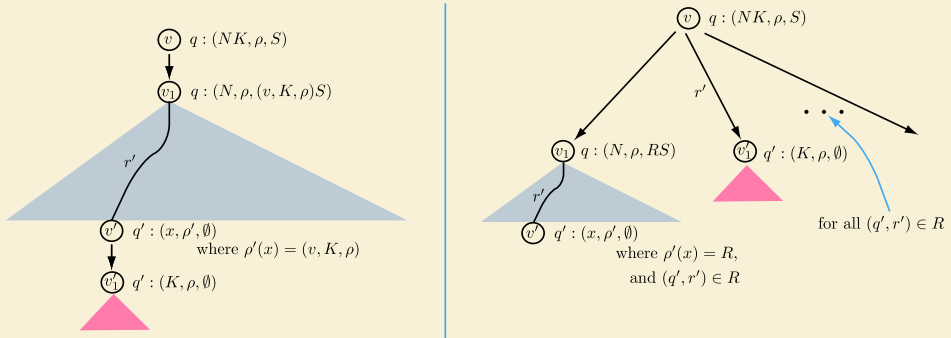
1. Krivine machine calculating $BT(M)$



2. Acceptance in terms of a game $\mathcal{K}(\mathcal{A}, M)$



3. Reduction of $\mathcal{K}(\mathcal{A}, M)$ to $G(\mathcal{A}, M)$



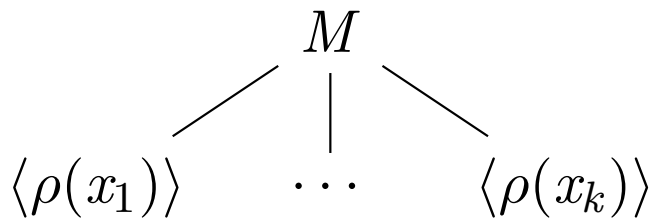
Thm: Eve wins in $G(\mathcal{A}, M)$ iff Eve wins in $\mathcal{K}(\mathcal{A}, M)$.

Obs: $G(\mathcal{A}, M)$ is finite.

Global model checking

Representing configurations of a Krivine machine

For closures: $\langle (M, \rho) \rangle$ is



For configurations: $\langle (M, \rho, S_1 \dots S_l) \rangle$ is $\langle (M, \rho) \rangle, \langle S_1 \rangle, \dots, \langle S_1 \rangle$.

Theorem

For every M and $\mathcal{A}$, the set:

$$\{\langle N, \rho, S \rangle \ : \ BT(E(N, \rho, S)) \in L(\mathcal{A})\}$$

is a regular language of finite trees.

Transfer Theorem

$$\begin{array}{ccc} M & \xrightarrow{eval} & BT(M) \\ \widehat{\varphi} & \leftarrow & \varphi \end{array}$$

Transfer Theorem

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

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

$$M \xrightarrow{eval} BT(M)$$
$$\widehat{\varphi} \leftarrow \varphi$$

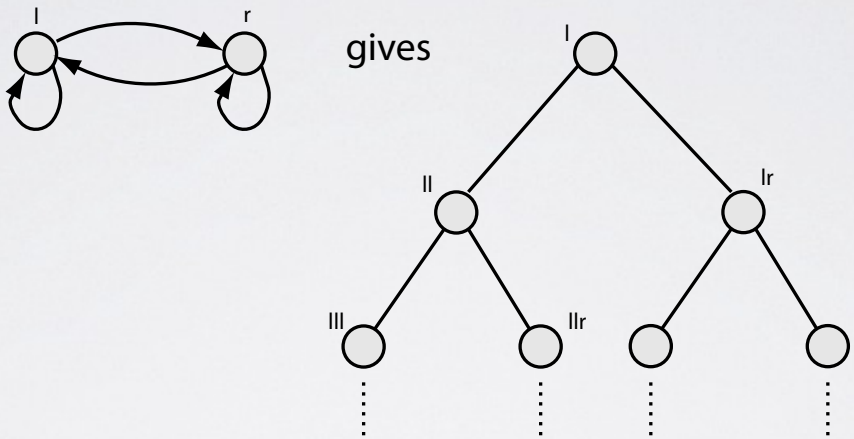
Transfer Theorem

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

Graph $\xrightarrow{unfold}$ 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 $M \in Terms(\Sigma, \mathcal{T}, \mathcal{X})$:

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

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

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

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

Transfer Theorem

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

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

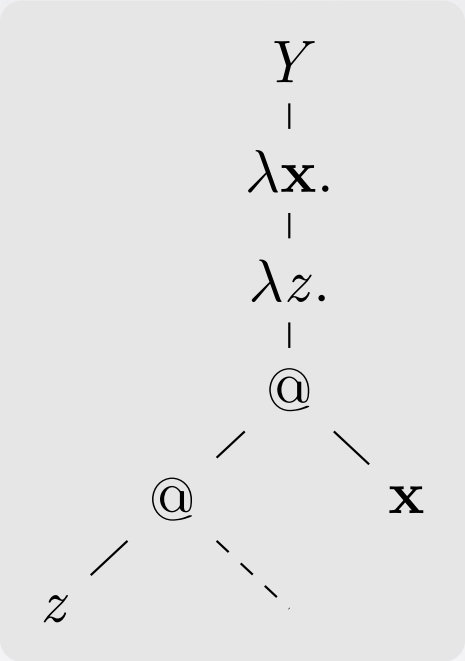
- Σ is a tree signature
- $\mathcal{T}$ is a finite set of terms
- $\mathcal{X}$ is a finite set of λ-variables
- $Terms(\Sigma, \mathcal{T}, \mathcal{X})$: terms over Σ with
  - all subterms having type in $\mathcal{T}$,
  - all λ-variables from $\mathcal{X}$.

Note: Theorem works also for infinite λY-terms, and unobunded number of Y variables.

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

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

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



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

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

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

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

Consequences of the transfer theorem

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

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

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 $M \models \widehat{\varphi}$.

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

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

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:

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

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. Decide if one can construct a λY term K from these terms such that $BT(K) \models \varphi$.

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.

Two ways looking at it.

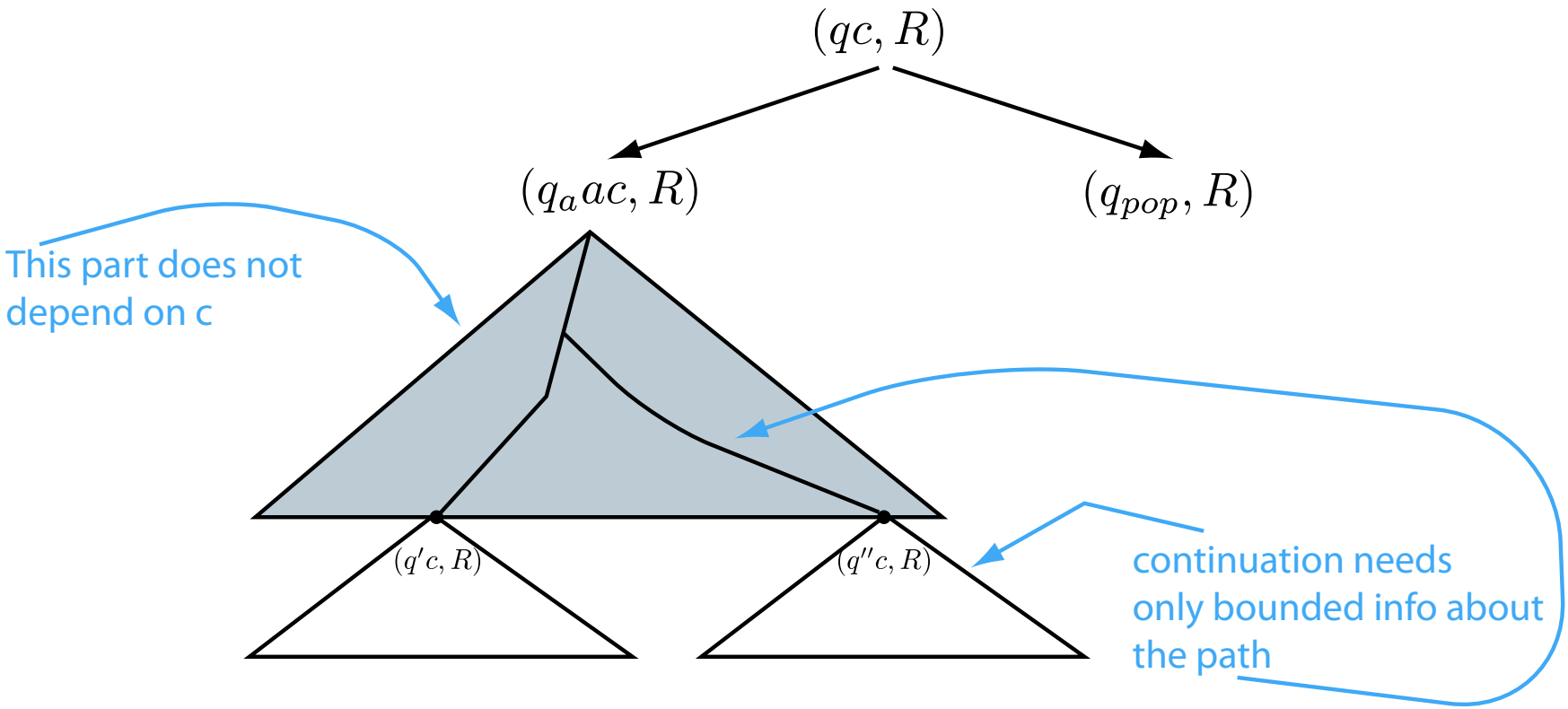
Studding new properties of evaluation in simple types.

Bringing verification to a new ground.

In the beginner's mind there are many possibilities, in the expert's mind there are few.

Decomposition property for a pushdown:

$$qc \mapsto q_a ac \qquad qc \mapsto q_{pop}$$



$$M \xrightarrow{eval} BT(M) \stackrel{?}{\in} L(\mathcal{A})$$

- Understanding $BT(M) \stackrel{?}{\in} L(\mathcal{A})$ in terms of models

$$[\![M]\!] = [\![BT(M)]\!]$$

- Understanding $BT(M) \stackrel{?}{\in} L(\mathcal{A})$ in terms of $\mathcal{K}(\mathcal{A}, M)$

$$\mathcal{K}(\mathcal{A}, M) \quad \text{equivalent to} \quad G(\mathcal{A}, M)$$

- MSO compatibility of evaluation

$$\begin{array}{ccc} M & \xrightarrow{eval} & BT(M) \\ \widehat{\varphi} & \leftarrow & \varphi \end{array}$$

- Getting closer to “real” computation.
- Transfer theorem covers: Rabin’s theorem, unfolding theorem, pushdown hierarchy, Ong’s theorem, global model-checking,. . .
- The use of old techniques in a new way.