Functional Reachability

Luke Ong Nikos Tzevelekos

Oxford University Computing Laboratory

24th Symposium on Logic in Computer Science
Los Angeles, August 2009.

The Problem

Reachability in functional computation.

- Consider a term M of a higher-order functional programming language.
- Now consider a point p inside M.
- Is there a program context C such that the computation of $C[M]$ reaches p?

The Problem

Reachability in functional computation.

- Consider a term M of a higher-order functional programming language.
- Now consider a point p inside M.
- Is there a program context C such that the computation of $C[M]$ reaches p?

The Problem

Reachability in functional computation.

- Consider a term M of a higher-order functional programming language.
- Now consider a point p inside M.
- Is there a program context C such that the computation of $C[M]$ reaches p?

Surprisingly, (Contextual) Reachability *per se* had not been studied in HO functional languages.

Relevant work

- Control Flow Analysis: *Approximate at compile time the flow of control to happen at run time.*
  - In a HO-setting, the crucial element is that of *closures*.
  - Reynolds ('70), Jones ('80), Shivers ('90), . . . Malacaria & Hankin (late 90's).
  - CFA $>$ Reach: more general.
    Reach $>$ CFA: open vs closed world approach.
- Useless code detection, etc.

The examined language: PCF

Types: $A, B ::= o \mid A \to B$

Terms: $M, N ::= x \mid \lambda x.M \mid M N \mid \mathsf{t} \mid \mathsf{f} \mid \mathsf{if}\, M\, N_1\, N_2 \mid \mathbf{Y}_A$

Contexts: $C ::= \ldots$

The examined language: PCF

Types: $A, B \ ::= o \mid A \to B$

Terms: $M, N \ ::= x \mid \lambda x.M \mid M\,N \mid \mathsf{t} \mid \mathsf{f} \mid \mathsf{if}\, M\, N_1\, N_2 \mid \mathbf{Y}_A$

Contexts: $C \ ::= \dots$

Reductions:
$$\begin{array}{rcl@{\qquad}rcl} (\lambda x.M)N & \to & M\{N/x\} & \mathsf{if}\,\mathsf{t} & \to & \lambda xy.x \\ \mathbf{Y}M & \to & M(\mathbf{Y}M) & \mathsf{if}\,\mathsf{f} & \to & \lambda xy.y \end{array}$$

$$M \to N \implies \mathrm{E}[M] \to \mathrm{E}[N]$$

Ev. Contexts: $\mathrm{E} \ ::= [-] \mid \mathrm{E}\,M \mid \mathsf{if}\,\mathrm{E}$

The examined language: PCF

Types: $A, B ::= o \mid A \to B$

Terms: $M, N ::= x \mid \lambda x.M \mid MN \mid \mathsf{t} \mid \mathsf{f} \mid \mathsf{if}\ M\ N_1\ N_2 \mid \mathbf{Y}_A$

Contexts: $C ::= \ldots$

Reductions: Call-by-name λ-calculus + if + $\mathbf{Y}$

- Write $(A_1, \ldots, A_n, o)$ for $A_1 \to \cdots \to A_n \to o$.
- Divergence definable, e.g. $\bot := \mathbf{Y}_o(\lambda x.x)$.
- *Finitary* restrictions (i.e. no $\mathbf{Y}$): fPCF, fPCF$_\bot$.

Reachability

- Given a PCF-term M and a coloured subterm L of M,
- Is there a program context C such that $C[M] \twoheadrightarrow \mathrm{E}[L']$ with L' coloured?

- Given a PCF-term M and a coloured subterm L of M,
- Is there a program context C such that $C[M] \twoheadrightarrow \mathrm{E}[L']$ with L' coloured?

Equivalently:

- Given a closed PCF-term $M : (A_1, ..., A_n, o)$ and a coloured subterm L of M,
- Are there closed PCF-terms $N_1, \ldots, N_n$ such that

$$M\vec{N} \twoheadrightarrow \mathrm{E}[L']$$

with L' coloured?

PCF-with-error: PCF*

Take base type $o = \{\mathsf{t}, \mathsf{f}, \star\}$ with $\star$ an error constant:

$$\mathrm{E}[\star] \twoheadrightarrow \star$$

$\star$-Reachability:

- Given a closed PCF*-term $M : (A_1, \ldots, A_n, o)$ that has exactly one occurrence of $\star$,
- are there closed PCF-terms $N_1, \ldots, N_n$ such that $M\vec{N} \twoheadrightarrow \star$?

PCF-with-error: PCF*

Take base type $o = \{\mathrm{t}, \mathrm{f}, \star\}$ with $\star$ an error constant:

$$\mathrm{E}[\star] \twoheadrightarrow \star$$

$\star$-Reachability:

- Given a closed PCF*-term $M : (A_1, ..., A_n, o)$ that has exactly one occurrence of $\star$,
- are there closed PCF-terms $N_1, ..., N_n$ such that $M\vec{N} \twoheadrightarrow \star$?

Lemma: Reachability $\cong$ $\star$-Reachability.

For $v \in \{\mathsf{t}, \mathsf{f}, \star\}$ and $\mathcal{L}_1, \mathcal{L}_2 \subseteq \mathsf{PCF}^\star$:

v-$\textsc{Reach}\,[\mathcal{L}_1, \mathcal{L}_2]$:
Given a closed $\mathcal{L}_1$-term $M : (A_1, ..., A_n, o)$, are there closed $\mathcal{L}_2$-terms $N_1, ..., N_n$ such that $M\vec{N} \twoheadrightarrow v$?

e.g. $\star$-Reachability $=$ $\star$-$\textsc{Reach}\,[\mathsf{PCF}^{1\star}, \mathsf{PCF}]$.

Reach template

For $v \in \{\mathsf{t}, \mathsf{f}, \star\}$ and $\mathcal{L}_1, \mathcal{L}_2 \subseteq \mathsf{PCF}^\star$:

v-$\textsc{Reach}\,[\mathcal{L}_1, \mathcal{L}_2]$:
Given a closed $\mathcal{L}_1$-term $M : (A_1, ..., A_n, o)$, are there closed $\mathcal{L}_2$-terms $N_1, ..., N_n$ such that $M\vec{N} \twoheadrightarrow v$?

Three classes of problems:

Reachability
$\star$-Reachability
$\star$-$\textsc{Reach}\,[\mathsf{PCF}^{1\star}, \mathsf{PCF}]$
$\star$-$\textsc{Reach}\,[\mathsf{PCF}^{1\star}, \mathsf{fPCF}]$

$\star$-$\textsc{Reach}\,[\mathsf{fPCF}^{1\star}, \mathsf{fPCF}]$
$\star$-$\textsc{Reach}\,[\mathsf{fPCF}^{\star}, \mathsf{fPCF}]$
$\ldots$

$\star$-$\textsc{Reach}\,[\mathsf{fPCF}^{1\star}_\perp, \mathsf{fPCF}]$
$\star$-$\textsc{Reach}\,[\mathsf{fPCF}^{\star}_\perp, \mathsf{fPCF}]$
$\ldots$

An undecidability result

Lemma: $\star$-$\textsc{Reach}\,[\mathsf{fPCF}^{1\star}_{\perp}, \mathsf{fPCF}]$ is undecidable.

Proof: By reduction of solvability of systems of $\mathsf{fPCF}_{\perp}$-equations (proved undecidable by [Loader'01]).

Reachability

$\star$-Reachability

$\star$-$\textsc{Reach}\,[\mathsf{PCF}^{1\star}, \mathsf{PCF}]$

$\star$-$\textsc{Reach}\,[\mathsf{PCF}^{1\star}, \mathsf{fPCF}]$

$\star$-$\textsc{Reach}\,[\mathsf{fPCF}^{1\star}, \mathsf{fPCF}]$

$\star$-$\textsc{Reach}\,[\mathsf{fPCF}^{\star}, \mathsf{fPCF}]$

$\ldots$

$\star$-$\textsc{Reach}\,[\mathsf{fPCF}^{1\star}_{\perp}, \mathsf{fPCF}]$

$\star$-$\textsc{Reach}\,[\mathsf{fPCF}^{\star}_{\perp}, \mathsf{fPCF}]$

$\ldots$

- We focus on v-$\textsc{Reach}\,[\mathsf{fPCF}^\star, \mathsf{fPCF}]$.
- For fPCF*-term $P : o$,

Computations of P

Traversals over its *computation tree*, $\lambda(P)$

Runs of an *Alternating Tree Automaton (ATA)* on $\lambda(P)$

Our approach

- We focus on v-$\textsc{Reach}\,[\mathsf{fPCF}^\star, \mathsf{fPCF}]$.
- For fPCF*-term $P : o$,

Computations of P

Traversals over its *computation tree*, $\lambda(P)$

Runs of an *Alternating Tree Automaton (ATA)* on $\lambda(P)$

- $P \twoheadrightarrow v$ iff an ATA accepts $\lambda(P)$ on initial state with *value* v.

Computation trees

Starting from a fPCF*-term M,

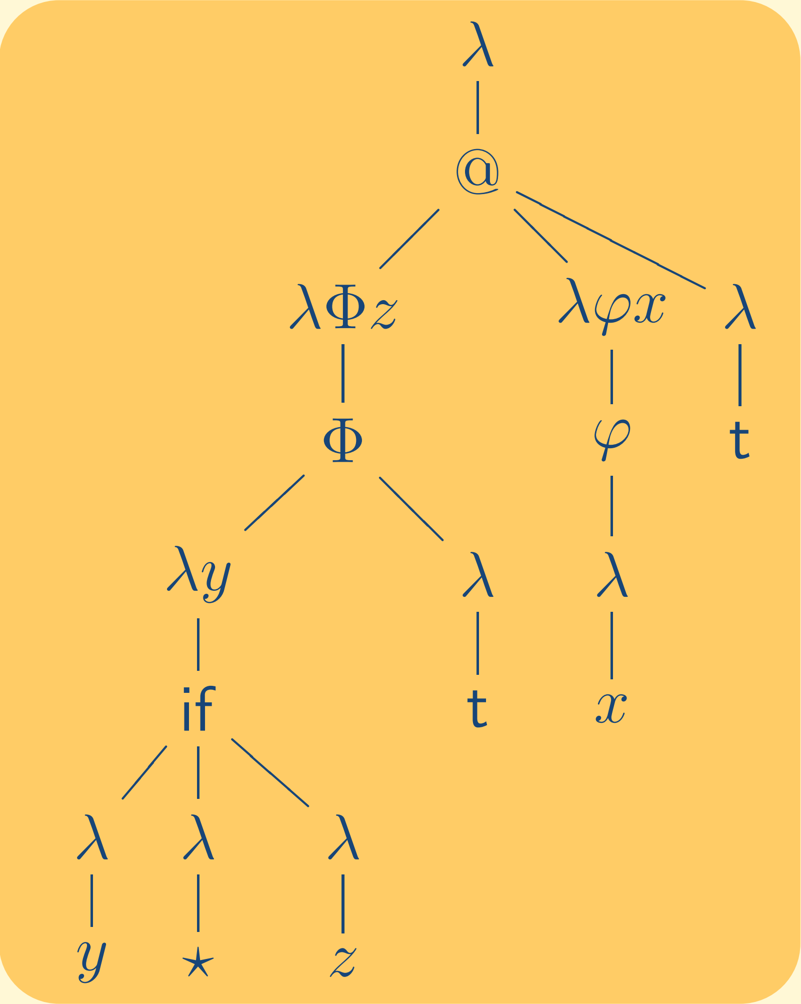
- take its η-long form,
- add application symbols ($@$),
- view the result as a tree, $\lambda(M)$.

Computation trees

Starting from a fPCF*-term M,

- take its η-long form,
- add application symbols (@),
- view the result as a tree, $\lambda(M)$.

$$(\,\lambda\Phi z.\,\Phi(\lambda y.\,\mathsf{if}\,y\star z)\mathsf{t}\,)\,(\lambda\varphi x.\,\varphi x)\,\mathsf{t} \quad\longmapsto$$



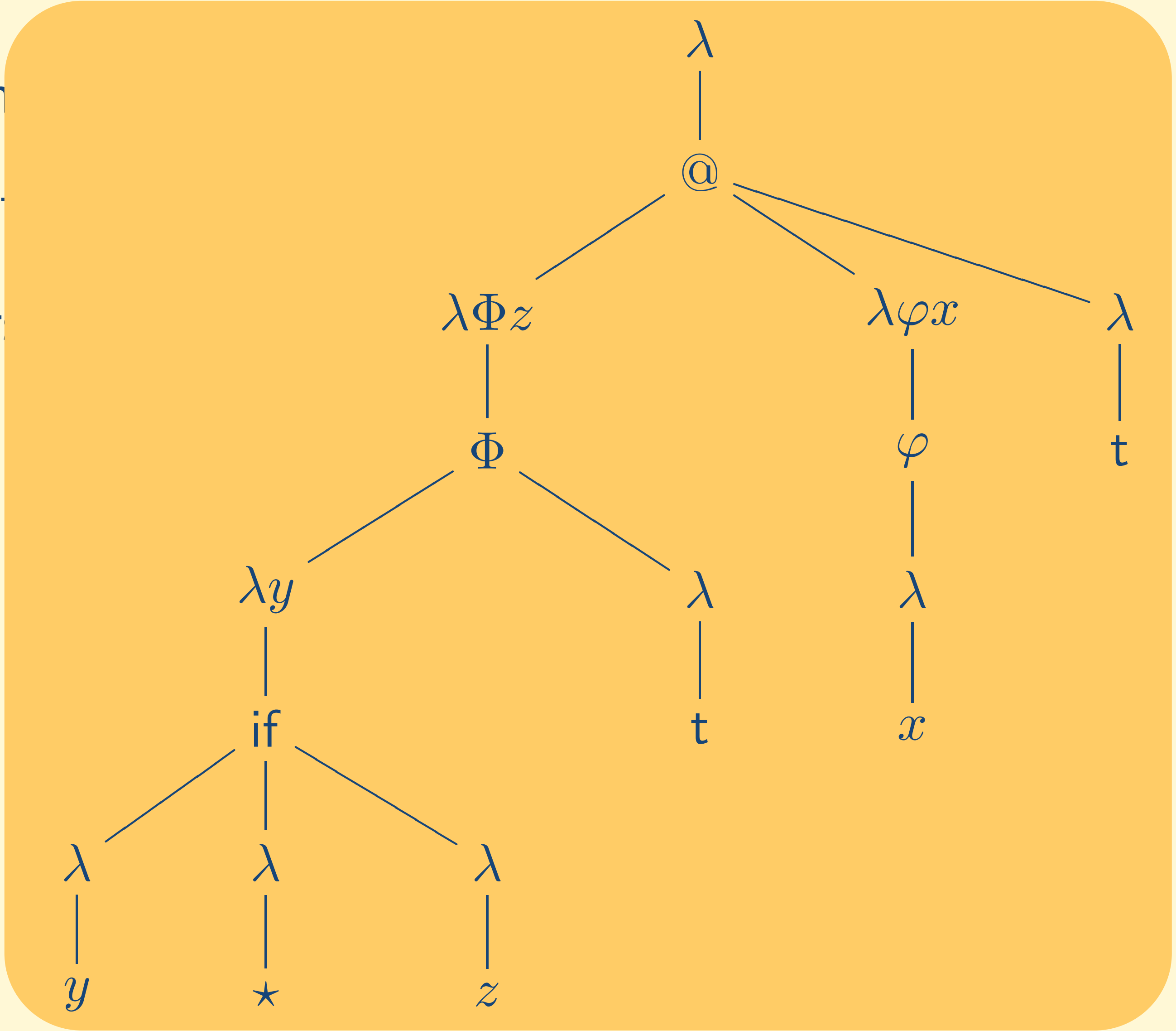
Traversals

A traversal [Blum, Ong] over a computation tree,

- follows the flow of control within it,
- seen from the perspective of *Game Semantics*.

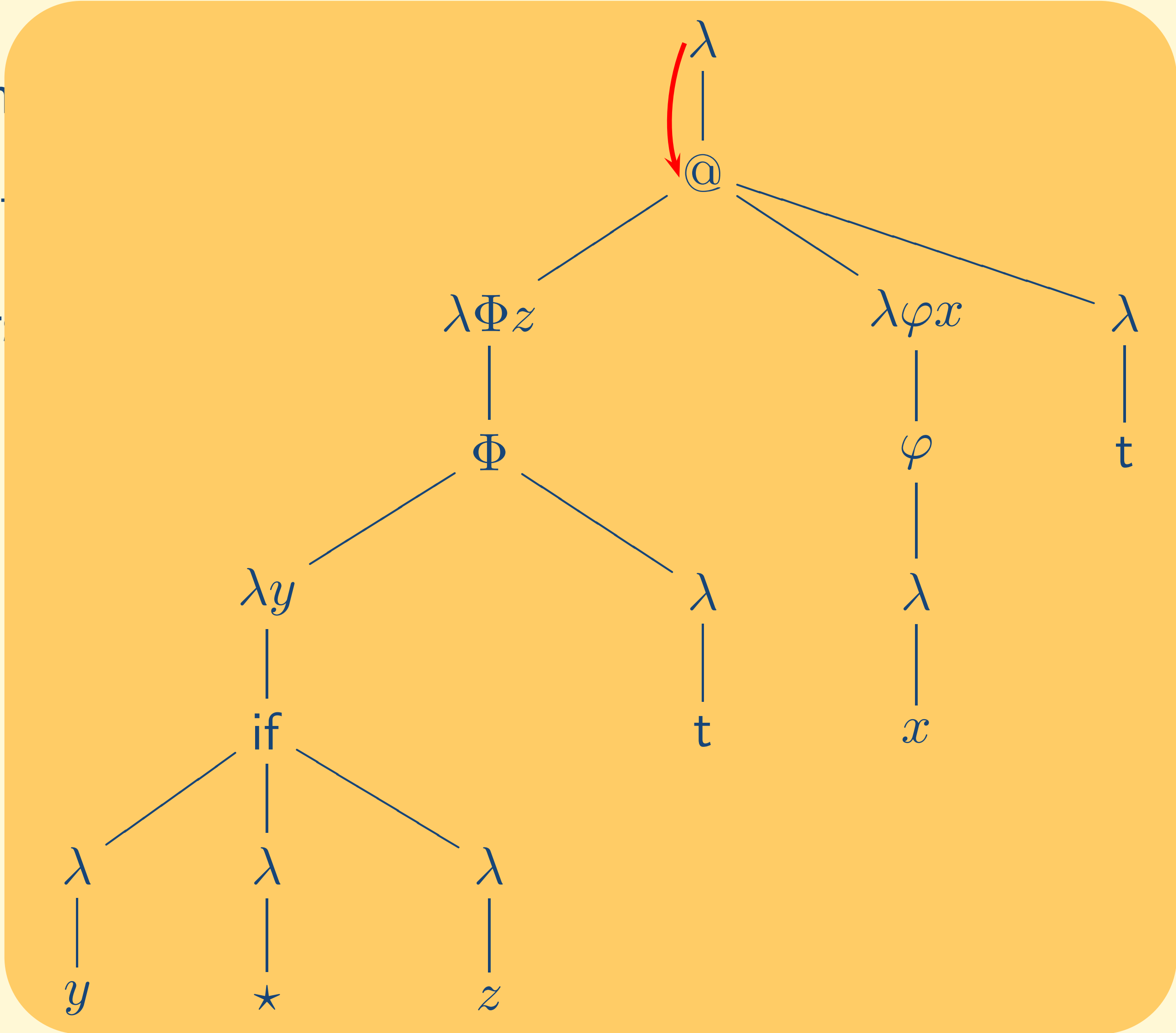
A traversal [Blum, On

- follows the flow o
- seen from the per



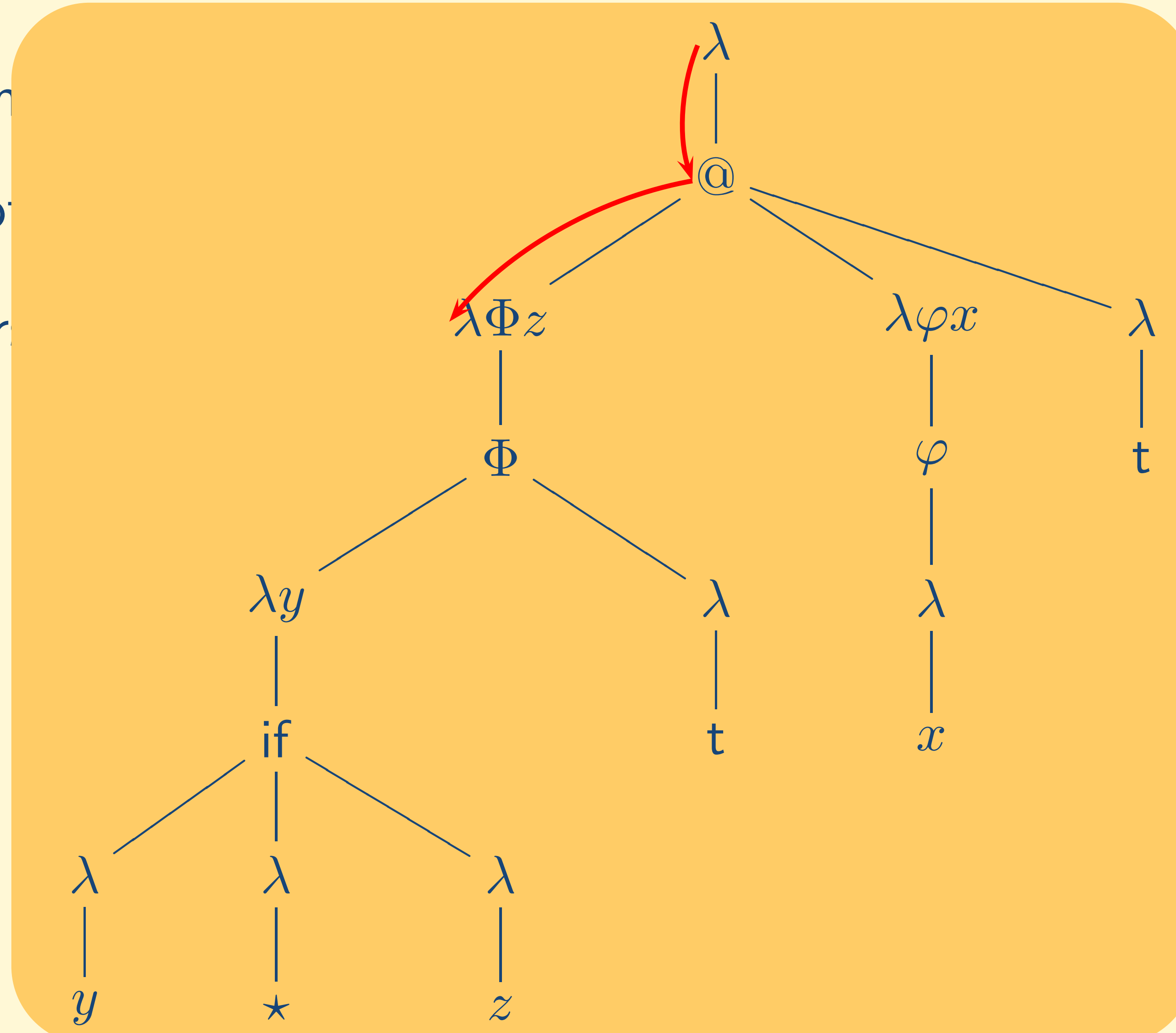
A traversal [Blum, On

- follows the flow o
- seen from the per



A traversal [Blum, On

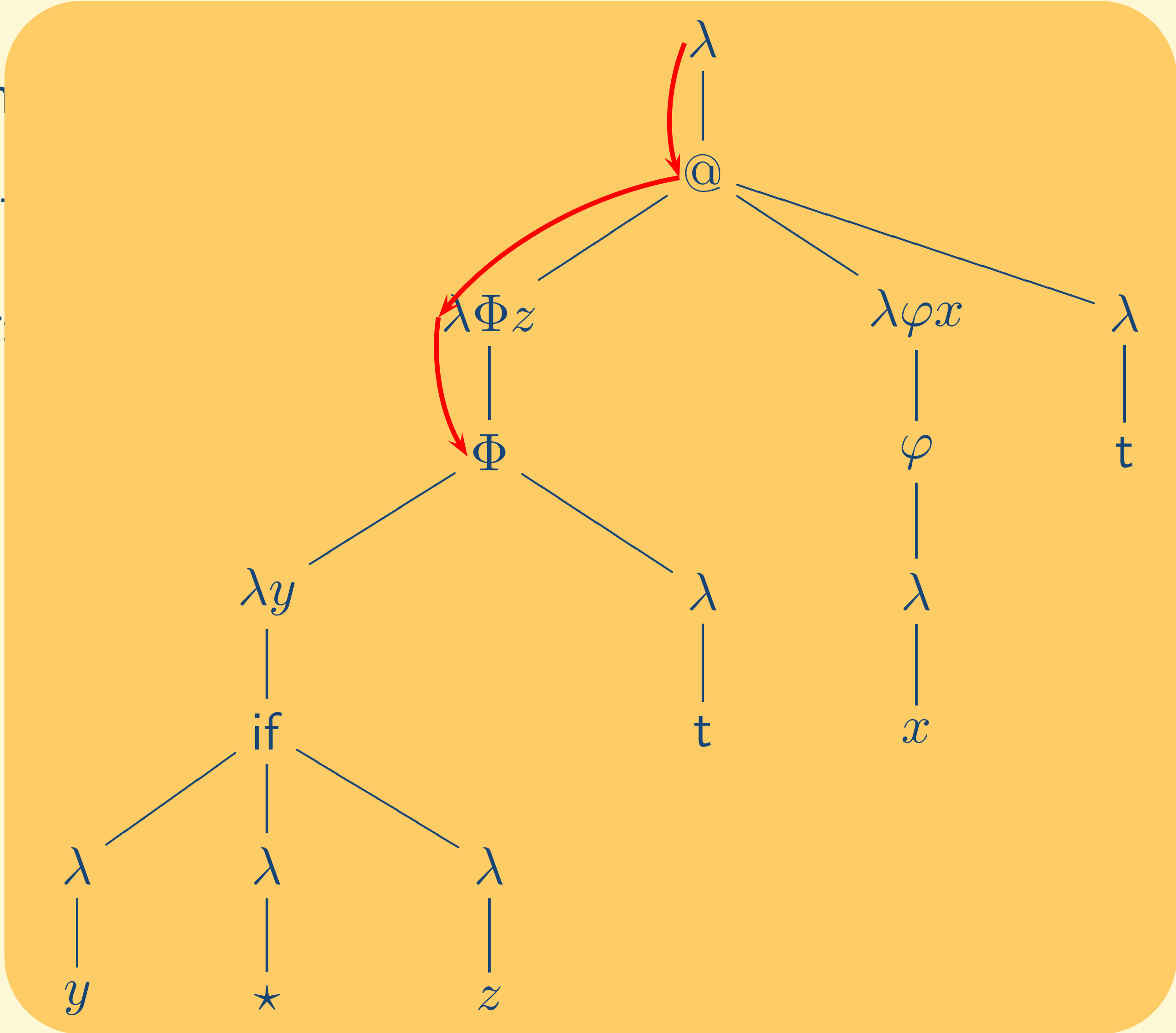
- follows the flow o
- seen from the per



Traversals

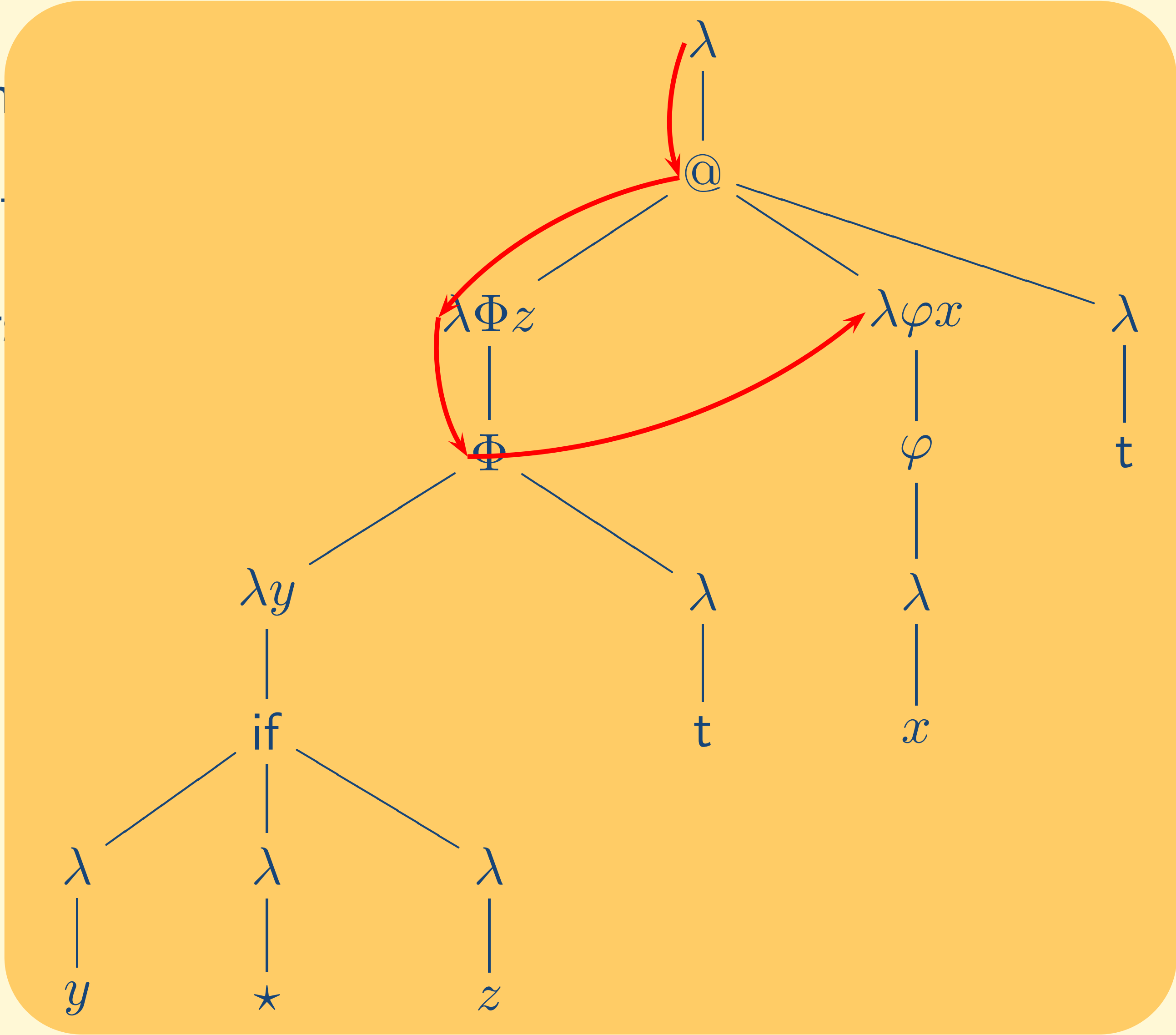
A traversal [Blum, On

- follows the flow o
- seen from the per



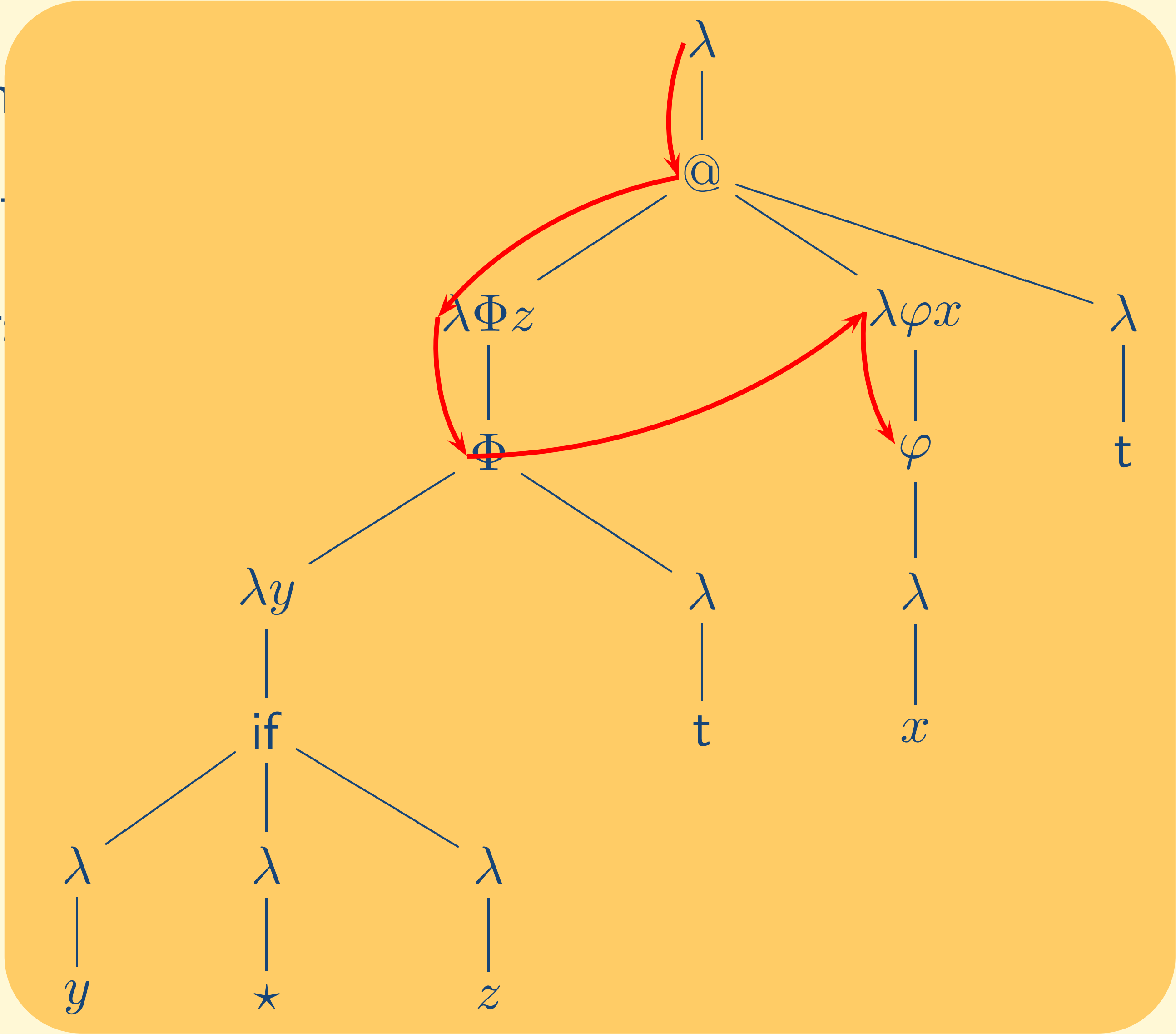
A traversal [Blum, On

- follows the flow o
- seen from the per



A traversal [Blum, On

- follows the flow o
- seen from the per

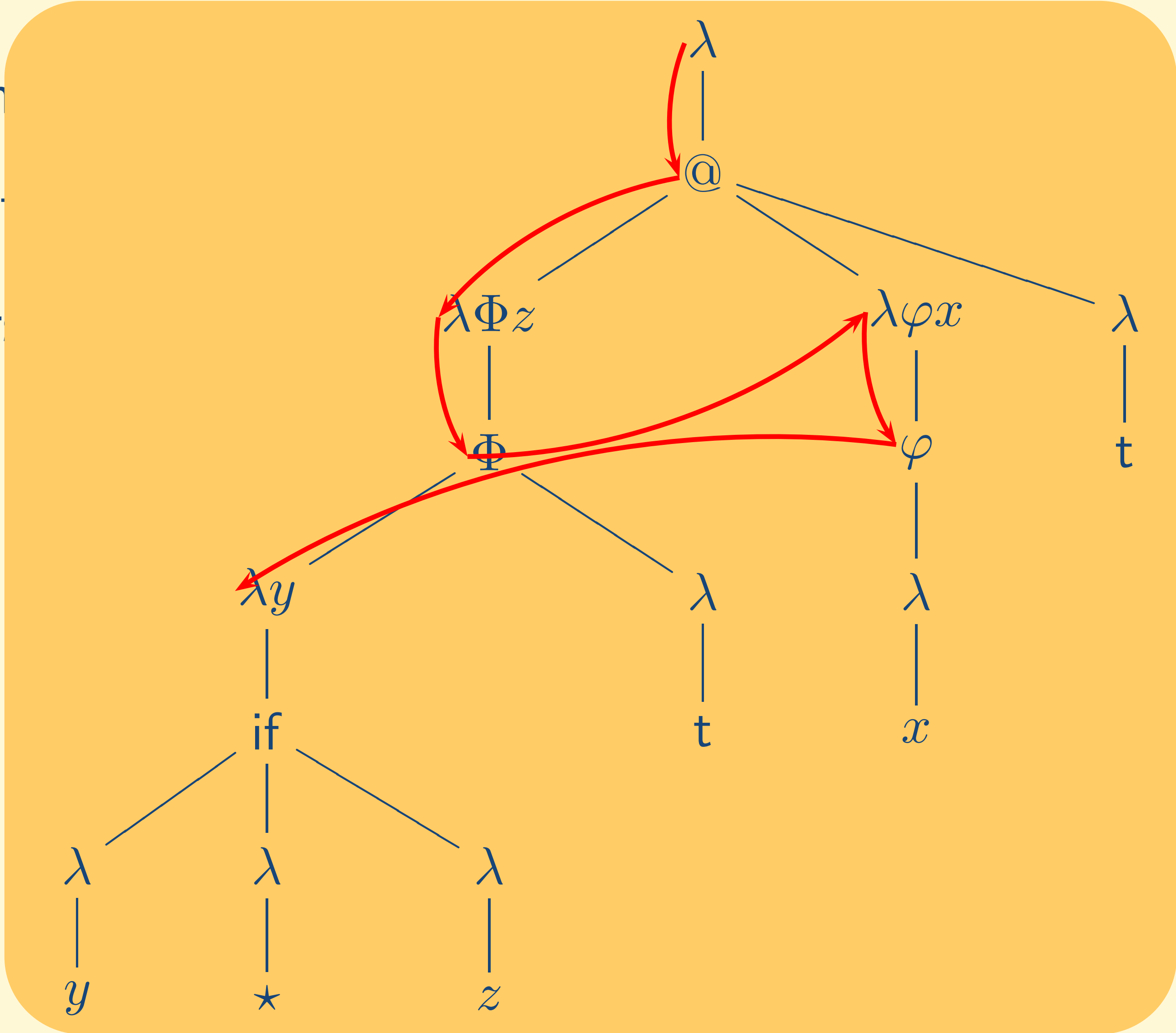


Traversals

A traversal [Blum, On

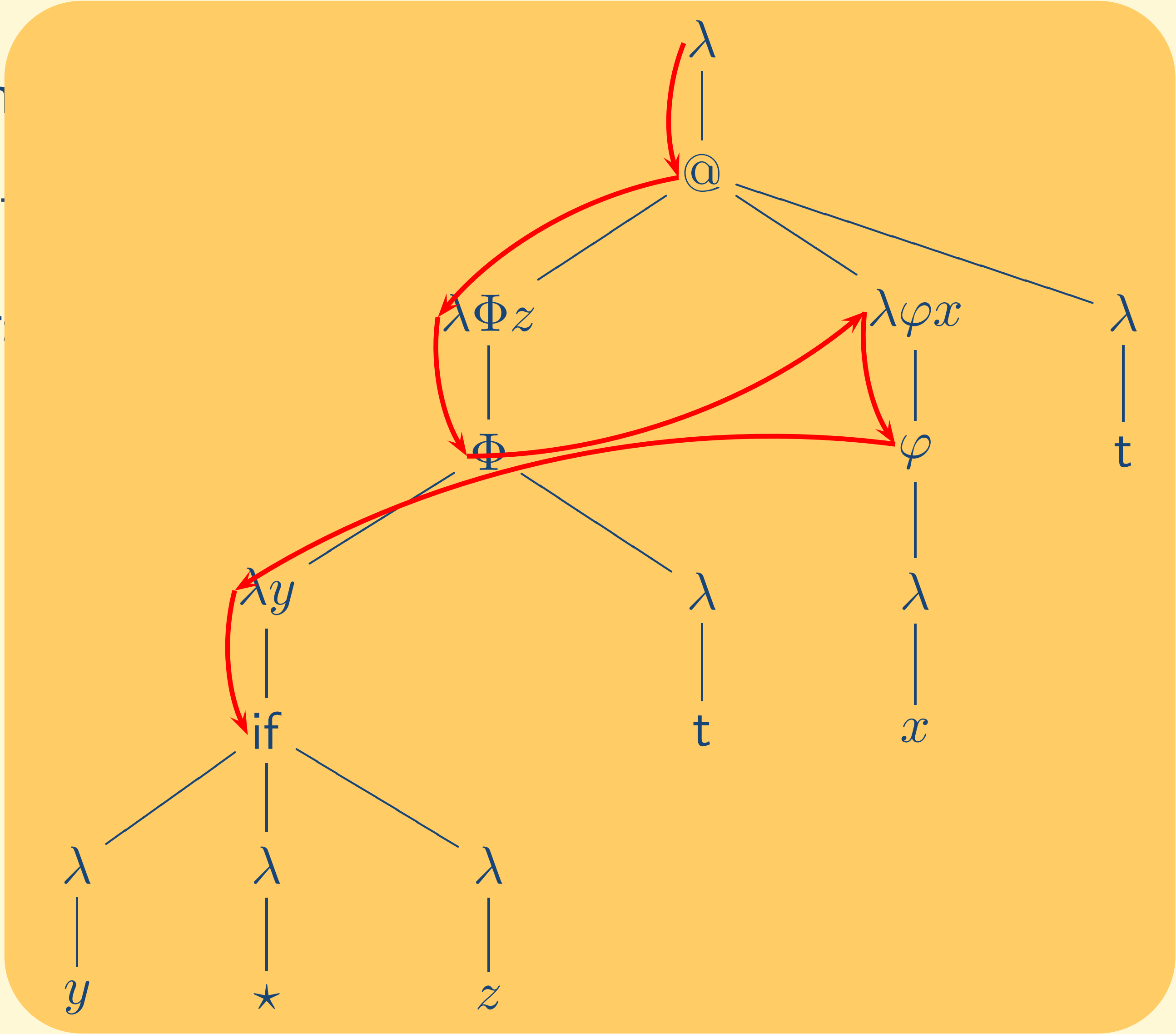
- follows the flow o

- seen from the per



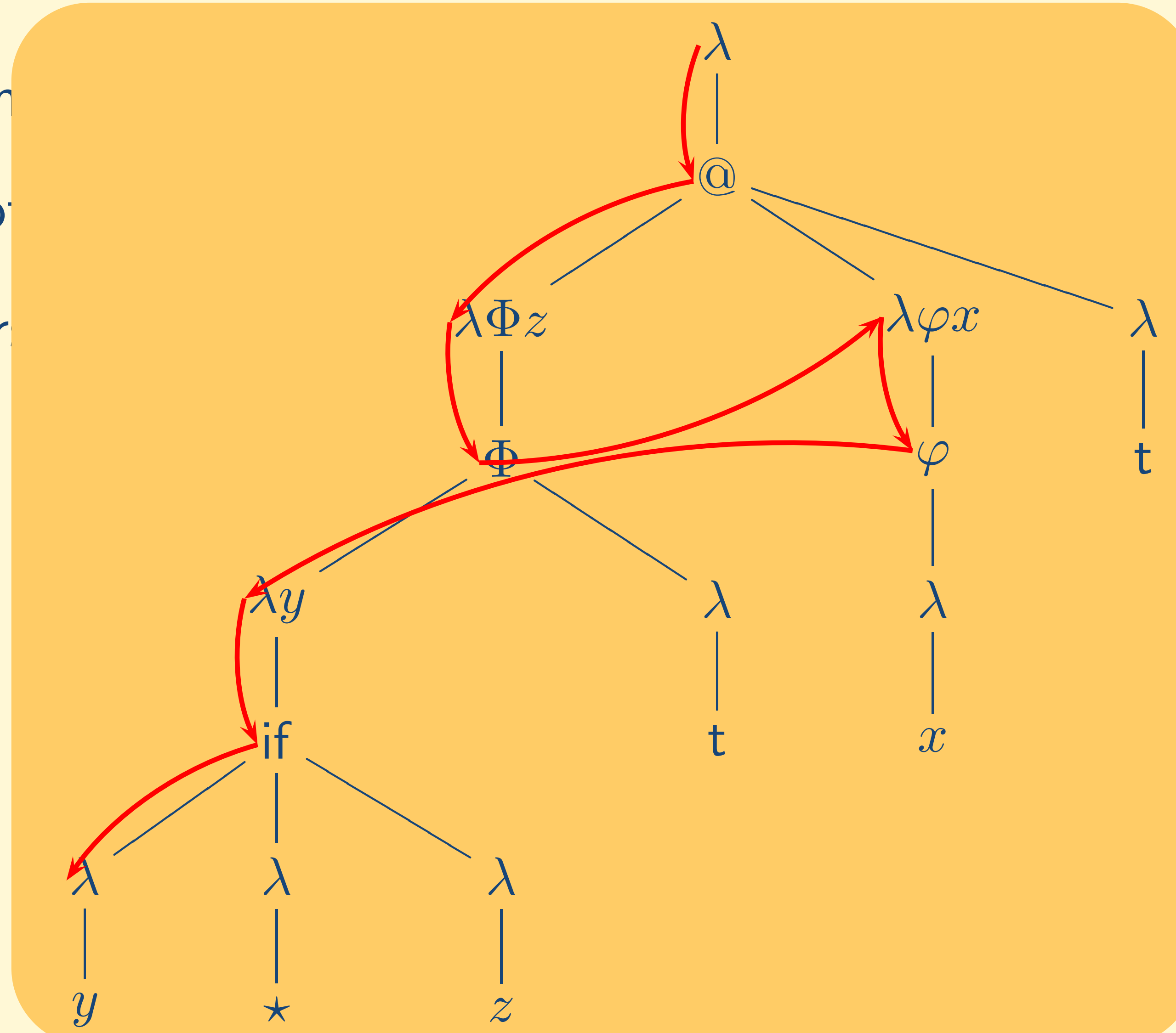
A traversal [Blum, On

- follows the flow o
- seen from the per



A traversal [Blum, On

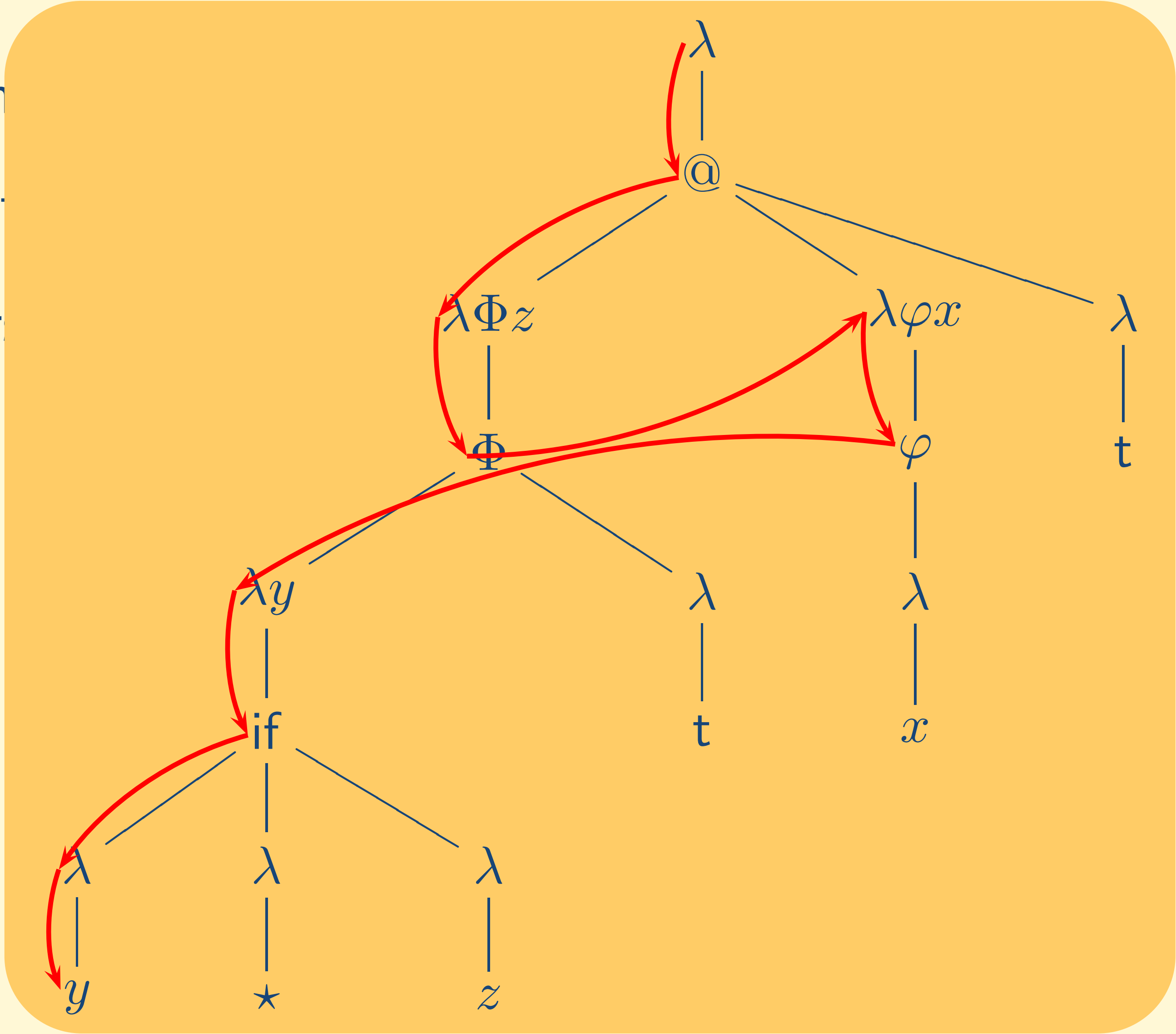
- follows the flow o
- seen from the per



A traversal [Blum, On

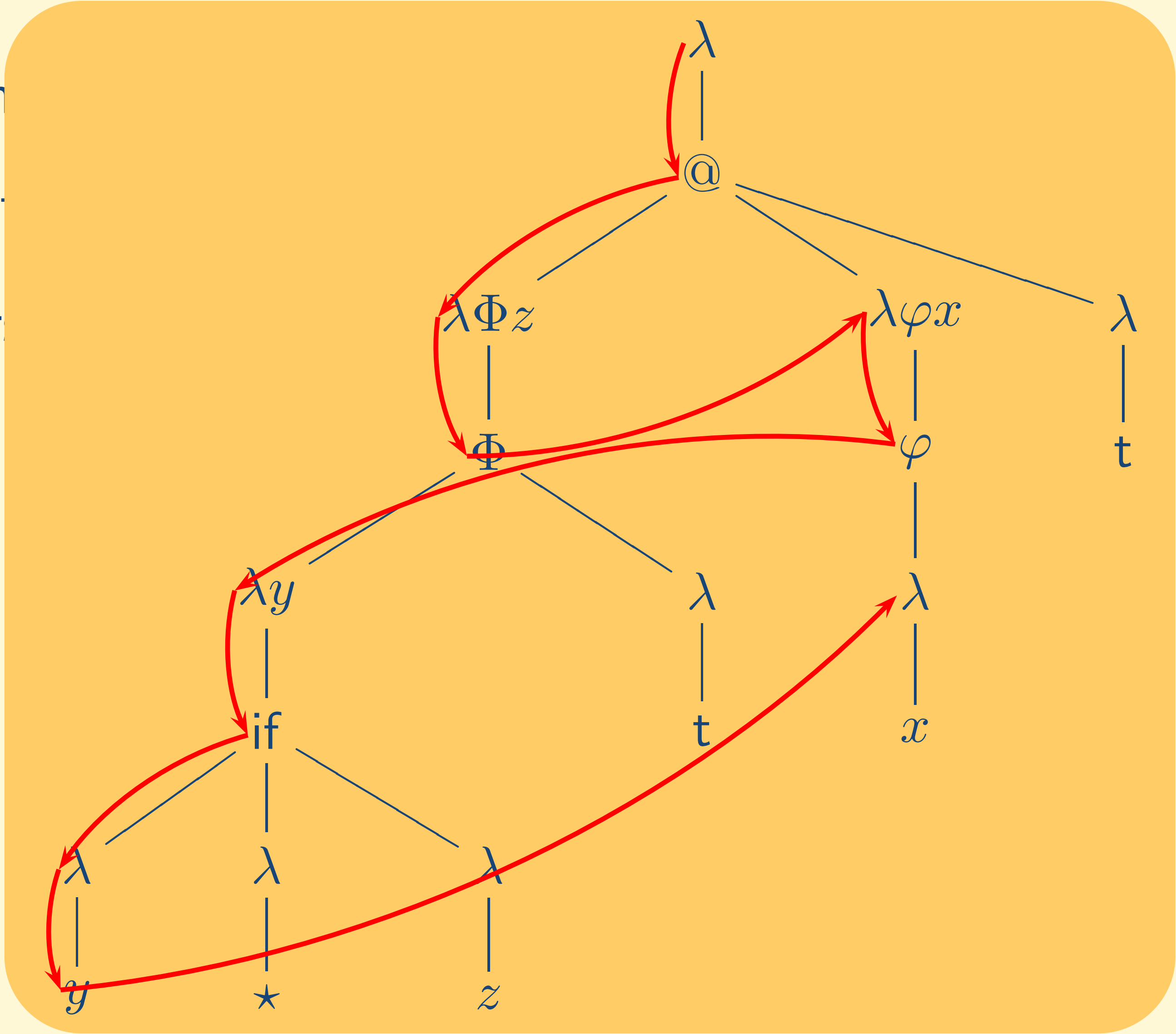
- follows the flow o

- seen from the per



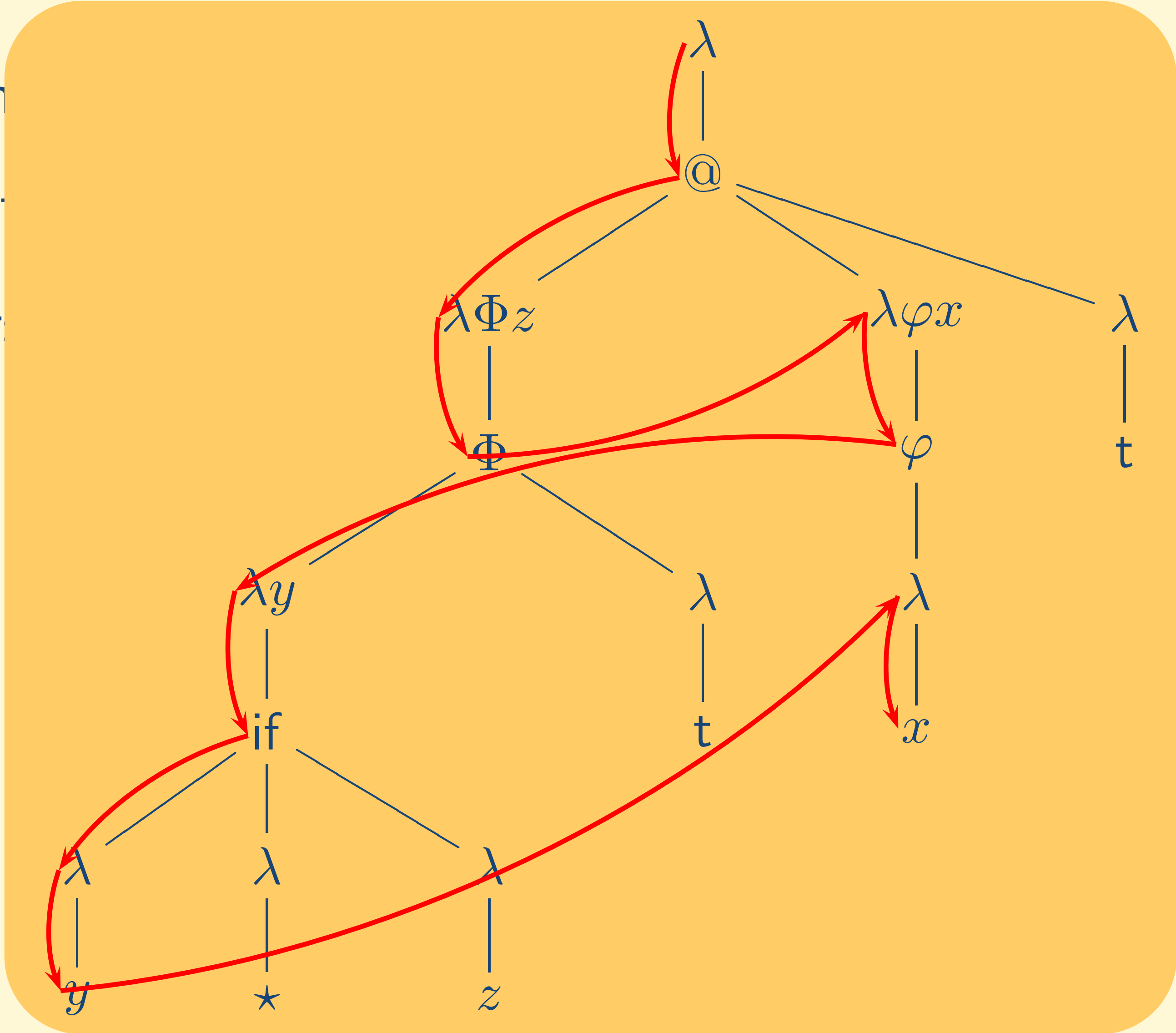
A traversal [Blum, On

- follows the flow o
- seen from the per



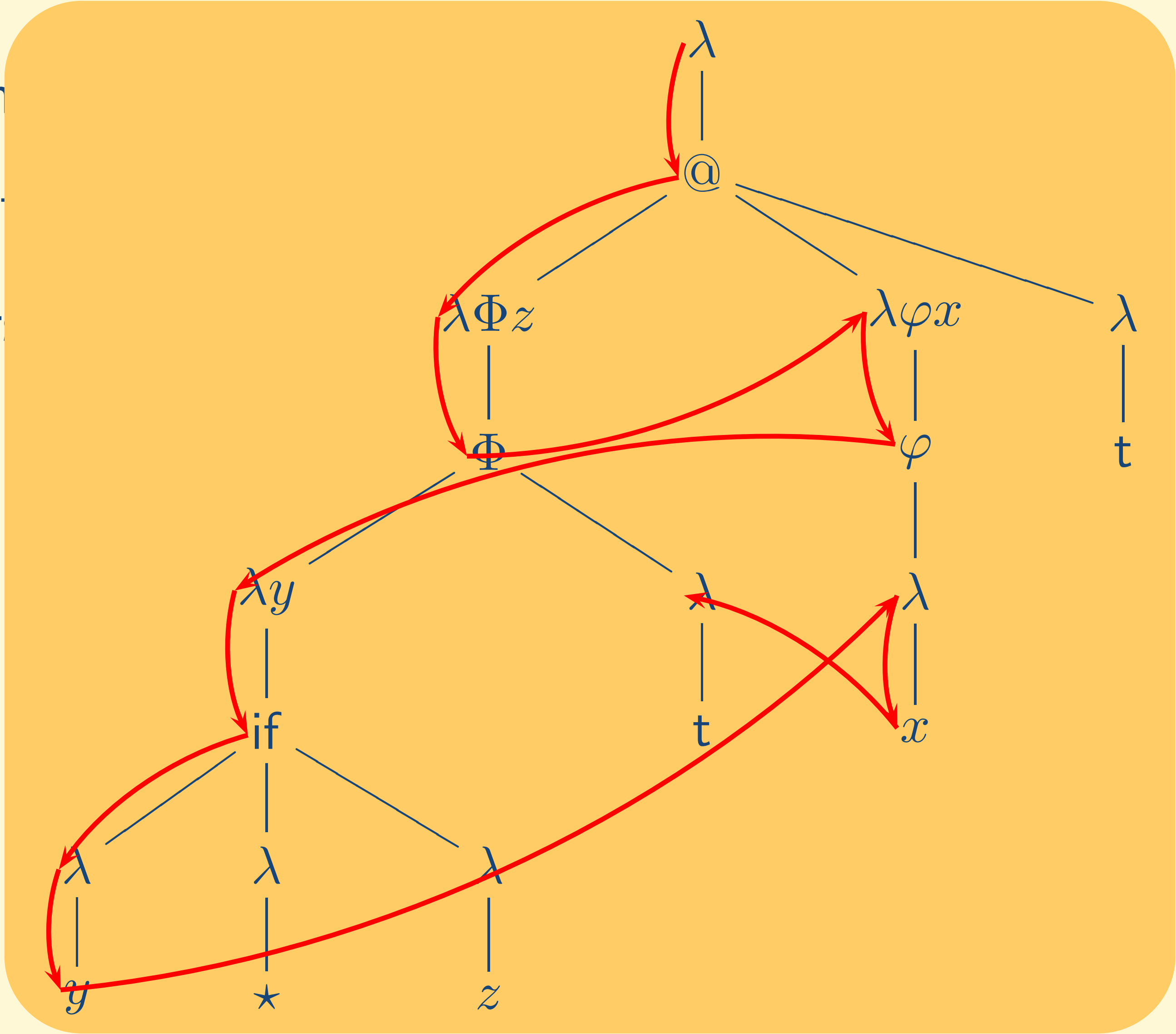
A traversal [Blum, On

- follows the flow o
- seen from the per



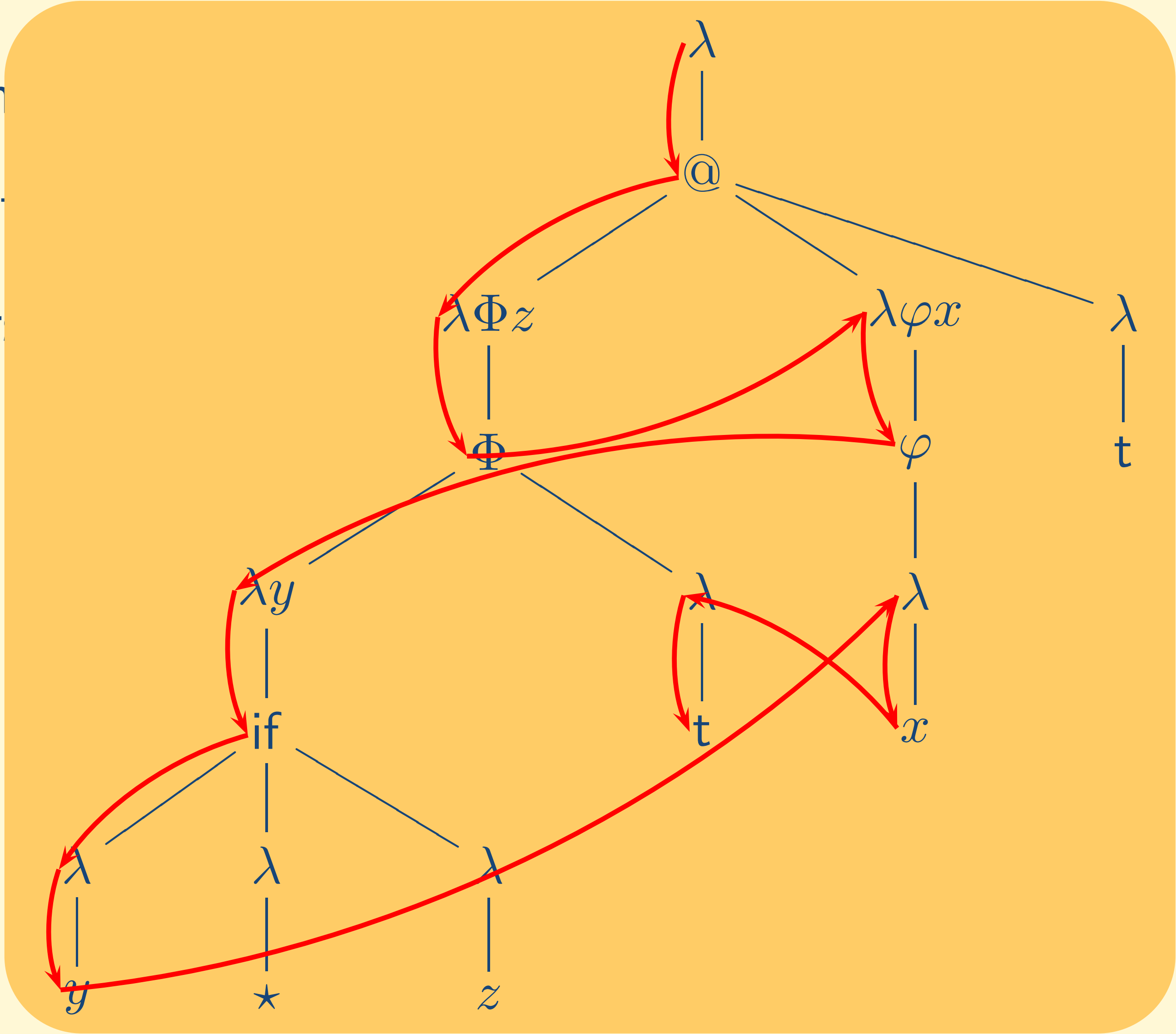
A traversal [Blum, On

- follows the flow o
- seen from the pers



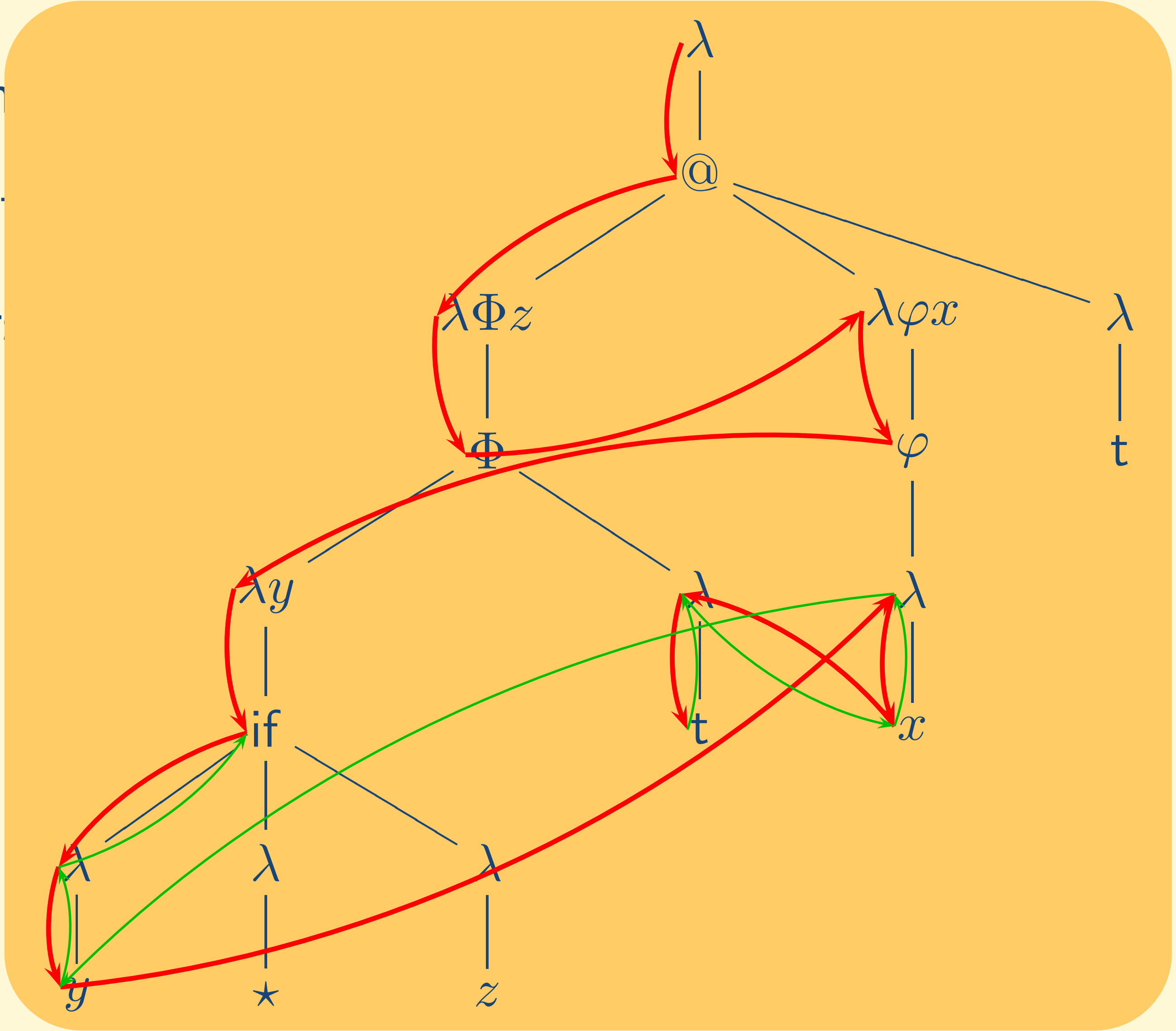
A traversal [Blum, On

- follows the flow o
- seen from the per



A traversal [Blum, On

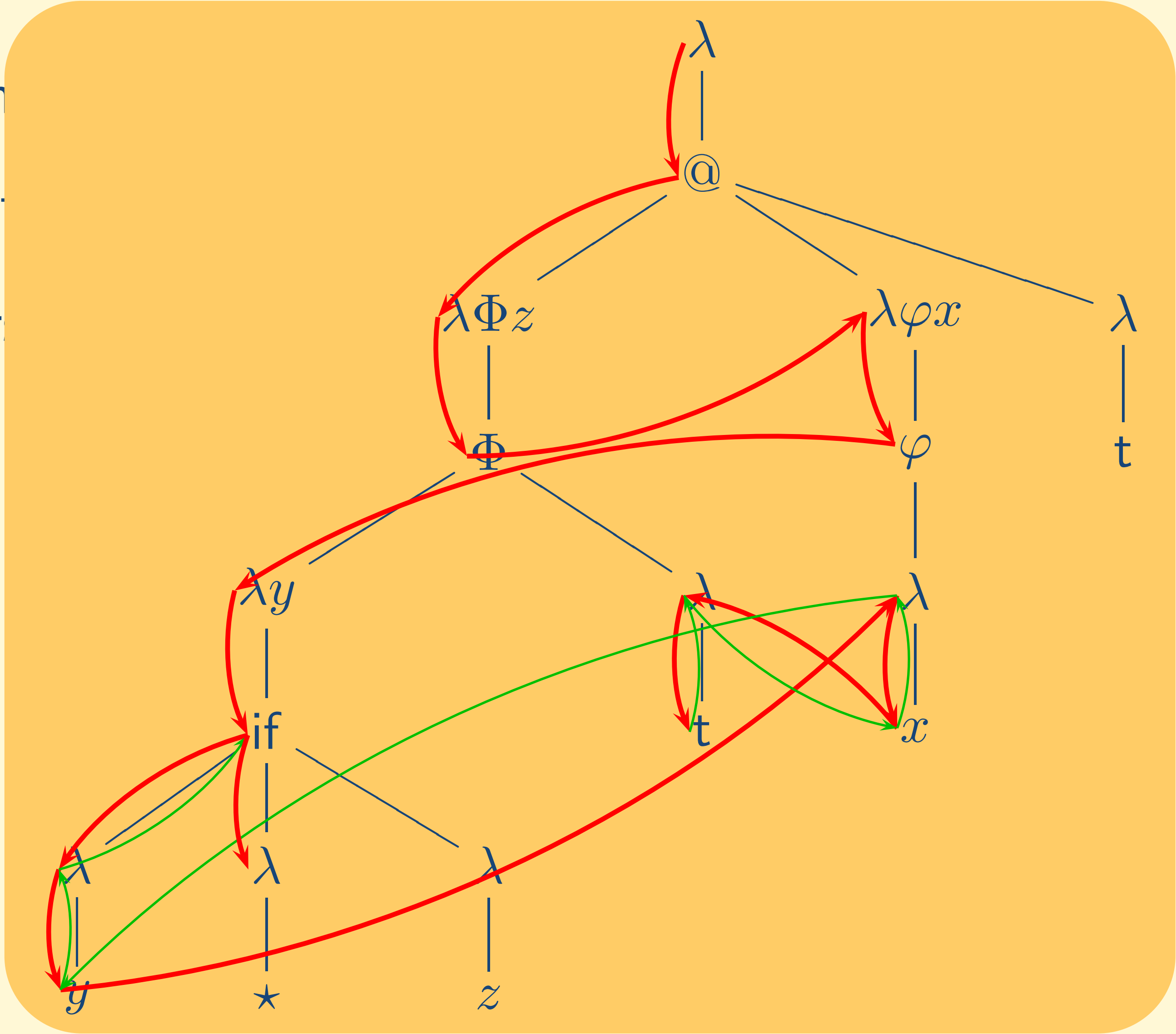
- follows the flow o
- seen from the per



A traversal [Blum, On

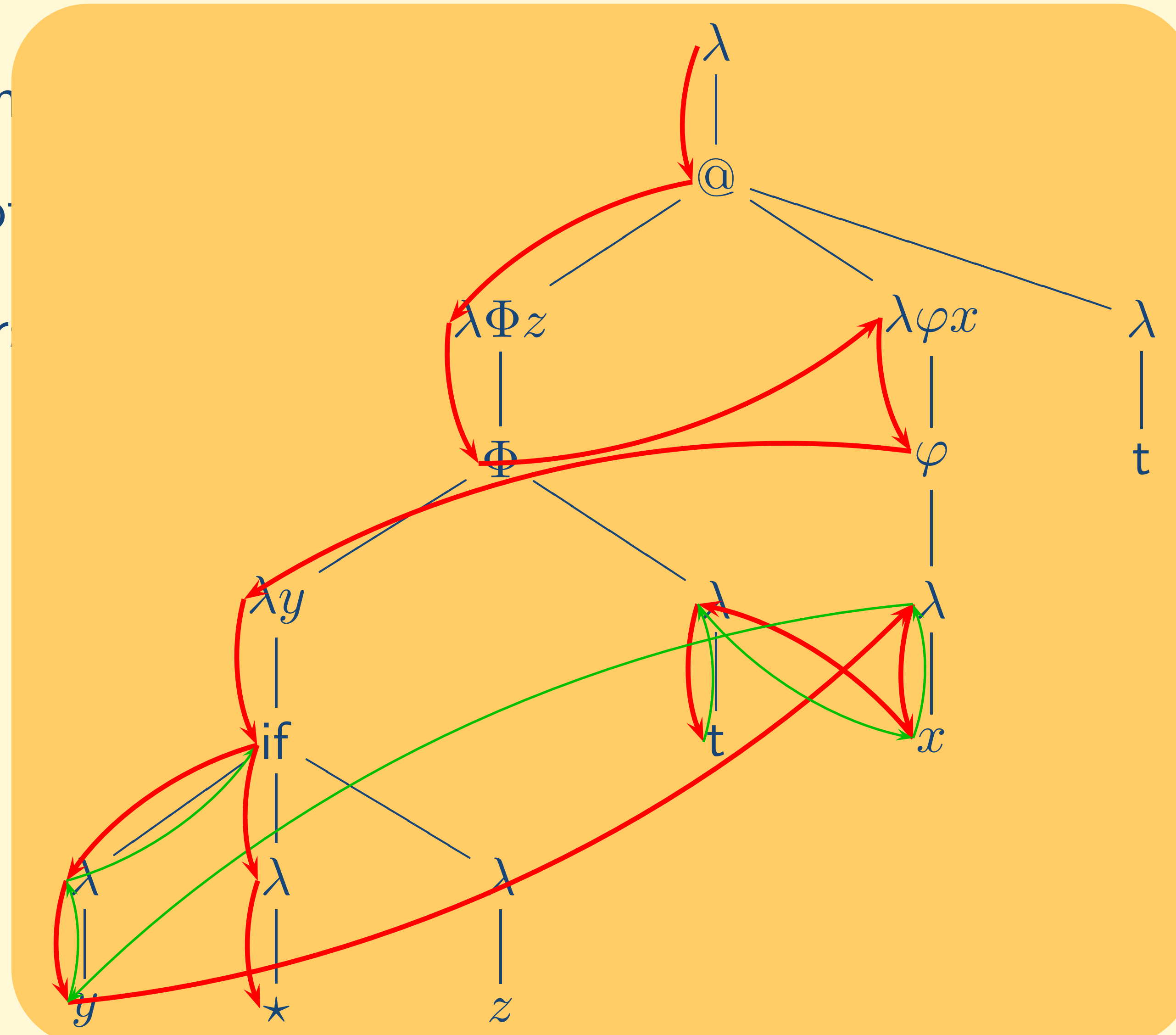
- follows the flow o

- seen from the per



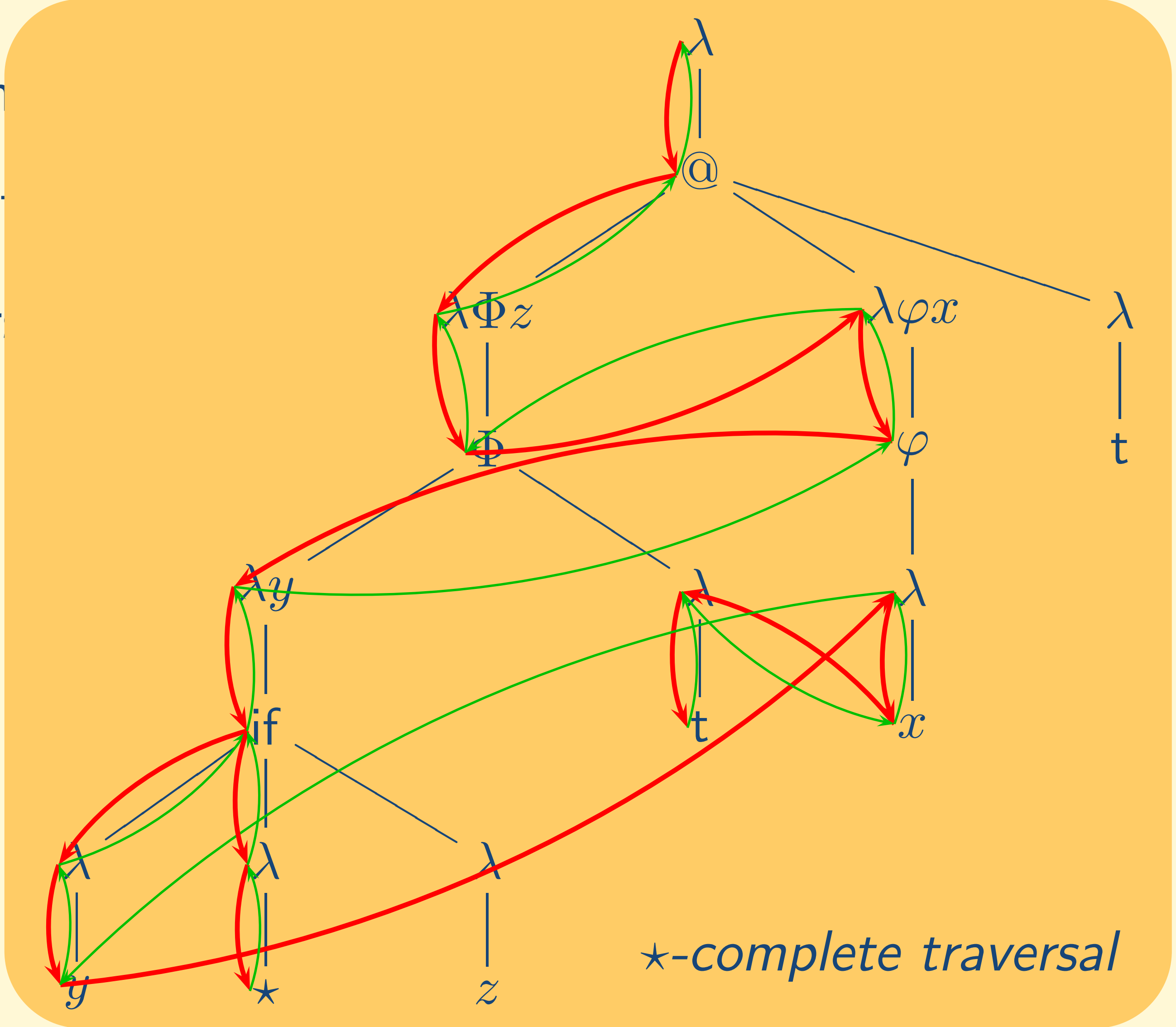
A traversal [Blum, On

- follows the flow o
- seen from the per



A traversal [Blum, On

- follows the flow o
- seen from the per



Traversals

A traversal [Blum, Ong] over a computation tree,

- follows the flow of control within it,
- seen from the perspective of *Game Semantics*.

A traversal is v-*complete* if every *question* (red visit) has been *answered* (green visit), and the root question has been answered with v.

Theorem: For any $P : o$ and value v, $P \twoheadrightarrow v$ iff there is a complete v-traversal over $\lambda(P)$.

Alternating Tree Automata

An ATA is a quadruple $\mathcal{A} = \langle Q, \Sigma, q_0, \Delta \rangle$ where:

- Q is a finite set of states,
- Σ is a finite ranked alphabet,
- $q_0 \in Q$ is the initial state,
- Δ is a finite transition relation: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.

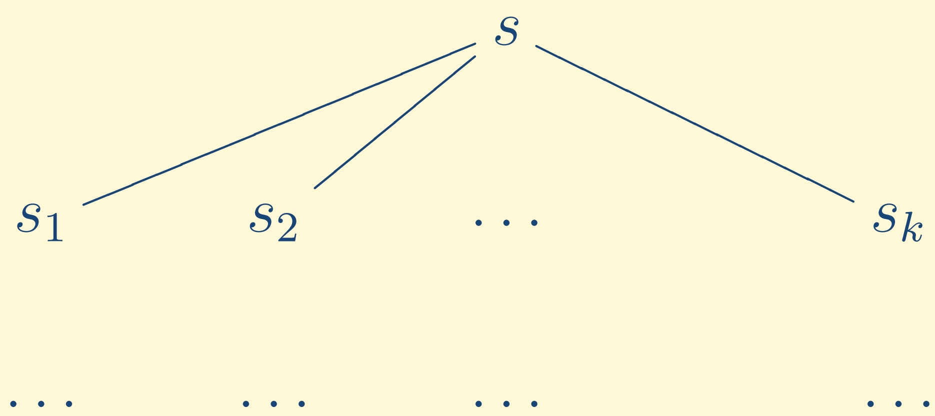
$s \in \Sigma$
$q \in Q$
$Q_1, \ldots, Q_k \subseteq Q$

Alternating Tree Automata

An ATA is a quadruple $\mathcal{A} = \langle Q, \Sigma, q_0, \Delta \rangle$ where:

- Q is a finite set of states,
- Σ is a finite ranked alphabet,
- $q_0 \in Q$ is the initial state,
- Δ is a finite transition relation: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.

$s \in \Sigma$
$q \in Q$
$Q_1, \ldots, Q_k \subseteq Q$

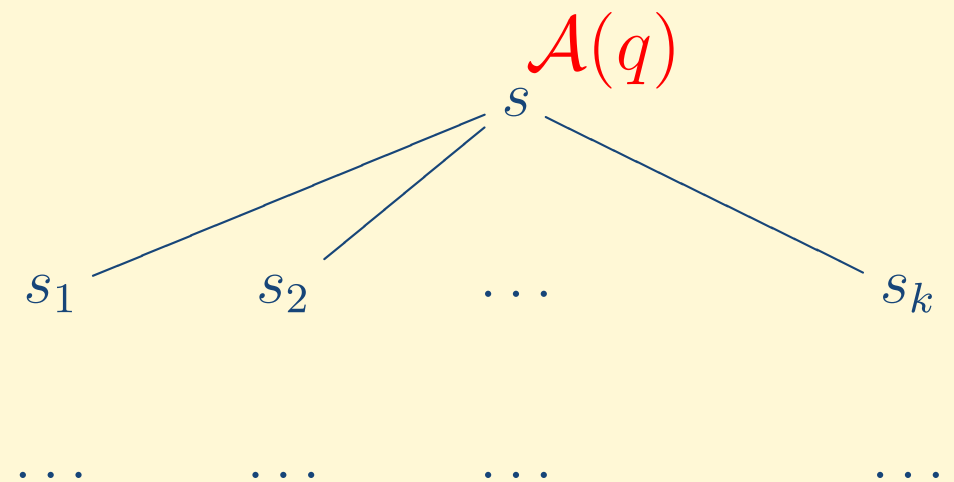


Alternating Tree Automata

An ATA is a quadruple $\mathcal{A} = \langle Q, \Sigma, q_0, \Delta \rangle$ where:

- Q is a finite set of states,
- Σ is a finite ranked alphabet,
- $q_0 \in Q$ is the initial state,
- Δ is a finite transition relation: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.

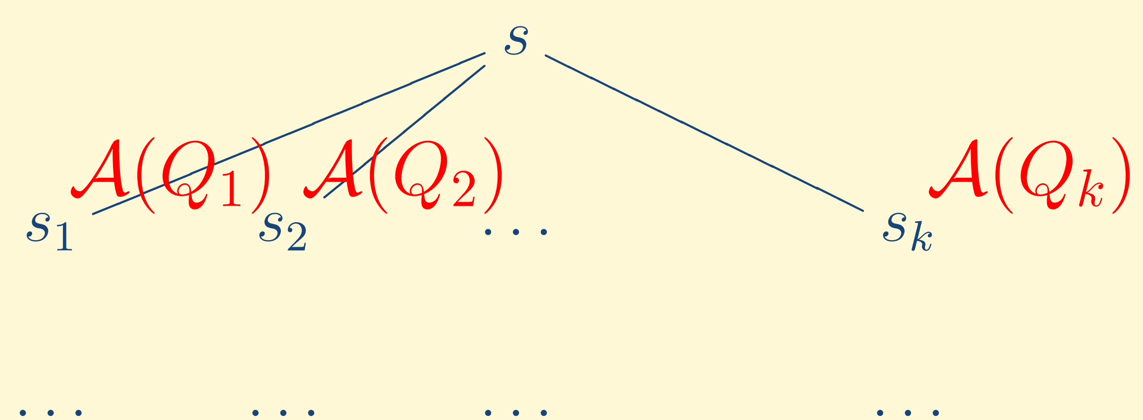
$s \in \Sigma$
$q \in Q$
$Q_1, \ldots, Q_k \subseteq Q$



An ATA is a quadruple $\mathcal{A} = \langle Q, \Sigma, q_0, \Delta \rangle$ where:

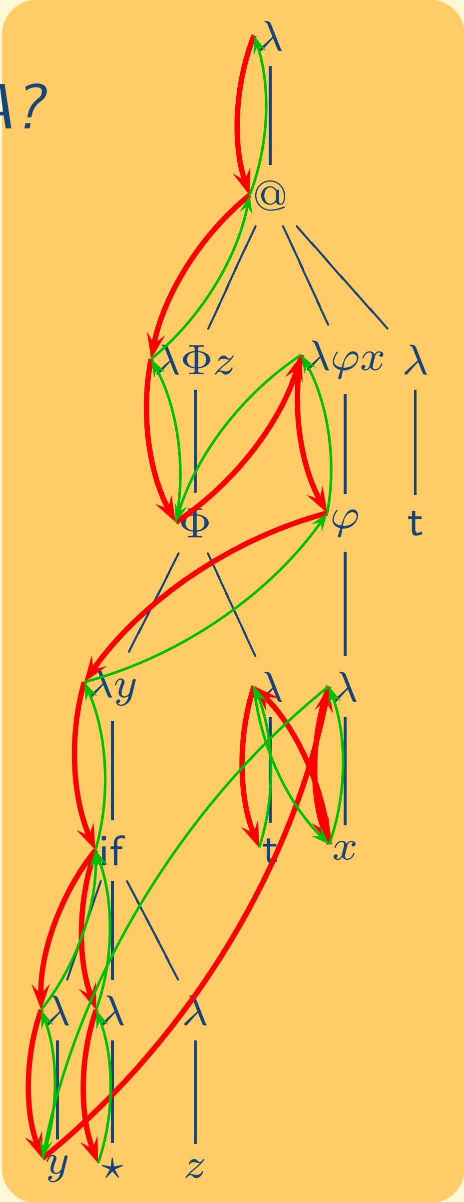
- Q is a finite set of states,
- Σ is a finite ranked alphabet,
- $q_0 \in Q$ is the initial state,
- Δ is a finite transition relation: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.

$s \in \Sigma$
$q \in Q$
$Q_1, \ldots, Q_k \subseteq Q$



Traversal-simulating ATA's

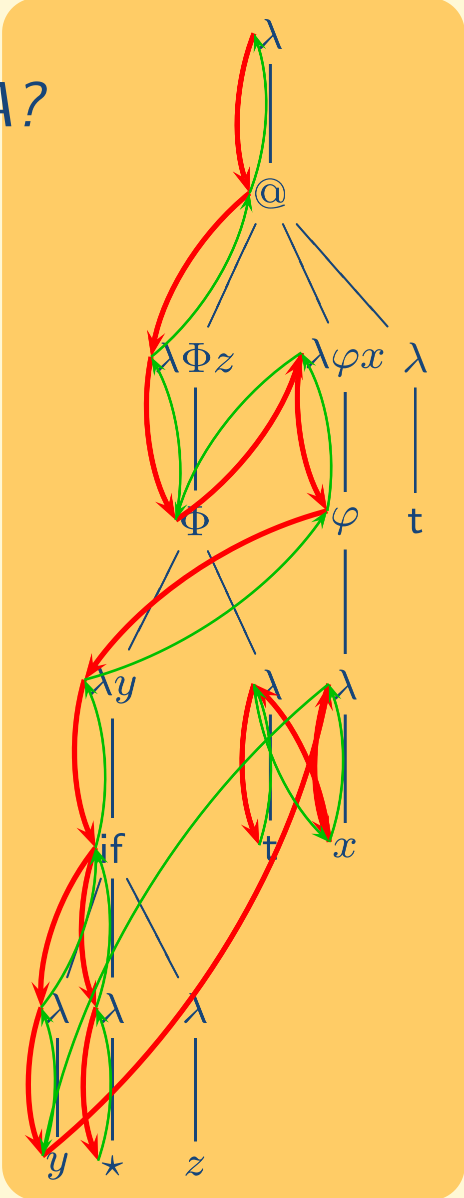
How can we simulate a complete traversal by an ATA?



Traversal-simulating ATA's

How can we simulate a complete traversal by an ATA?

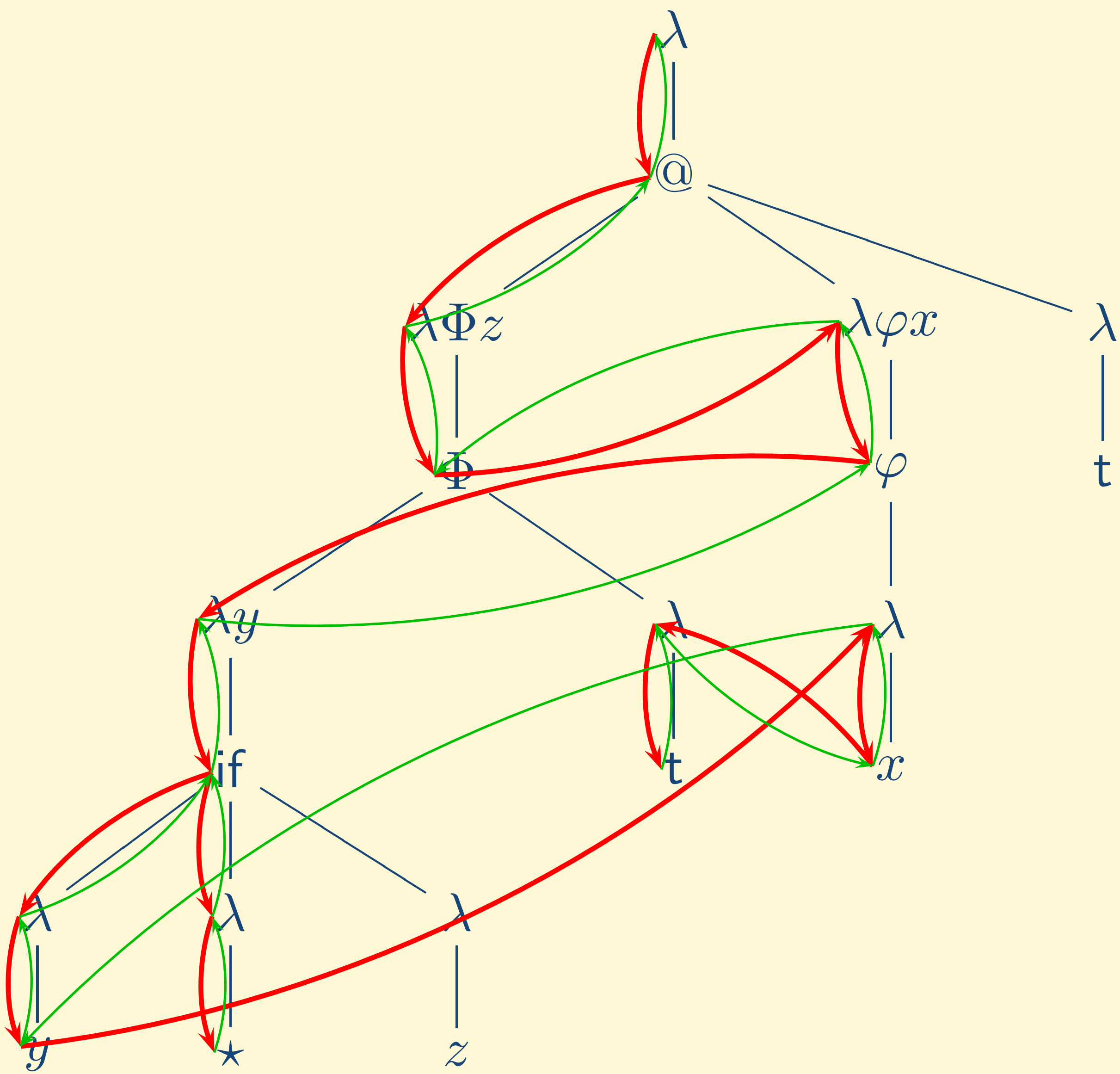
- By *guessing* the number of visits of each node.
- By *guessing* the *profile* of each variable per visit.
- By verifying these guesses.



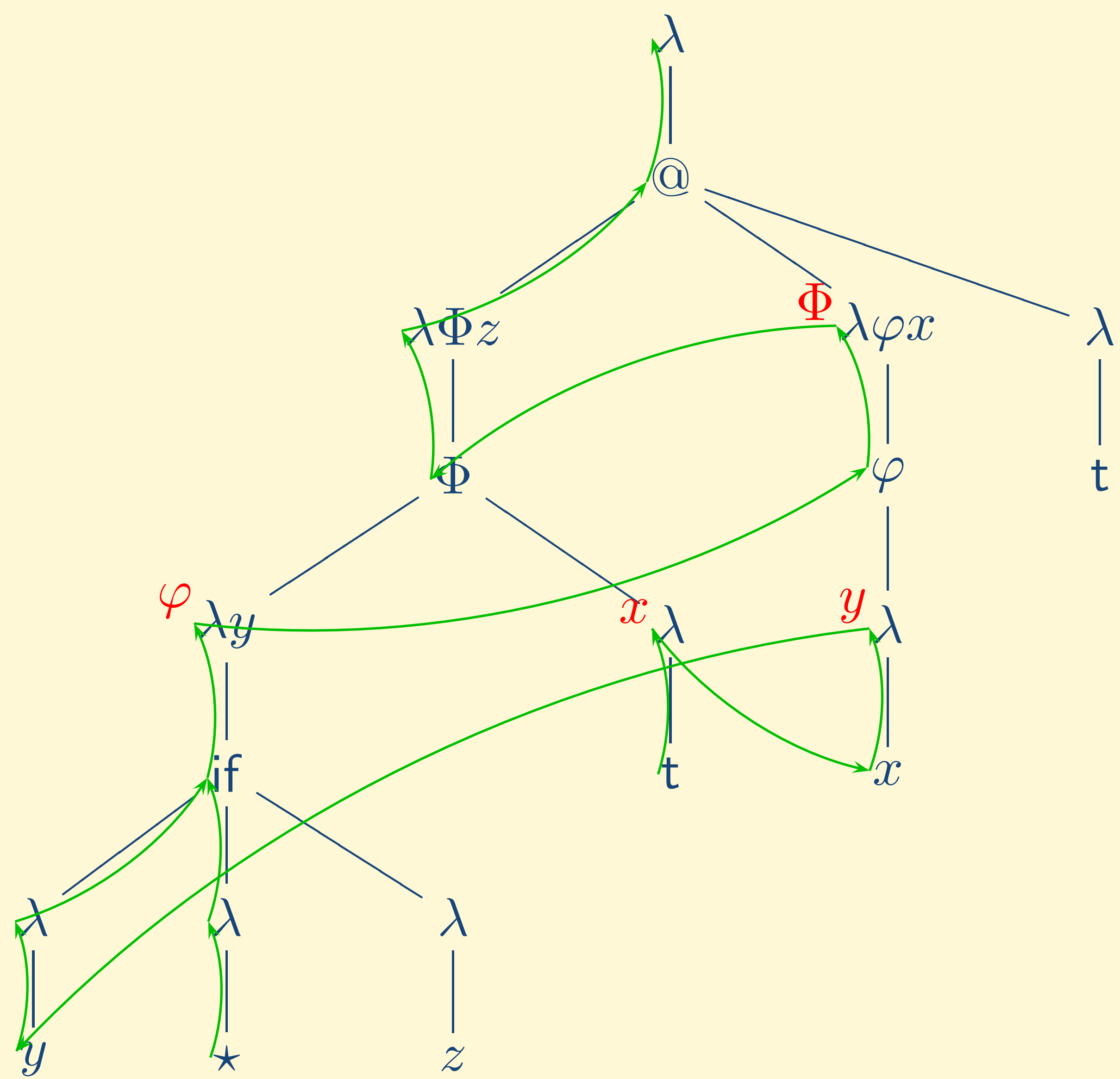
Variable profiles

- Introduced by [Ong'06].
- $\mathbf{VP}(A_1,\dots,A_n,o) := \mathit{Var} \times \mathit{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \mathbf{VP}(A_i))$
- Notation: (x,v), $(x,v \mid \pi_1,\dots,\pi_n)$

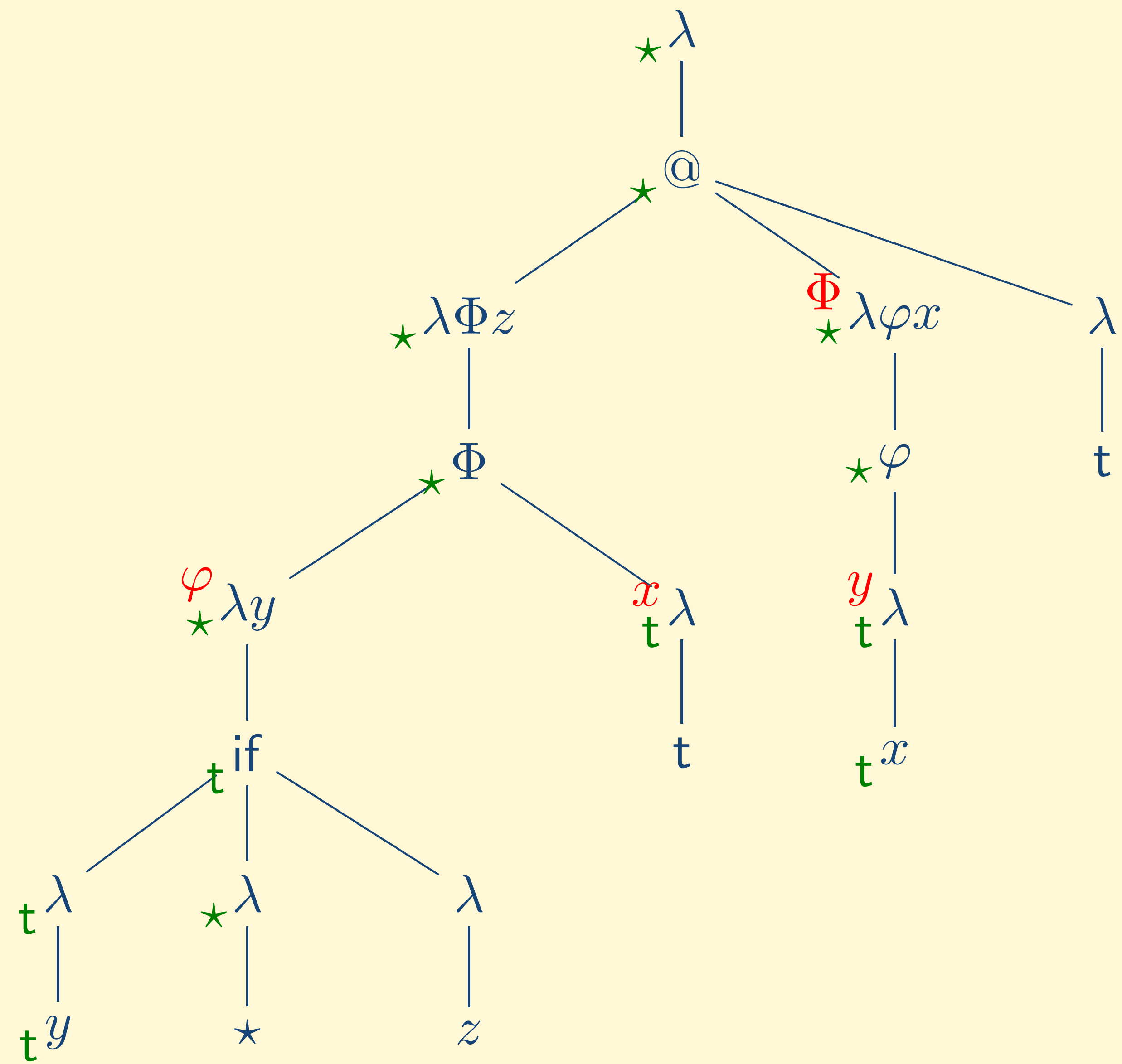
Variable profiles



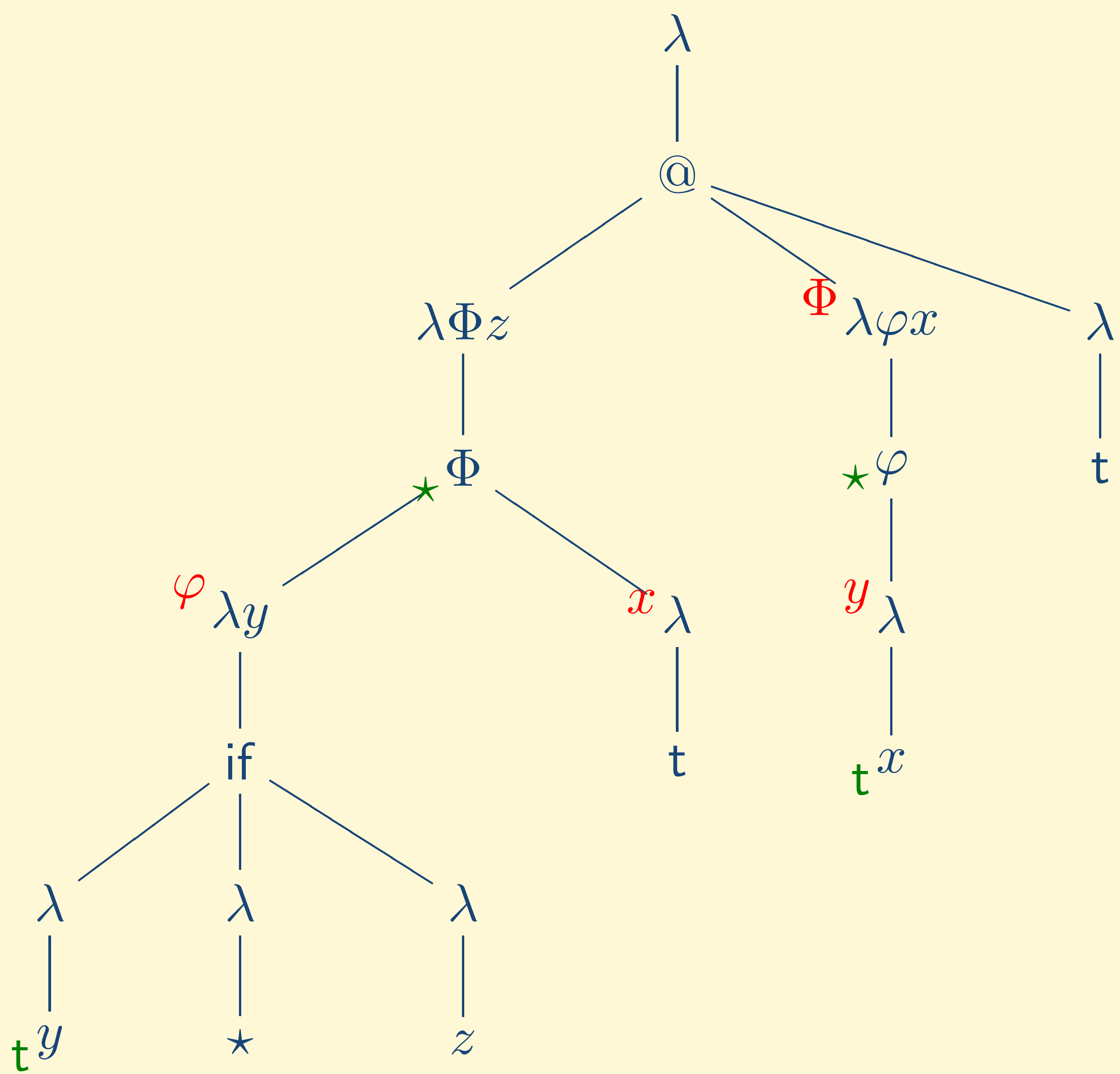
Variable profiles



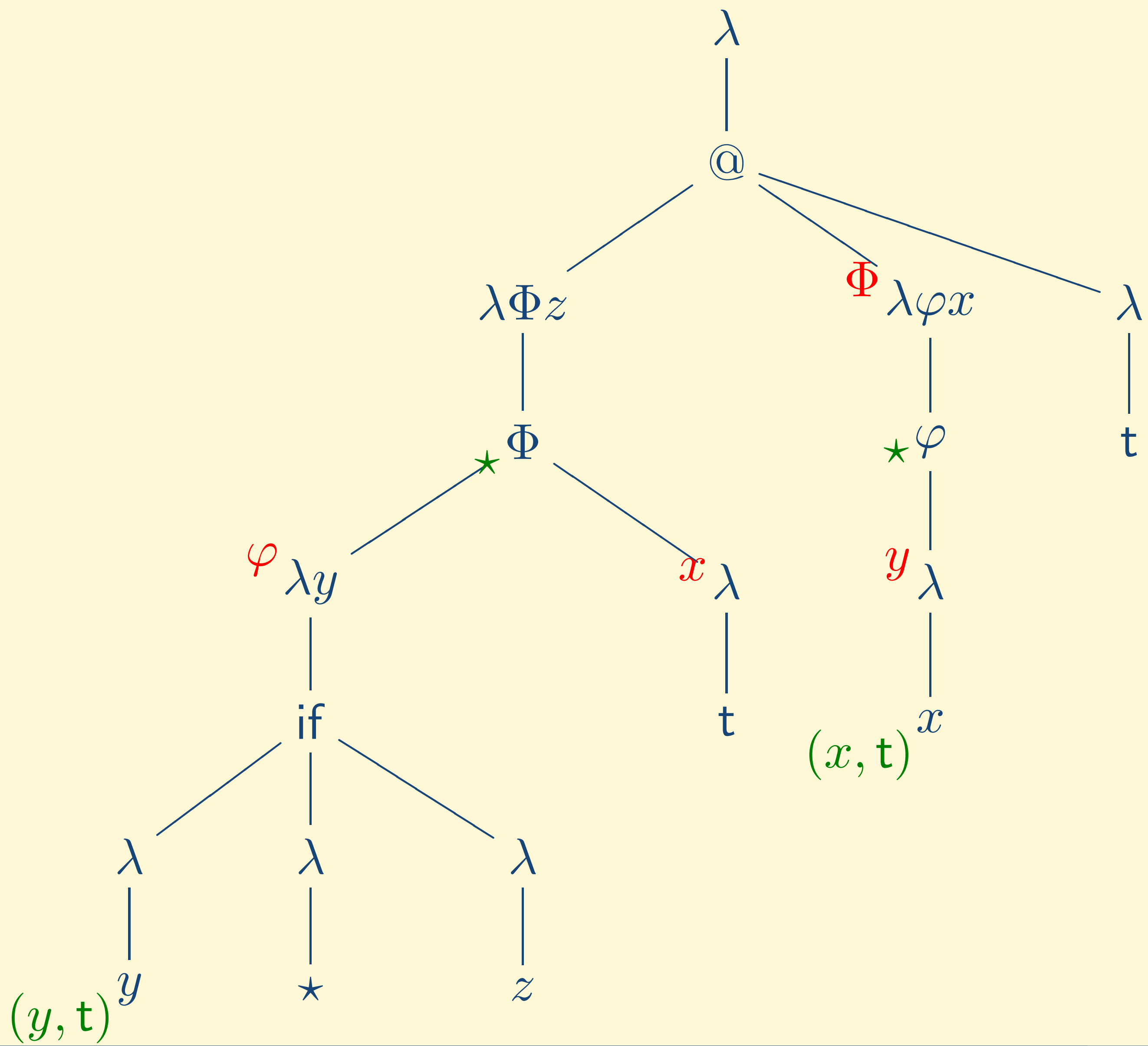
Variable profiles



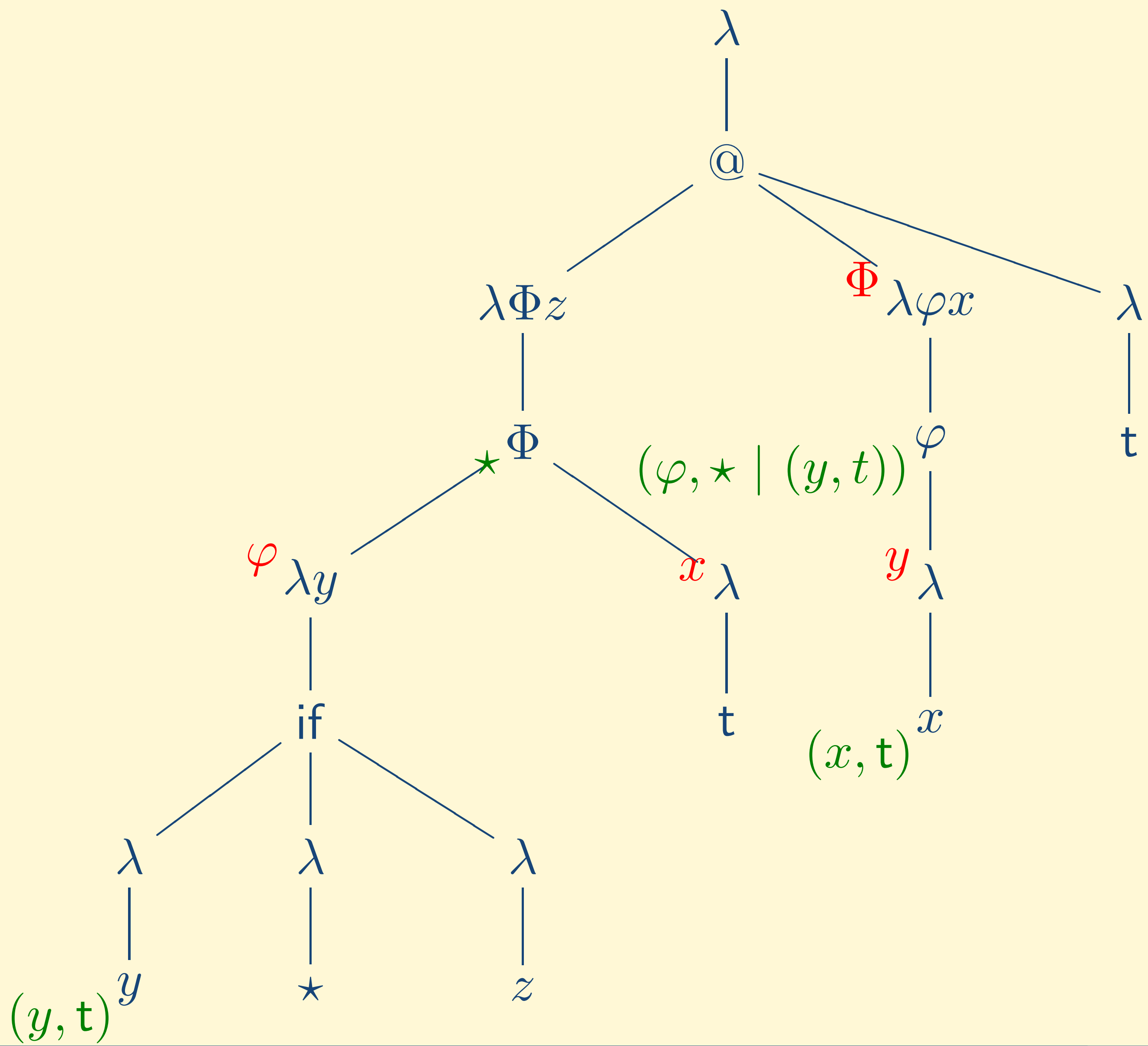
Variable profiles



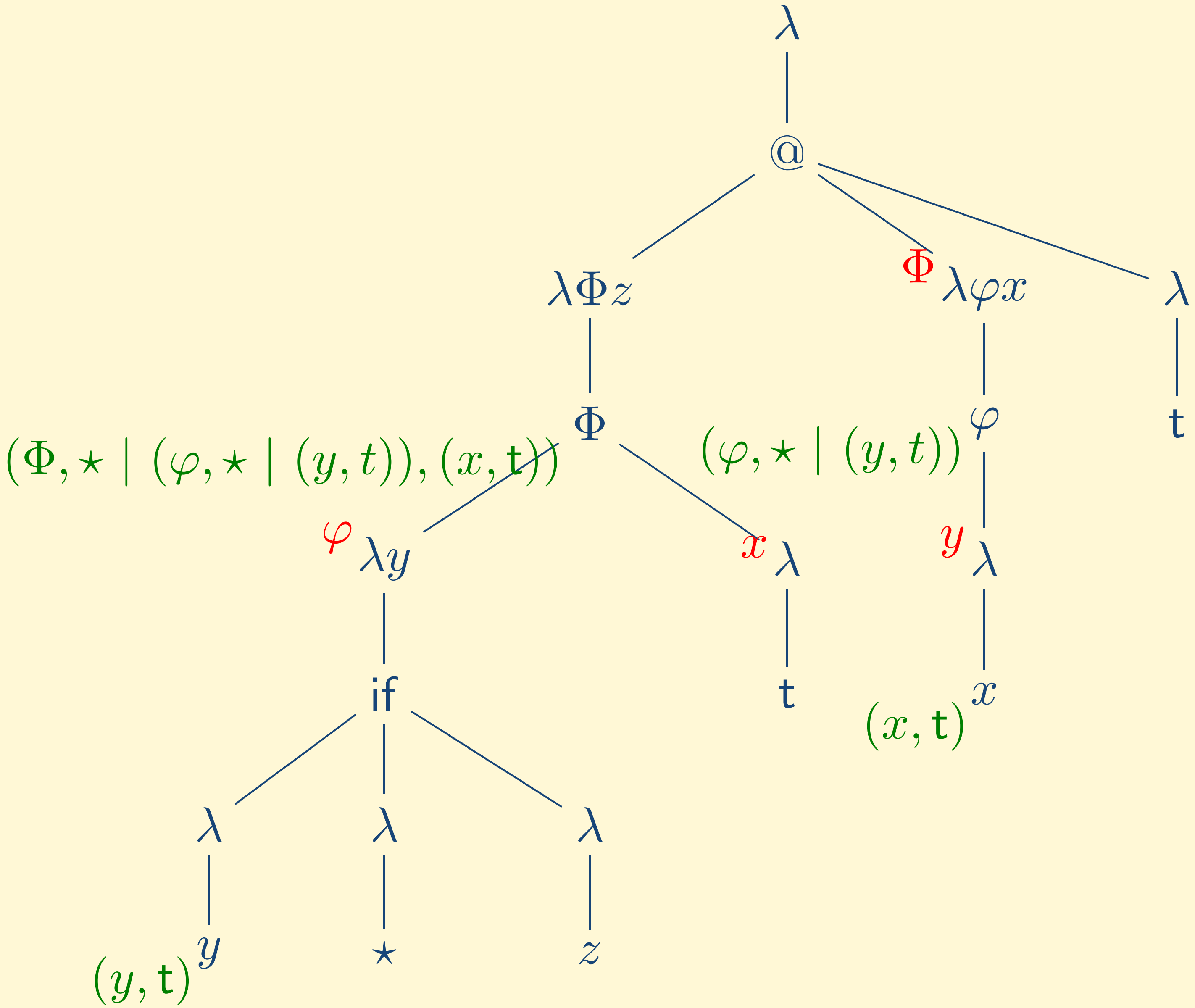
Variable profiles



Variable profiles



Variable profiles



ATA correspondence

Given a finite fPCF*-alphabet Σ, the states of the *traveral-simulating* ATA $\mathcal{A}_\Sigma$ are:

$$Q := \mathit{Val} \times \mathcal{P}(\mathbf{VP}_\Sigma) \times \mathcal{P}(\mathbf{VP}_\Sigma)$$

ATA correspondence

Given a finite fPCF*-alphabet Σ, the states of the *traveral-simulating* ATA $\mathcal{A}_\Sigma$ are:

$$Q := \mathit{Val} \times \mathcal{P}(\mathbf{VP}_\Sigma) \times \mathcal{P}(\mathbf{VP}_\Sigma)$$

- $M\vec{N} \twoheadrightarrow v$ iff $\mathcal{A}_\Sigma$ accepts $\lambda(M\vec{N})$ on initial state with *value* v.
- Any tree accepted by $\tilde{A}_\Sigma$ *is* a closed fPCF-term.

Results

Theorem: $M \in v\text{-}\textsc{Reach}\left[\mathsf{fPCF}^{\star}_{\Sigma}, \mathsf{fPCF}_{\Sigma}\right]$ iff there is an initial state q_0 with value v such that:

- $\mathcal{A}_{\Sigma}(q_o)$ accepts $\lambda(M)$,
- $\forall i$, the language accepted by $\tilde{\mathcal{A}}_{\Sigma}(q_0 \upharpoonright A_i)$ is non-empty.

Theorem: $M \in v\text{-}\textsc{Reach}\,[\mathsf{fPCF}^\star_\Sigma, \mathsf{fPCF}_\Sigma]$ iff there is an initial state q_0 with value v such that:

- $\mathcal{A}_\Sigma(q_o)$ accepts $\lambda(M)$,
- $\forall i$, the language accepted by $\tilde{\mathcal{A}}_\Sigma(q_0 \upharpoonright A_i)$ is non-empty.

Corollary: $\star\text{-}\textsc{Reach}\,[\mathsf{fPCF}^\star, \mathsf{fPCF}(n)]$ is decidable.

Corollary: $\star\text{-}\textsc{Reach}\,[\mathsf{fPCF}^\star, \mathsf{fPCF}]$ is decidable up to order 3.

Alternating Dependency Tree Automata

- For the general case we can use Alternating Dependency Tree Automata [Stirling'09].
- *Corollary:* Emptiness problem is undecidable for ADTA's.

Conclusion and on

- A new kind of Reachability problems.
- Some undecidability results.
- Some technology from game semantics.
- Characterisation by ATA's and ADTA's.
- Some (relativised) decidability results.

Conclusion and on

- A new kind of Reachability problems.
- Some undecidability results.
- Some technology from game semantics.
- Characterisation by ATA's and ADTA's.
- Some (relativised) decidability results.
- Revisit (semantic) CFA?
- Reachability through intersection types?
- Conjecture: $\star$-$\textsc{Reach}\,[\mathsf{fPCF}^\star, \mathsf{fPCF}]$?

Conclusion and on

- A new kind of Reachability problems.
- Some undecidability results.
- Some technology from game semantics.
- Characterisation by ATA's and ADTA's.
- Some (relativised) decidability results.
- Revisit (semantic) CFA?
- Reachability through intersection types?
- Conjecture: $\star$-$\textsc{Reach}\,[\mathsf{fPCF}^\star, \mathsf{fPCF}]$?

THANKS!