Functional Reachability

Nikos Tzevelekos

Oxford University Computing Laboratory

joint work with Luke Ong

Schloss Dagstuhl, June 2010

Starting Point

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 computation of $C[M]$ reaches p?

Starting Point

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 computation of $C[M]$ reaches p?

Starting Point

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 computation of $C[M]$ reaches p?

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

Idea: Use Games, Traversals, Automata.

Relevant work

- Control Flow Analysis.

  > *Compute at compile time the flow of control that is going to happen at run time.*

Relevant work

- Control Flow Analysis.

  Compute at compile time the flow of control that is going 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.

Relevant work

- Control Flow Analysis.

  > *Compute at compile time the flow of control that is going to happen at run time.*

  - In a HO-setting, the crucial element is that of *closures*.
  - Reynolds ('70), Jones ('80), Shivers ('90), $\ldots$
    Malacaria & Hankin (late 90's).
  - CFA $>$ Reach: more general.
    Reach $>$ CFA: open vs closed world approach.

- Useless code detection.
- Strictness analysis, etc.

PCF

The examined language: PCF.

- lambda-calculus,
- Boolean base type,
- recursion at all types.

$$
\begin{aligned}
A, B &::= o \mid A \to B \\
v &::= \mathrm{t} \mid \mathrm{f} \\
M, N &::= v \mid x \mid \lambda x.M \mid M\,N \mid \textsf{if}\ M\ N_1\ N_2 \mid \mathbf{Y}_A
\end{aligned}
$$

PCF

$$
\begin{aligned}
A, B &::= o \mid A \to B \\
v &::= \mathrm{t} \mid \mathrm{f} \\
M, N &::= v \mid x \mid \lambda x.M \mid M\,N \mid \textsf{if}\ M\ N_1\ N_2 \mid \mathbf{Y}_A
\end{aligned}
$$

PCF

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

$$v ::= \mathrm{t} \mid \mathrm{f}$$

$$M, N ::= v \mid x \mid \lambda x.M \mid M\,N \mid \textsf{if}\, M\, N_1\, N_2 \mid \mathbf{Y}_A$$

$$\begin{array}{rclcrcl}
(\lambda x.M)N & \to & M\{N/x\} & \quad & \textsf{if}\,\mathrm{t} & \to & \lambda xy.x \\
\mathbf{Y}M & \to & M(\mathbf{Y}M) & & \textsf{if}\,\mathrm{f} & \to & \lambda xy.y
\end{array}$$

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

$$\mathrm{E} ::= [-] \mid \mathrm{E}\,M \mid \textsf{if}\,\mathrm{E}$$

PCF

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

$$v ::= \mathrm{t} \mid \mathrm{f}$$

$$M, N ::= v \mid x \mid \lambda x.M \mid M\,N \mid \text{if}\; M\; N_1\; N_2 \mid \mathbf{Y}_A$$

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. $\perp := \mathbf{Y}_o(\lambda x.x)$.

PCF

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

$$v \ ::= \ \mathrm{t} \mid \mathrm{f}$$

$$M, N \ ::= \ v \mid x \mid \lambda x.M \mid M\,N \mid \text{if}\ M\ N_1\ N_2 \mid \mathbf{Y}_A$$

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 $\quad M, N \ ::= \ v \mid x \mid \lambda x.M \mid M\,N \mid \text{if}\ M\ N_1\ N_2$

  $\text{fPCF}_\bot \quad M, N \ ::= \ v \mid x \mid \lambda x.M \mid M\,N \mid \text{if}\ M\ N_1\ N_2 \mid \bot$

Reachability

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

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

Reachability

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

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

We can make things simpler

PCF-with-error: PCF*

- Include an error constant: $o = \{\mathrm{t}, \mathrm{f}, \star\}$
- New rules: $\mathrm{E}[\star] \twoheadrightarrow \star$

PCF-with-error: PCF*

- Include an error constant: $o = \{\mathrm{t}, \mathrm{f}, \star\}$
- New rules: $\mathrm{E}[\star] \twoheadrightarrow \star$

$\star$-Reachability:

- Given a closed $\mathrm{PCF}^\star$-term M with *exactly one* $\star$,
- Are there closed PCF-terms $N_1, \ldots, N_n$ such that $M\vec{N} \twoheadrightarrow \star$?

PCF-with-error: PCF*

- Include an error constant: $o = \{\mathrm{t}, \mathrm{f}, \star\}$
- New rules: $\mathrm{E}[\star] \twoheadrightarrow \star$

Reachability $\cong$ $\star$-Reachability

$\star$-Reachability:

- Given a closed $\mathrm{PCF}^\star$-term M with *exactly one* $\star$,
- Are there closed PCF-terms $N_1, \ldots, N_n$ such that $M\vec{N} \twoheadrightarrow \star$?

$\textsc{Reach}$ template

v-$\textsc{Reach}\,[\mathcal{L}_1, \mathcal{L}_2]$:

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

- Given a closed $\mathcal{L}_1$-term M ,
- Are there closed $\mathcal{L}_2$-terms $N_1, \ldots, N_n$ such that $M\,\vec{N} \twoheadrightarrow v$?

E.g. $\left\{ \begin{array}{l} \star\text{-Reachability} \;=\; \star\text{-}\textsc{Reach}\,[\mathrm{PCF}^{1\star}, \mathrm{PCF}] \\ \\ \end{array} \right.$

$\textsc{Reach}$ template

v-$\textsc{Reach}\,[\mathcal{L}_1, \mathcal{L}_2]$:

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

- Given a closed $\mathcal{L}_1$-term M ,
- Are there closed $\mathcal{L}_2$-terms $N_1, \ldots, N_n$ such that $M\,\vec{N} \twoheadrightarrow v$?

E.g. $\begin{cases} \star\text{-Reachability} \;=\; \star\text{-}\textsc{Reach}\,[\mathrm{PCF}^{1\star}, \mathrm{PCF}] \\ v\text{-}\textsc{Reach}\,[\mathcal{L}, \mathrm{PCF}] \;=\; v\text{-}\textsc{Reach}\,[\mathcal{L}, \mathrm{fPCF}] \end{cases}$

$\textsc{Reach}$ template

v-$\textsc{Reach}\,[\mathcal{L}_1, \mathcal{L}_2]$:

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

- Given a closed $\mathcal{L}_1$-term M ,
- Are there closed $\mathcal{L}_2$-terms $N_1, \ldots, N_n$ such that $M\,\vec{N} \twoheadrightarrow v$?

E.g. $\begin{cases} \star\text{-Reachability} = \star\text{-}\textsc{Reach}\,[\mathrm{PCF}^{1\star}, \mathrm{PCF}] \\ v\text{-}\textsc{Reach}\,[\mathcal{L}, \mathrm{PCF}] = v\text{-}\textsc{Reach}\,[\mathcal{L}, \mathrm{fPCF}] \end{cases}$

From [Loader]:

Observational equivalence in $\mathrm{fPCF}_\perp$ is undecidable.

therefore:

t-$\textsc{Reach}\,[\mathrm{fPCF}_\perp, \mathrm{fPCF}]$ is undecidable.

Undecidability

The following problems are undecidable.

- t-$\textsc{Reach}\,[\mathrm{fPCF}_{\perp}, \mathrm{fPCF}]$
- $\star$-$\textsc{Reach}\,[\mathrm{fPCF}_{\perp}^{1\star}, \mathrm{fPCF}]$
- $\star$-Reachability, i.e. $\star$-$\textsc{Reach}\,[\mathrm{PCF}^{1\star}, \mathrm{PCF}]$
- Reachability

Undecidability

The following problems are undecidable.

- t-REACH $[\mathrm{fPCF}_{\perp}, \mathrm{fPCF}]$
- $\star$-REACH $[\mathrm{fPCF}_{\perp}^{1\star}, \mathrm{fPCF}]$
- $\star$-Reachability, i.e. $\star$-REACH $[\mathrm{PCF}^{1\star}, \mathrm{PCF}]$
- Reachability

Not all is lost

- $\star$-REACH $[\mathrm{fPCF}^{1\star}, \mathrm{fPCF}]$?
- Reachability for *finitary* M ?
- $\star$-REACH $[\mathrm{fPCF}^{\star}, \mathrm{fPCF}]$?

Our approach

We focus on v-REACH $[\mathrm{fPCF}^{\star}, \mathrm{fPCF}]$.

Computations of $\mathrm{fPCF}^{\star}$-term $P : o$

$\updownarrow$

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

$\updownarrow$

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

Our approach

We focus on v-REACH $[\mathrm{fPCF}^\star, \mathrm{fPCF}]$.

Computations of $\mathrm{fPCF}^\star$-term $P : o$

$\updownarrow$

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

$\updownarrow$

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 $\mathrm{fPCF}^{\star}$-term M,

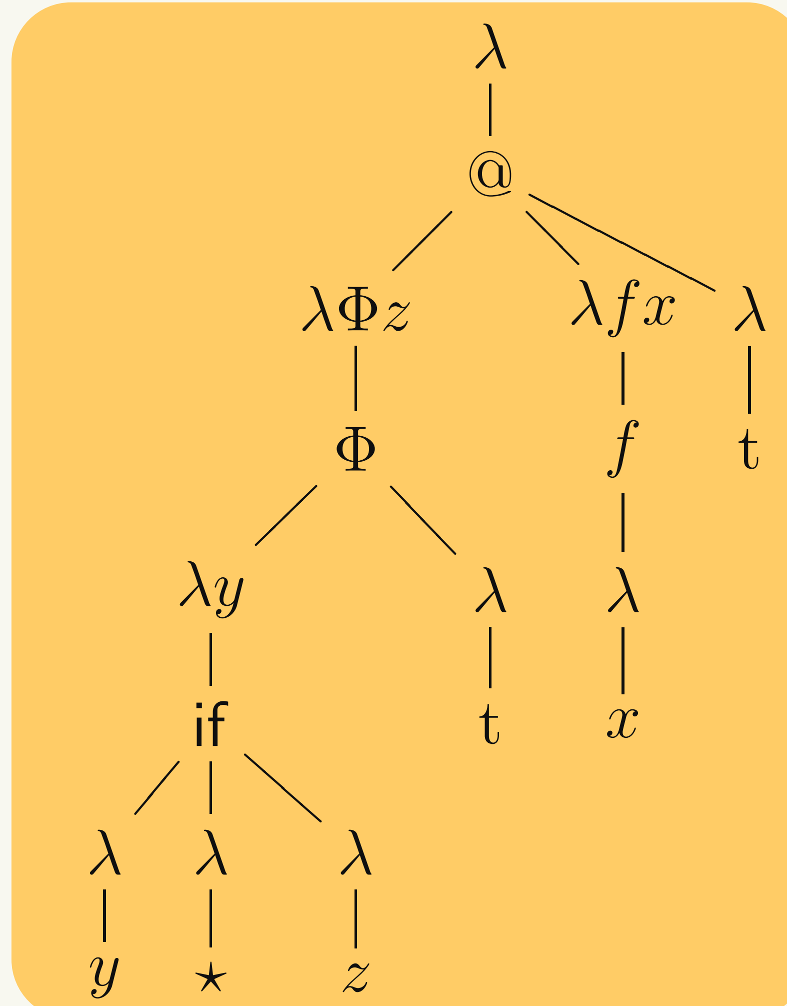
- take its η-long form,
- add application symbols ($@$),
- view the result as a tree, $\lambda(M)$.

Computation trees

Starting from a $\mathrm{fPCF}^{\star}$-term M,

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

$$(\,\lambda \Phi z.\, \Phi(\lambda y.\, \mathsf{if}\, y \star z)\mathrm{t}\,)\,(\lambda f x.\, f x)\,\mathrm{t} \qquad \longmapsto$$

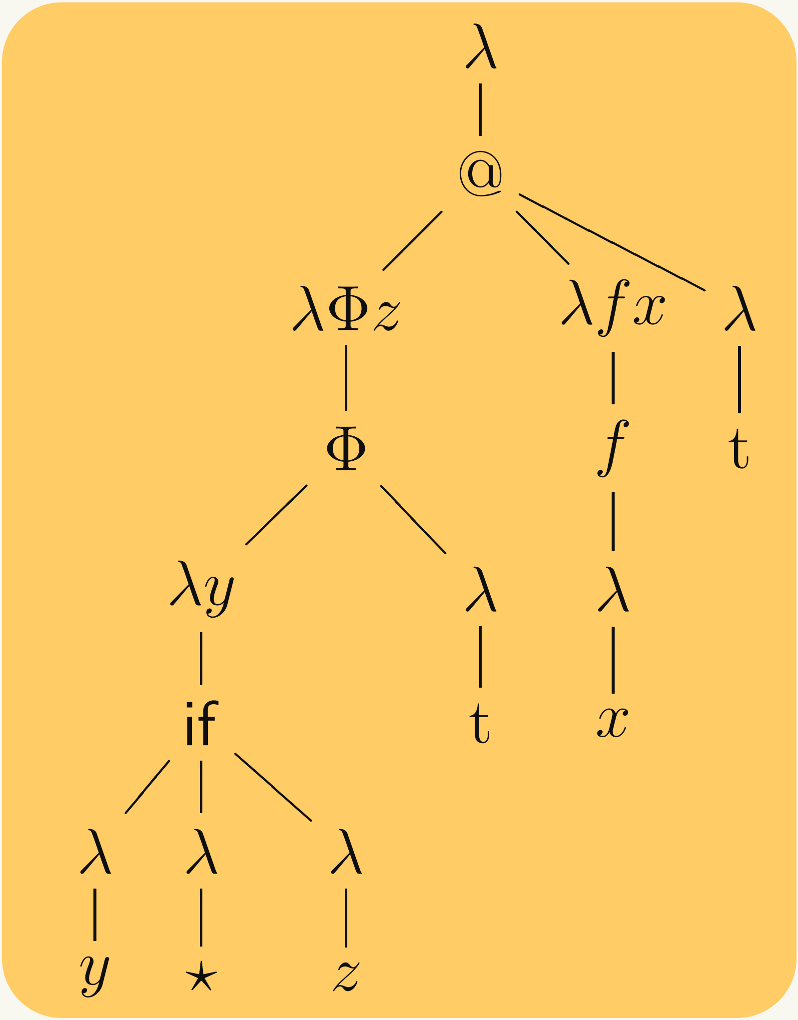


Computation trees

Starting from a $\mathrm{fPCF}^{\star}$-term M,

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

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



For a finite $\mathrm{fPCF}^{\star}$-alphabet Σ:

- $\lambda(M)$ is a Σ-labelled tree.
- $\lambda(M)$ is a Σ-labelled *binding* tree.

[Stirling'09]

Traversals

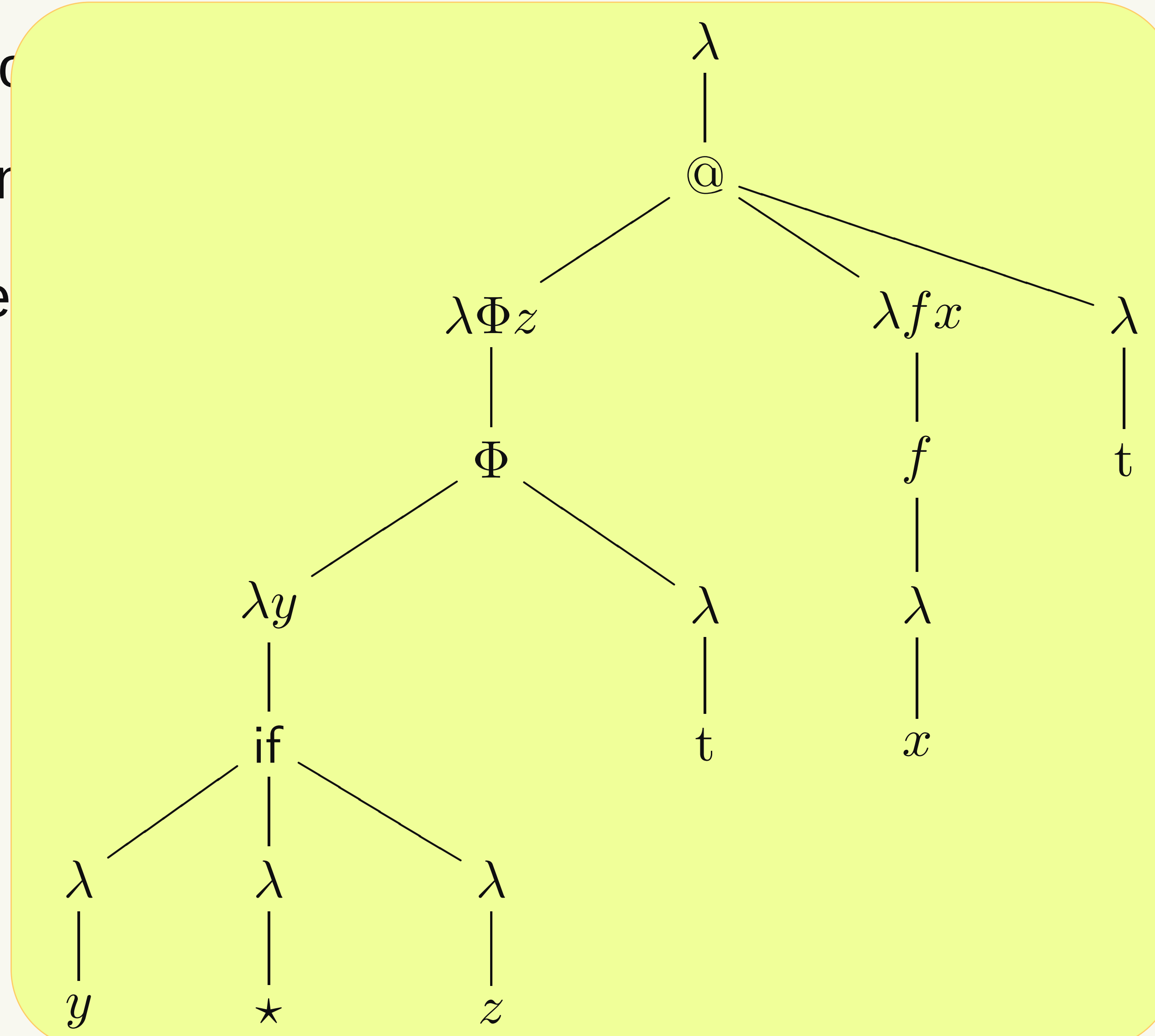
A traversal [Blum, Ong] over a *full* computation tree:

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

Traversals

A traversal [Blum, Ong] c

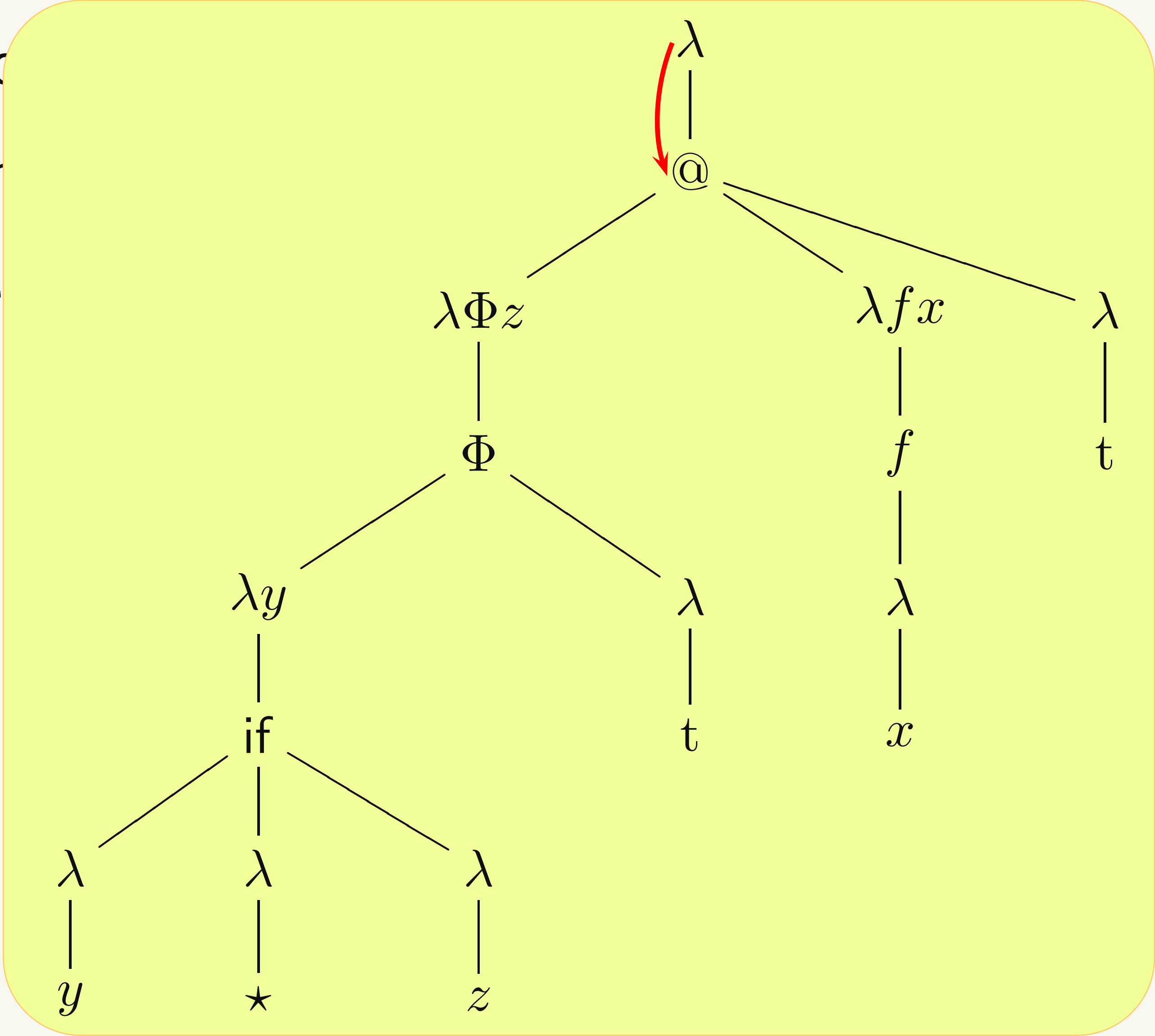
- follows the flow of cor
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

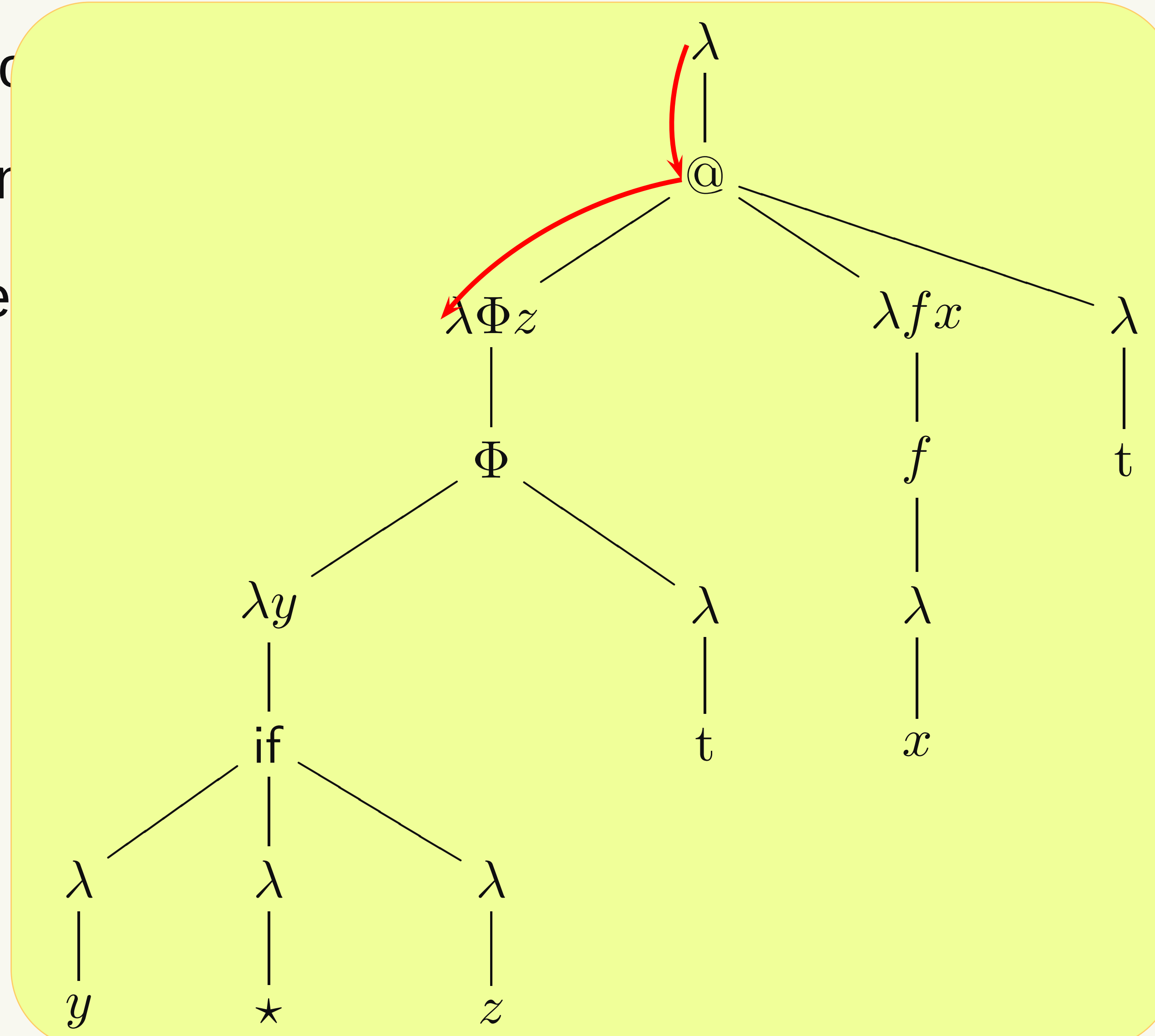
- follows the flow of cor
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

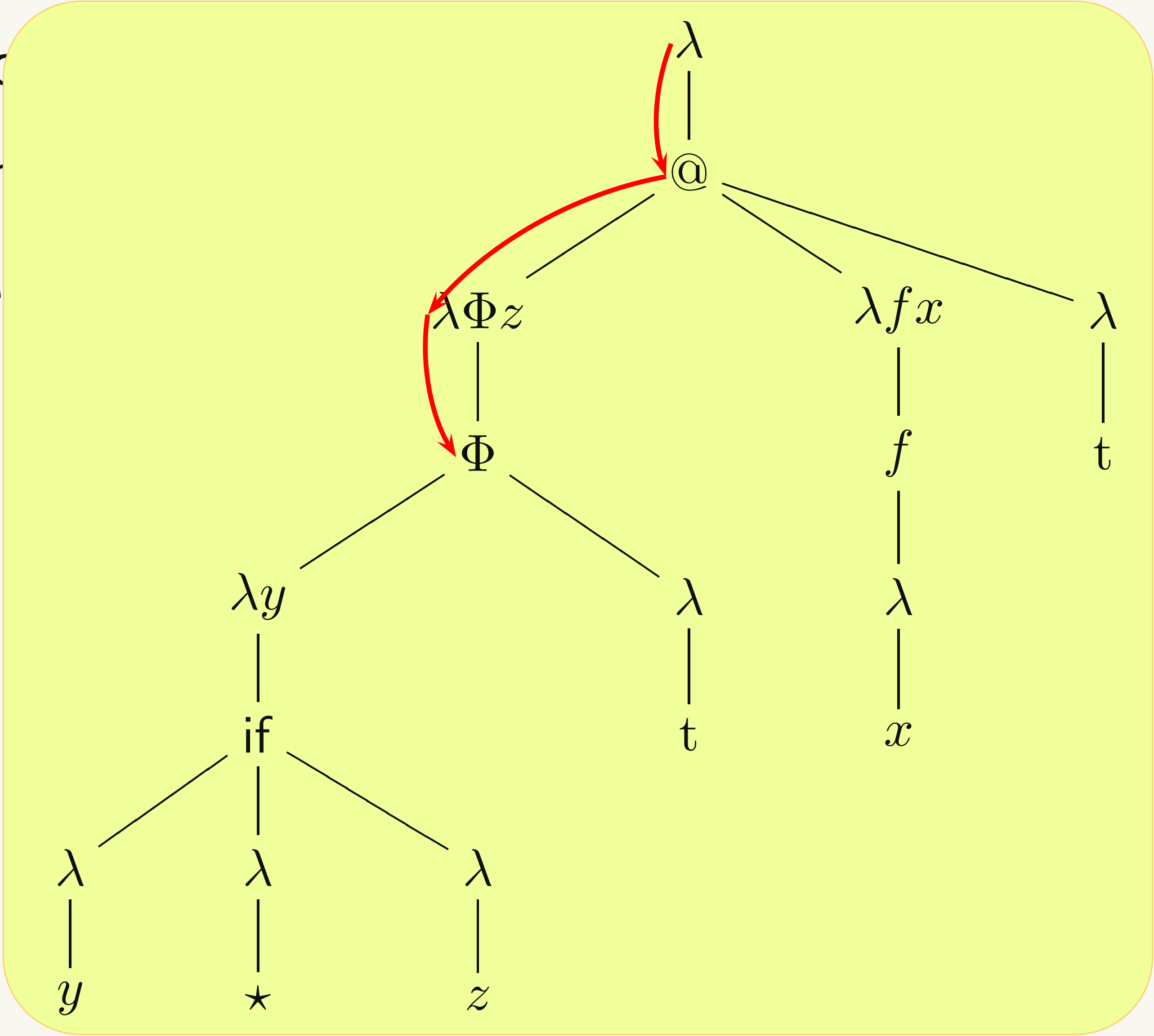
- follows the flow of cor
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

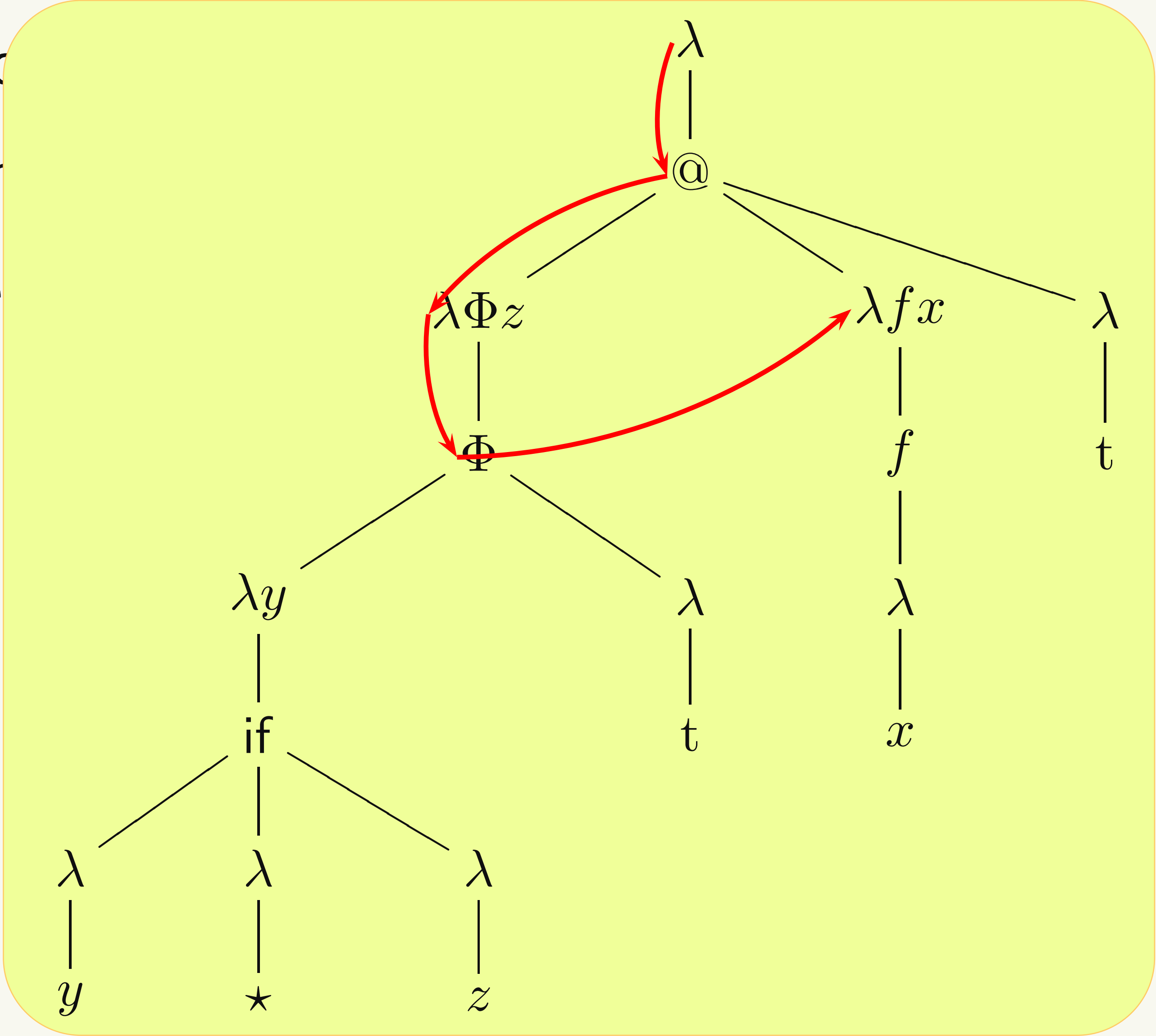
- follows the flow of cor
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

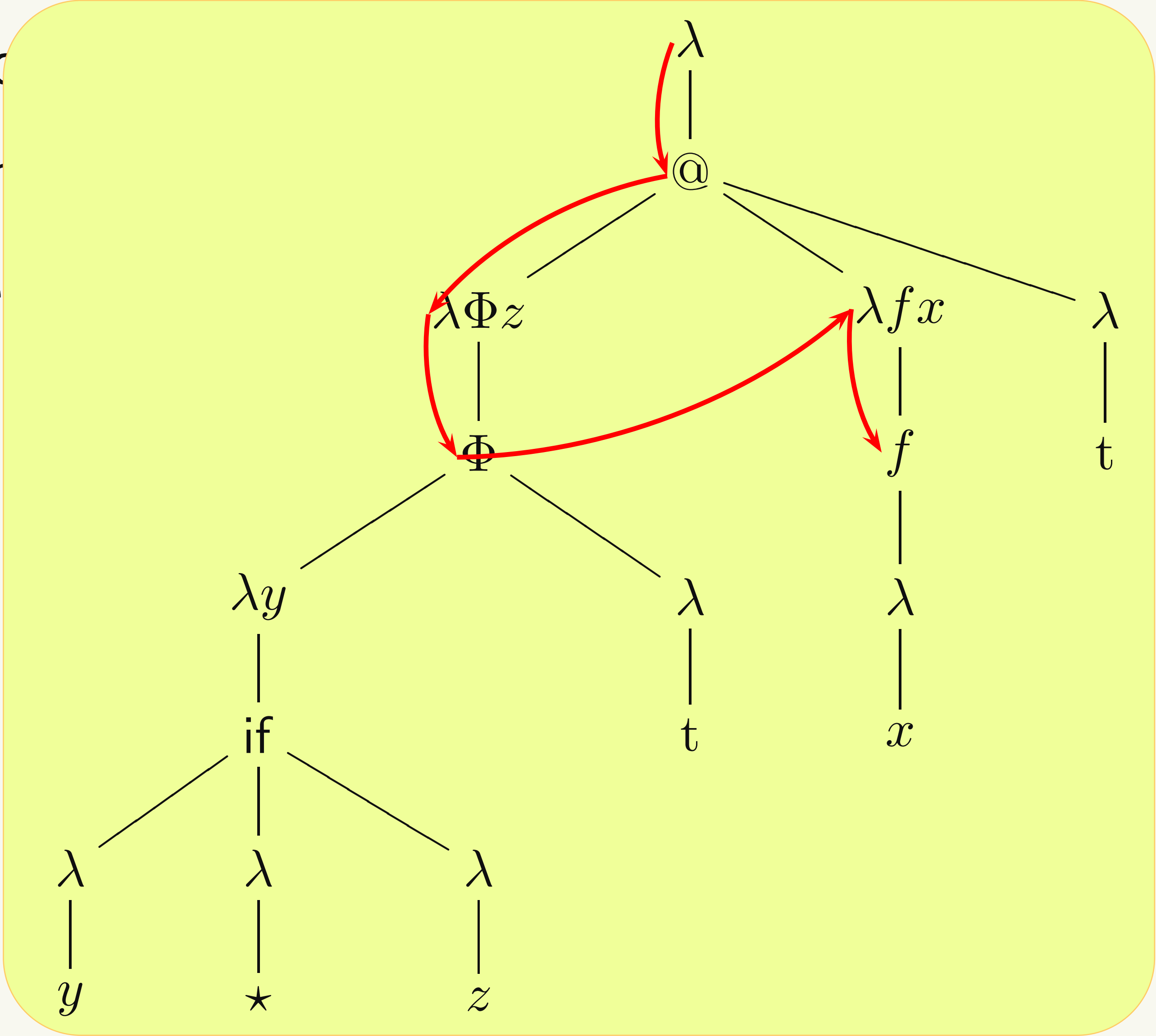
- follows the flow of cor
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

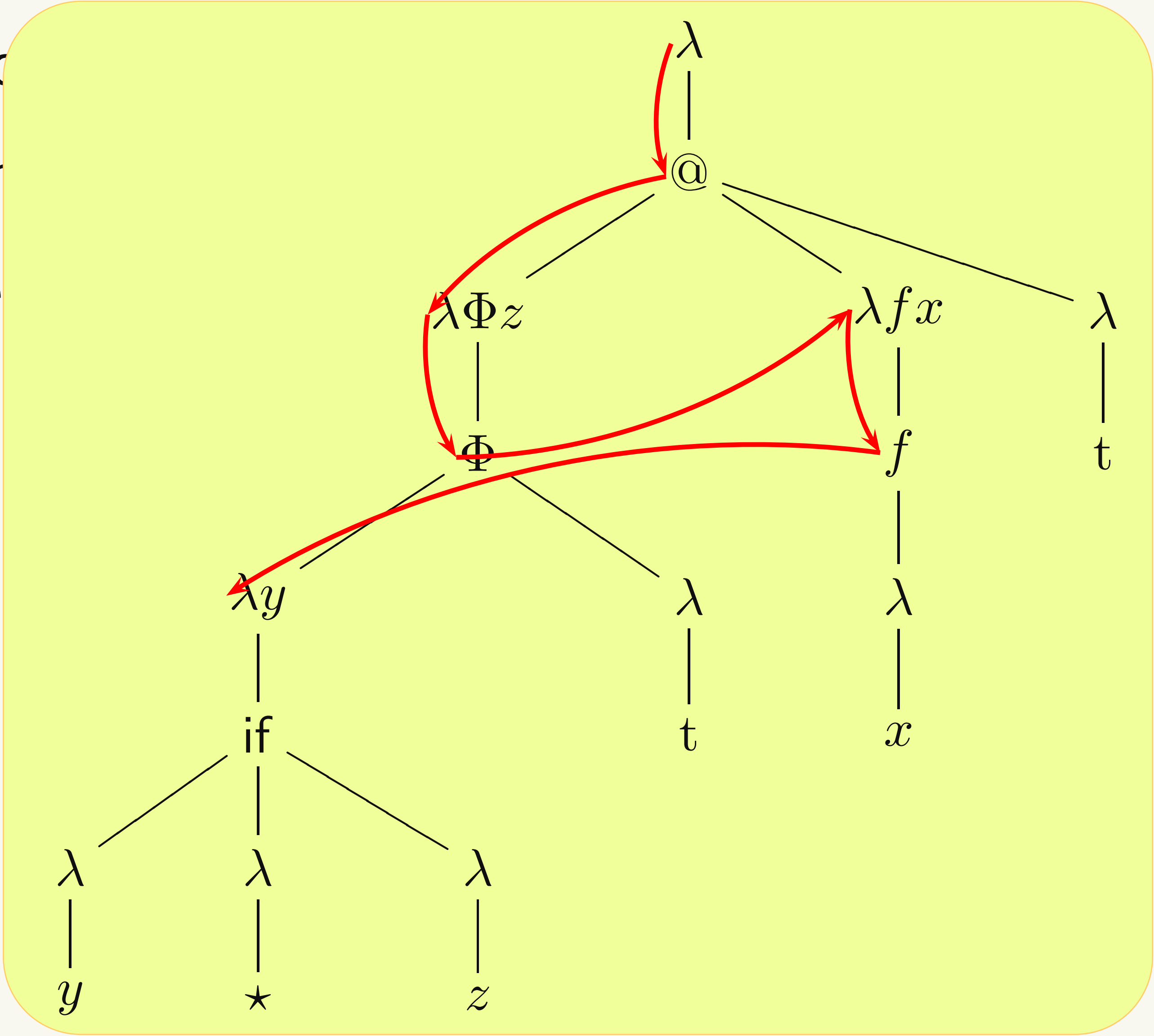
- follows the flow of co
- seen from the perspe



Traversals

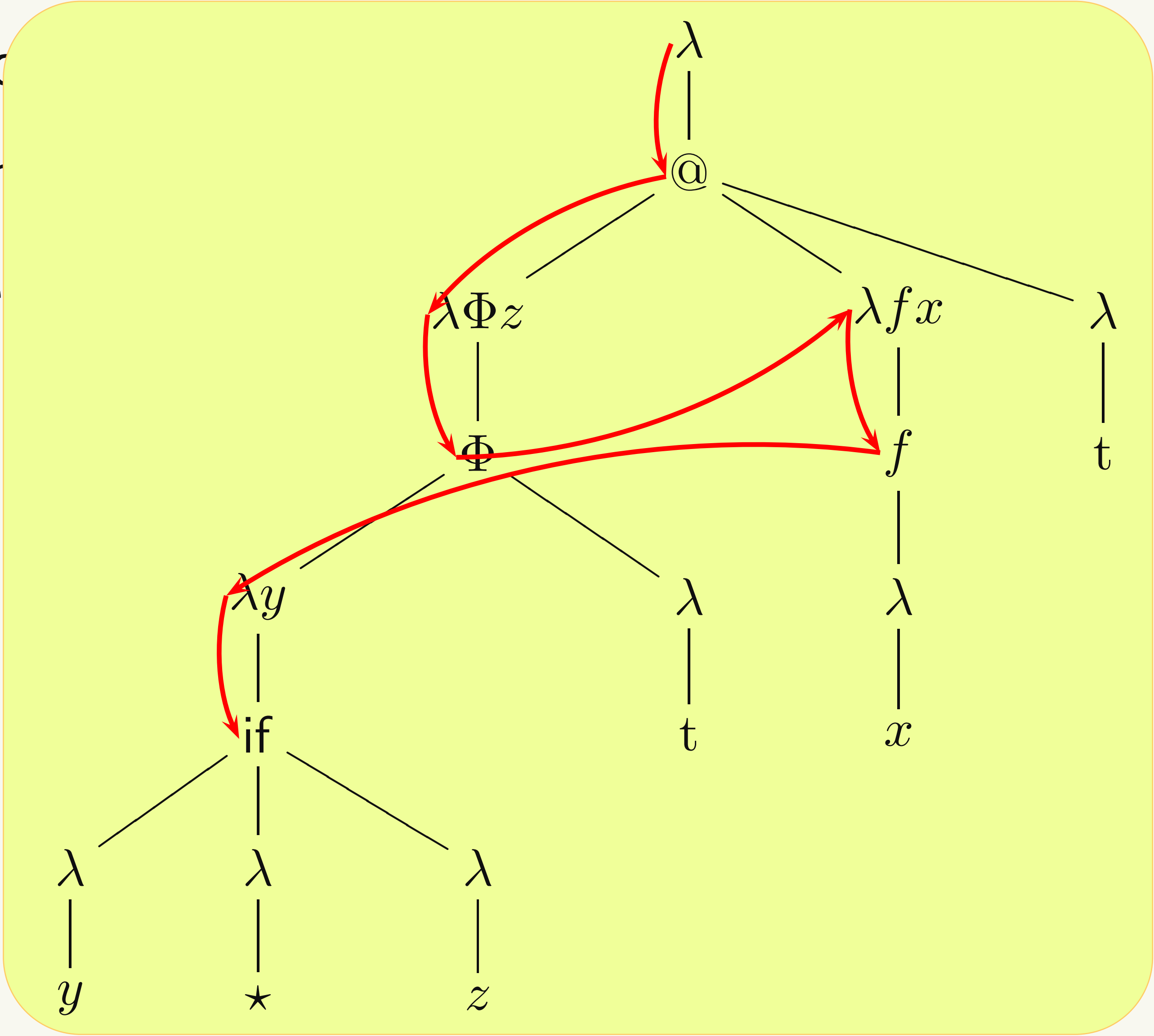
A traversal [Blum, Ong] c

- follows the flow of cor
- seen from the perspe



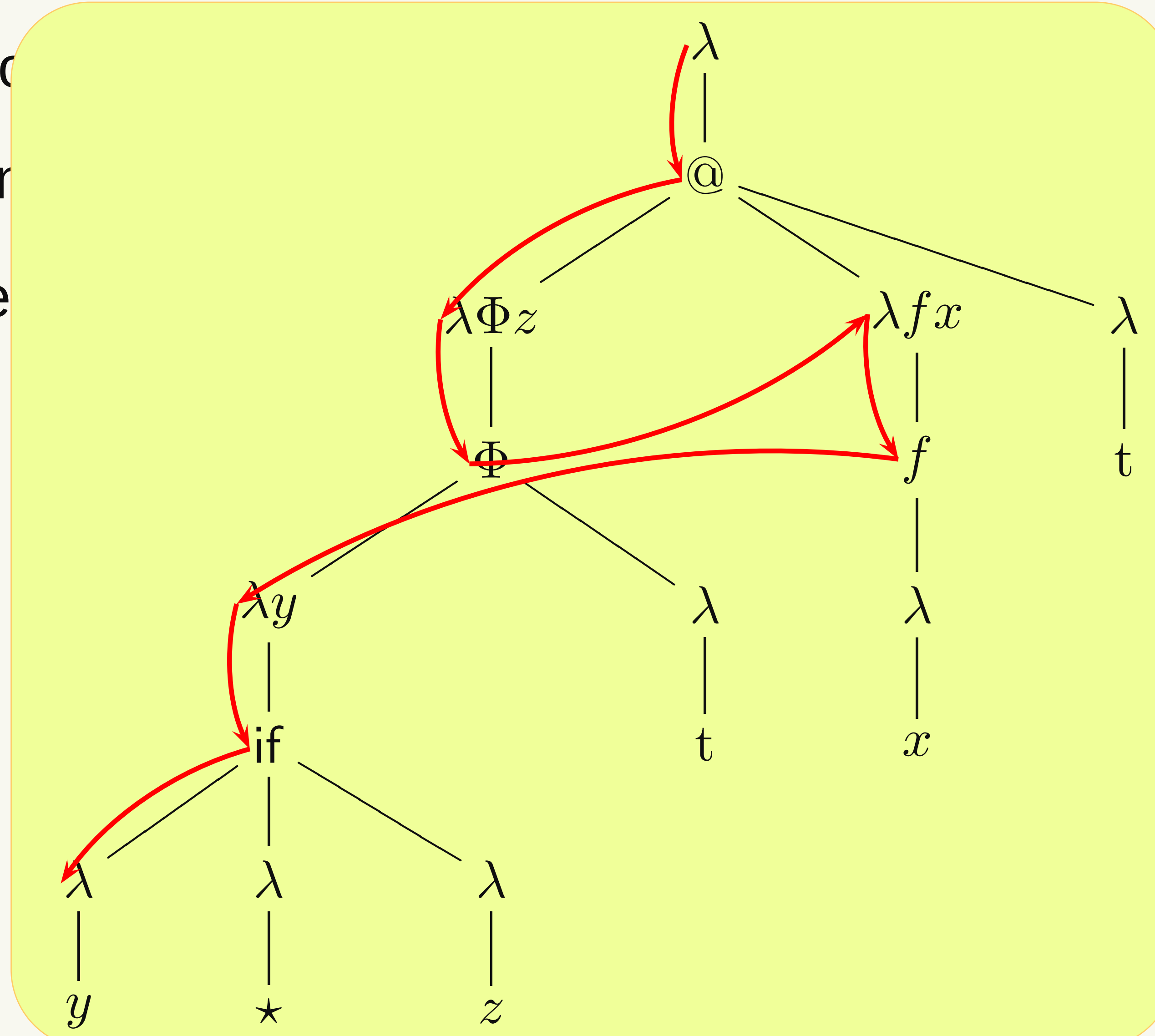
A traversal [Blum, Ong] c

- follows the flow of cor
- seen from the perspe



A traversal [Blum, Ong] c

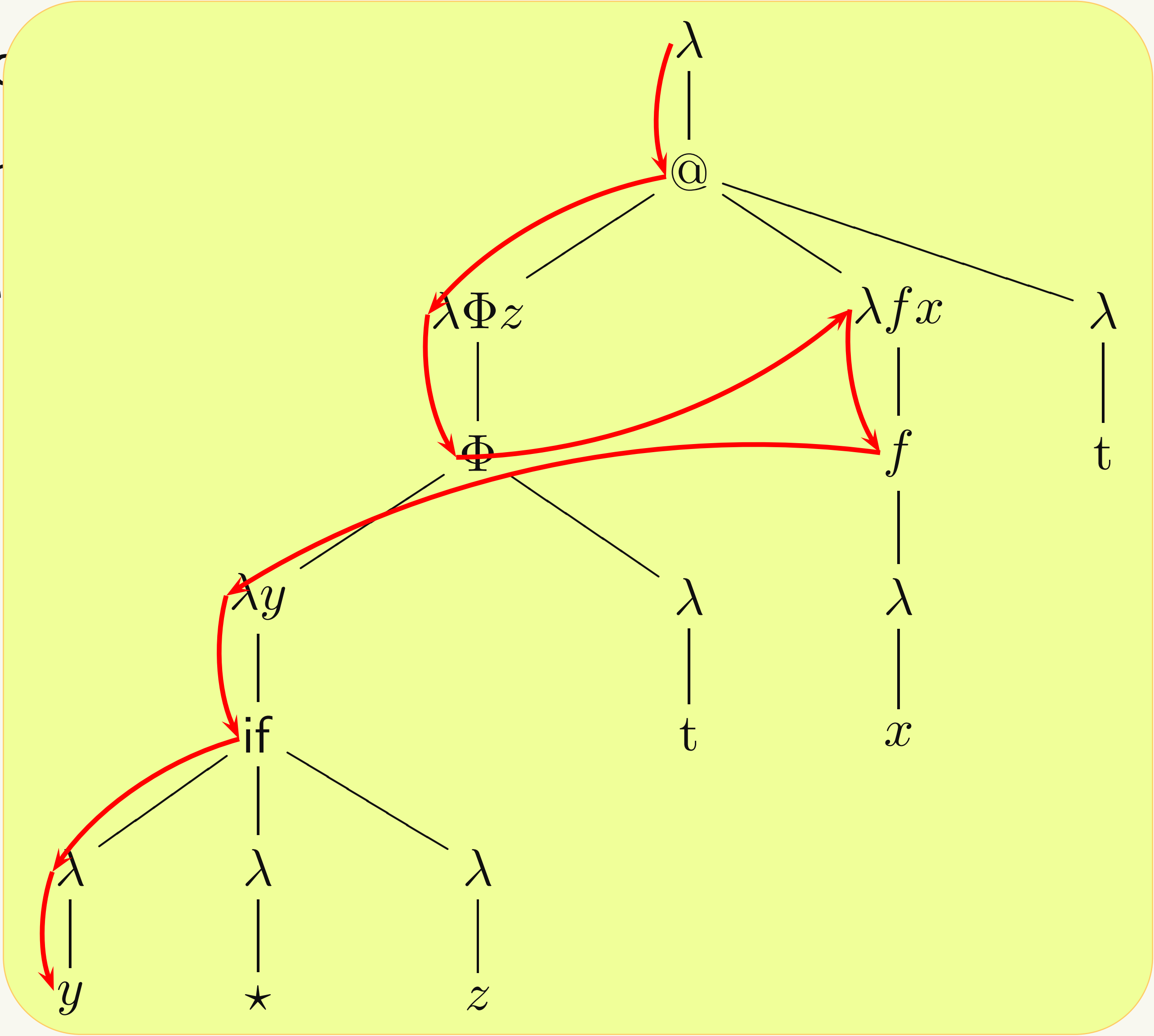
- follows the flow of cor
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

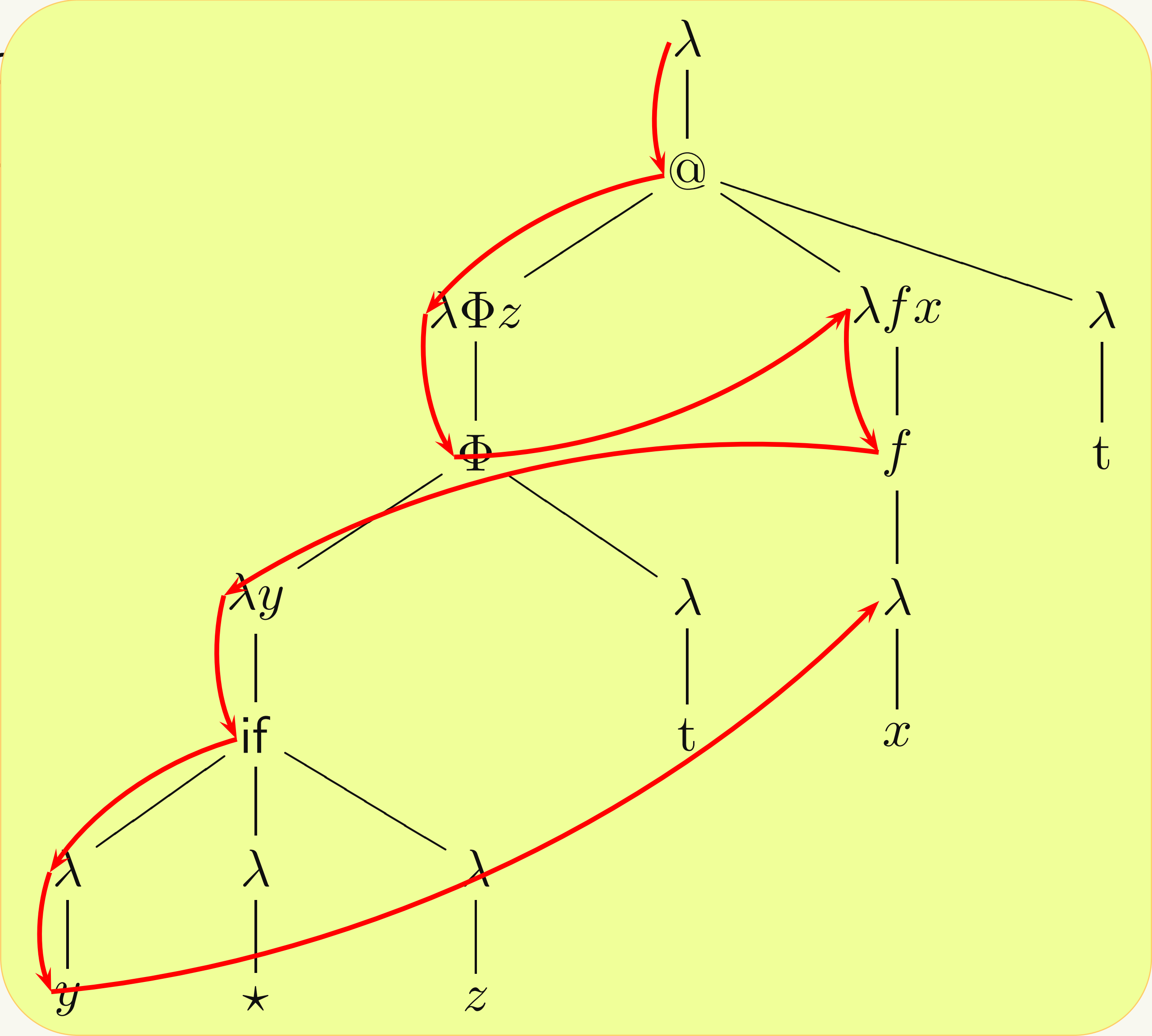
- follows the flow of cor
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

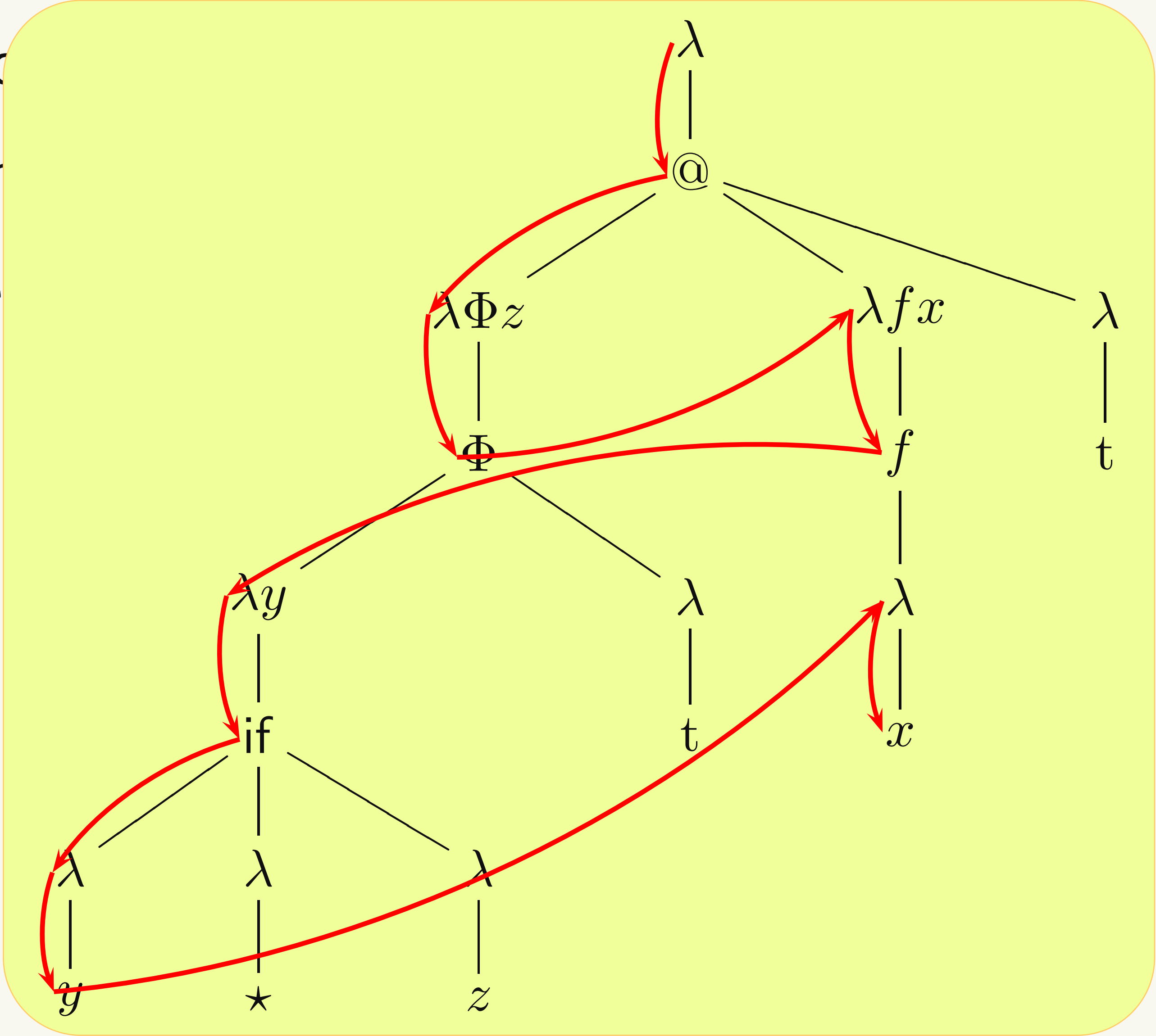
- follows the flow of cor
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

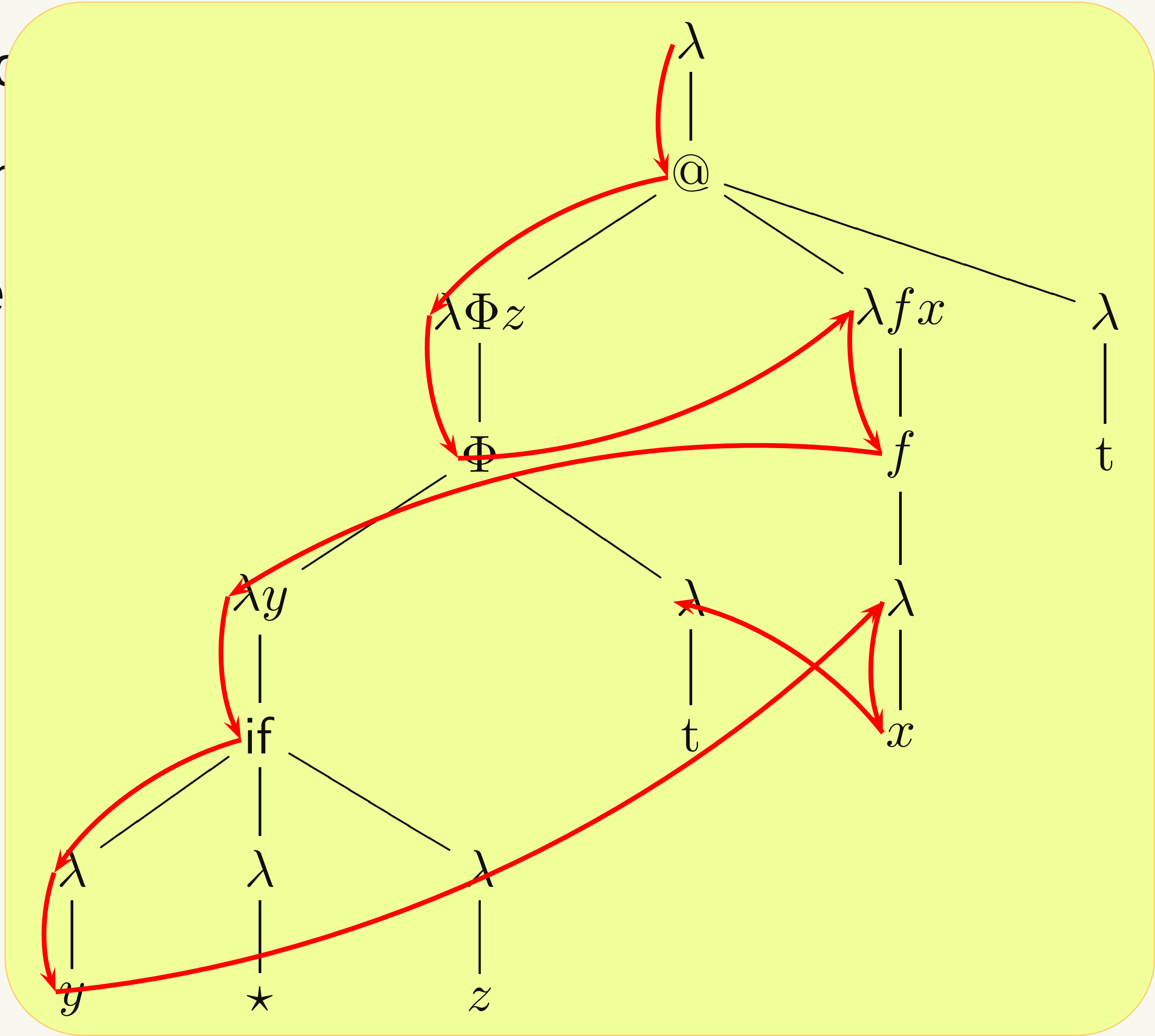
- follows the flow of con
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

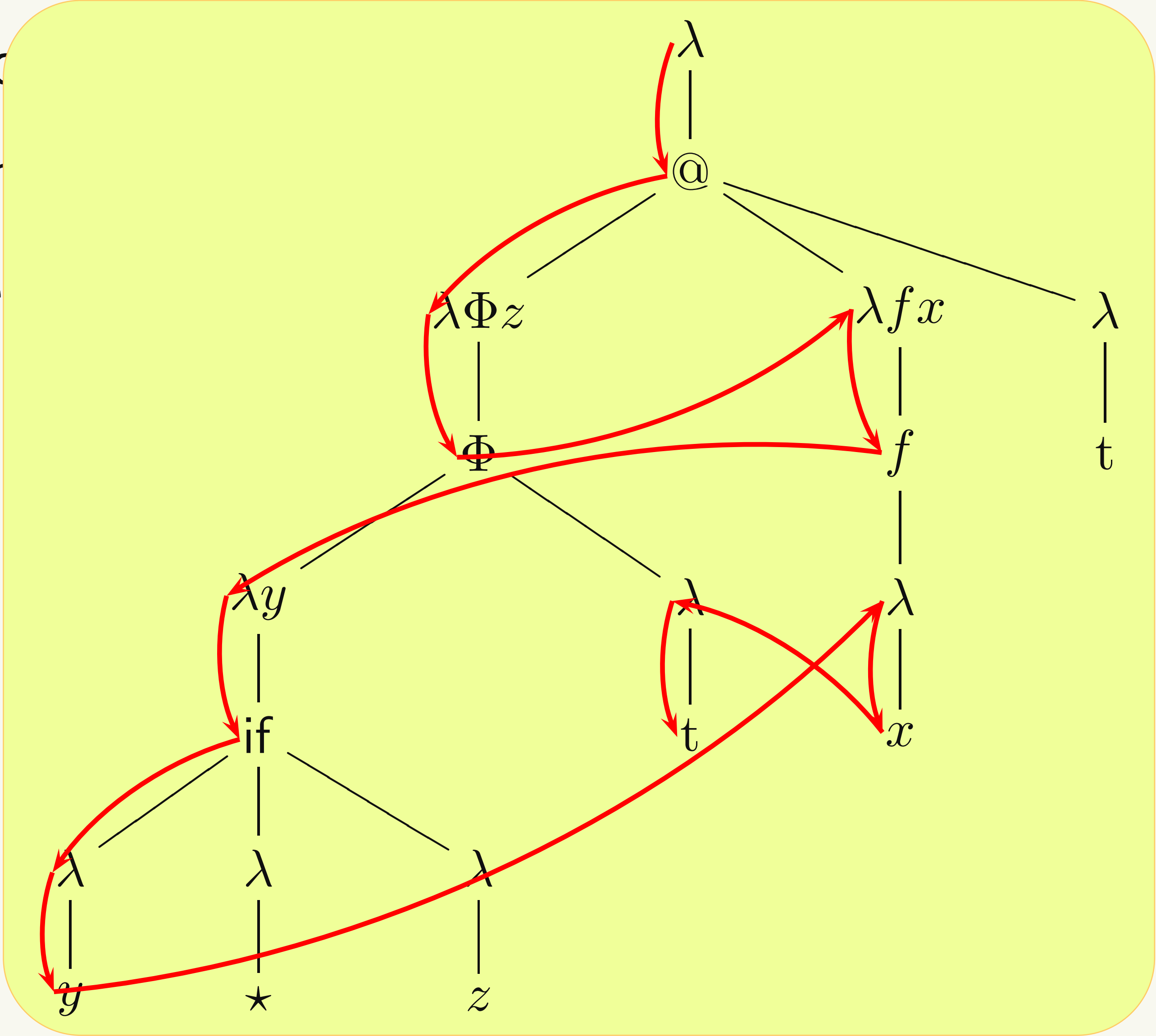
- follows the flow of cor
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

- follows the flow of cor
- seen from the perspe

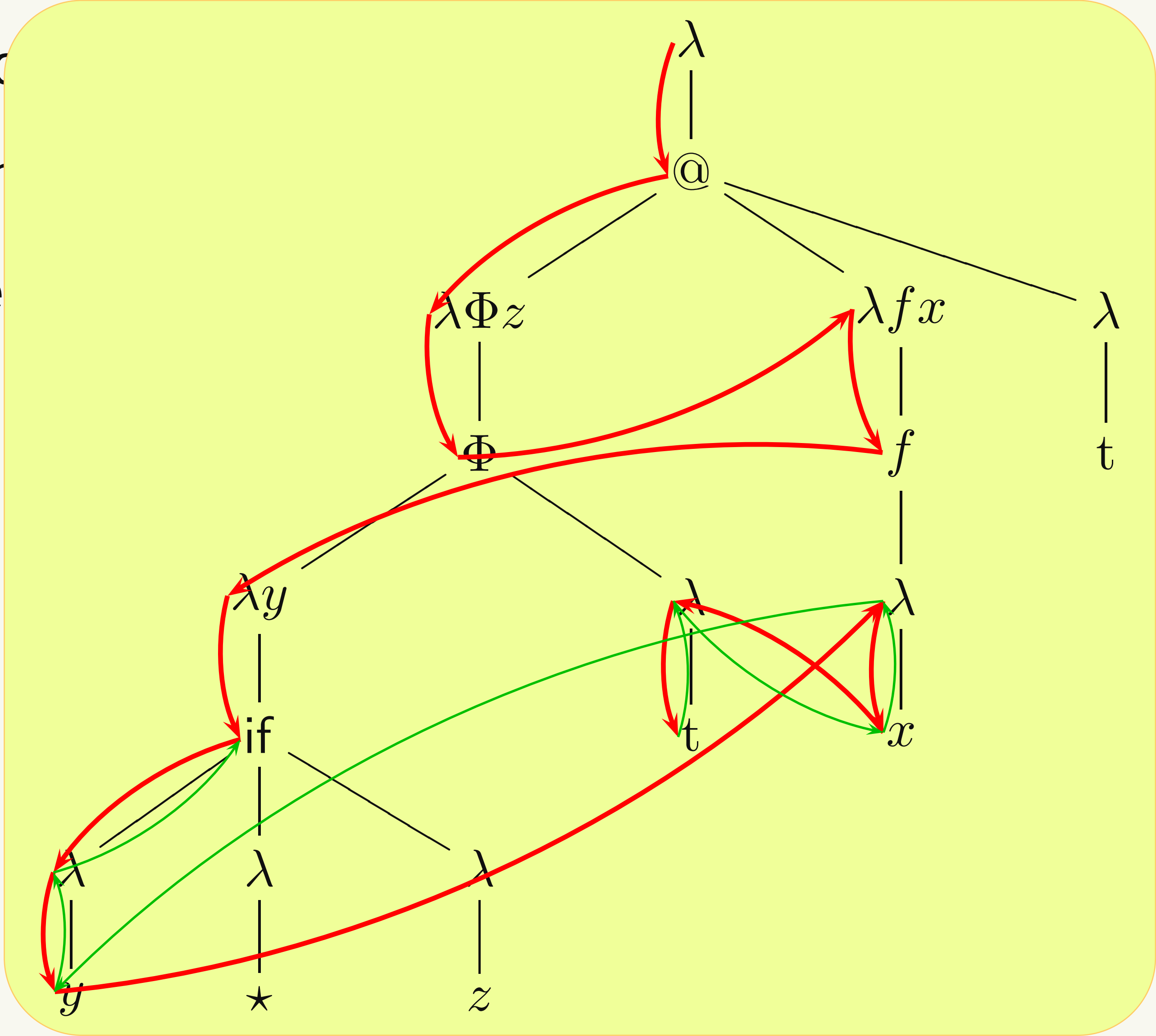


Traversals

A traversal [Blum, Ong] c

- follows the flow of cor

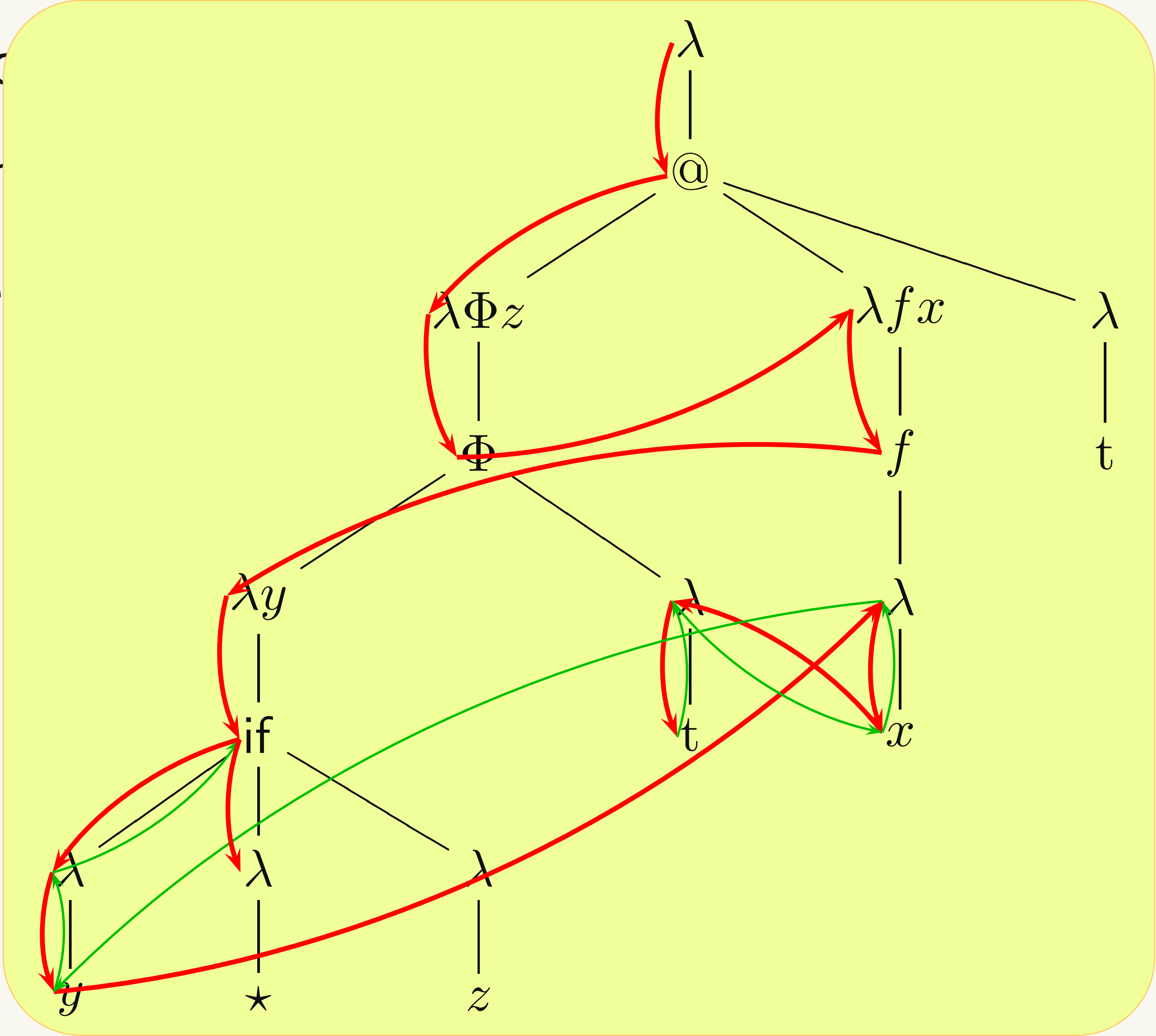
- seen from the perspe



Traversals

A traversal [Blum, Ong] c

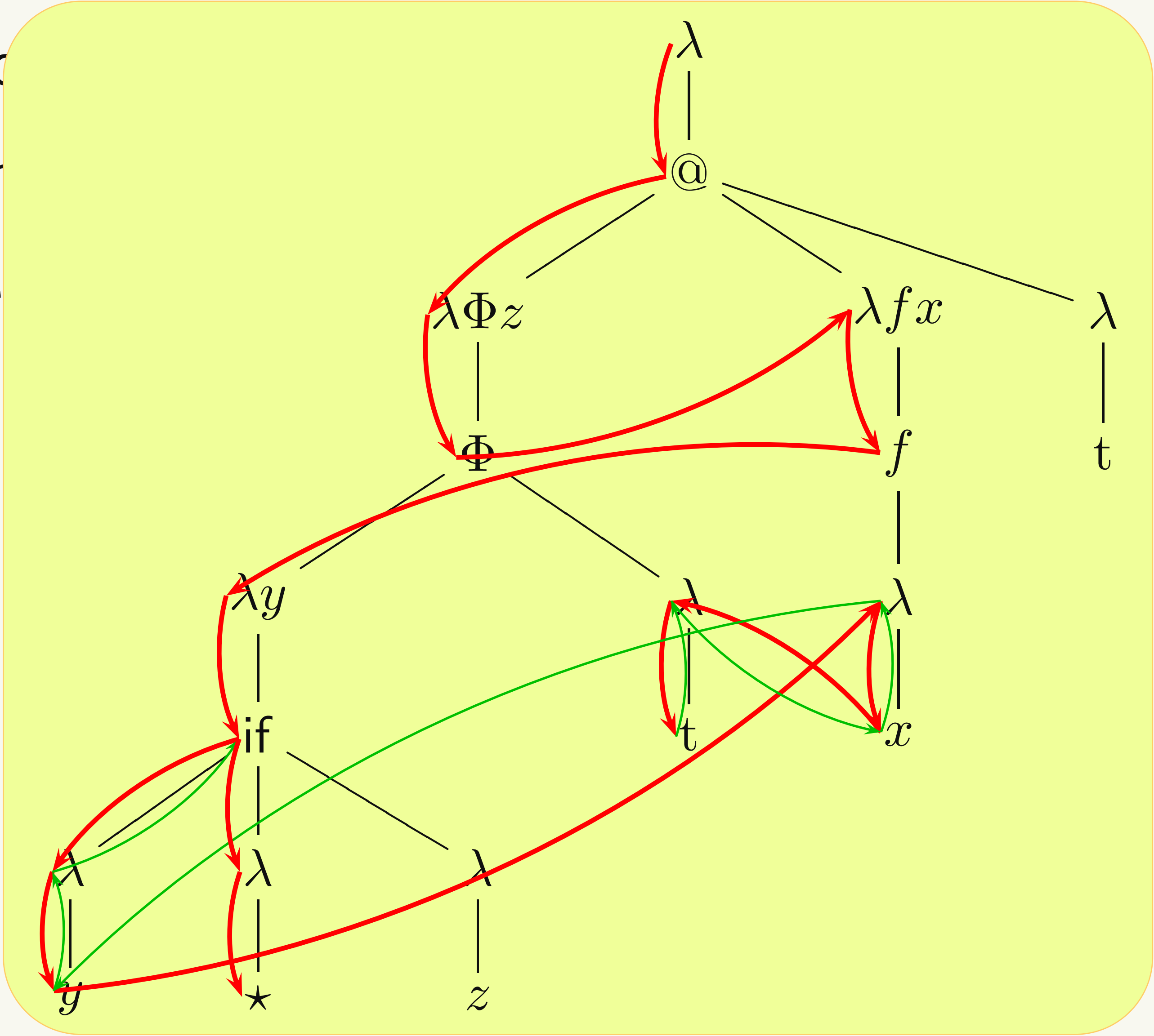
- follows the flow of cor
- seen from the perspe



Traversals

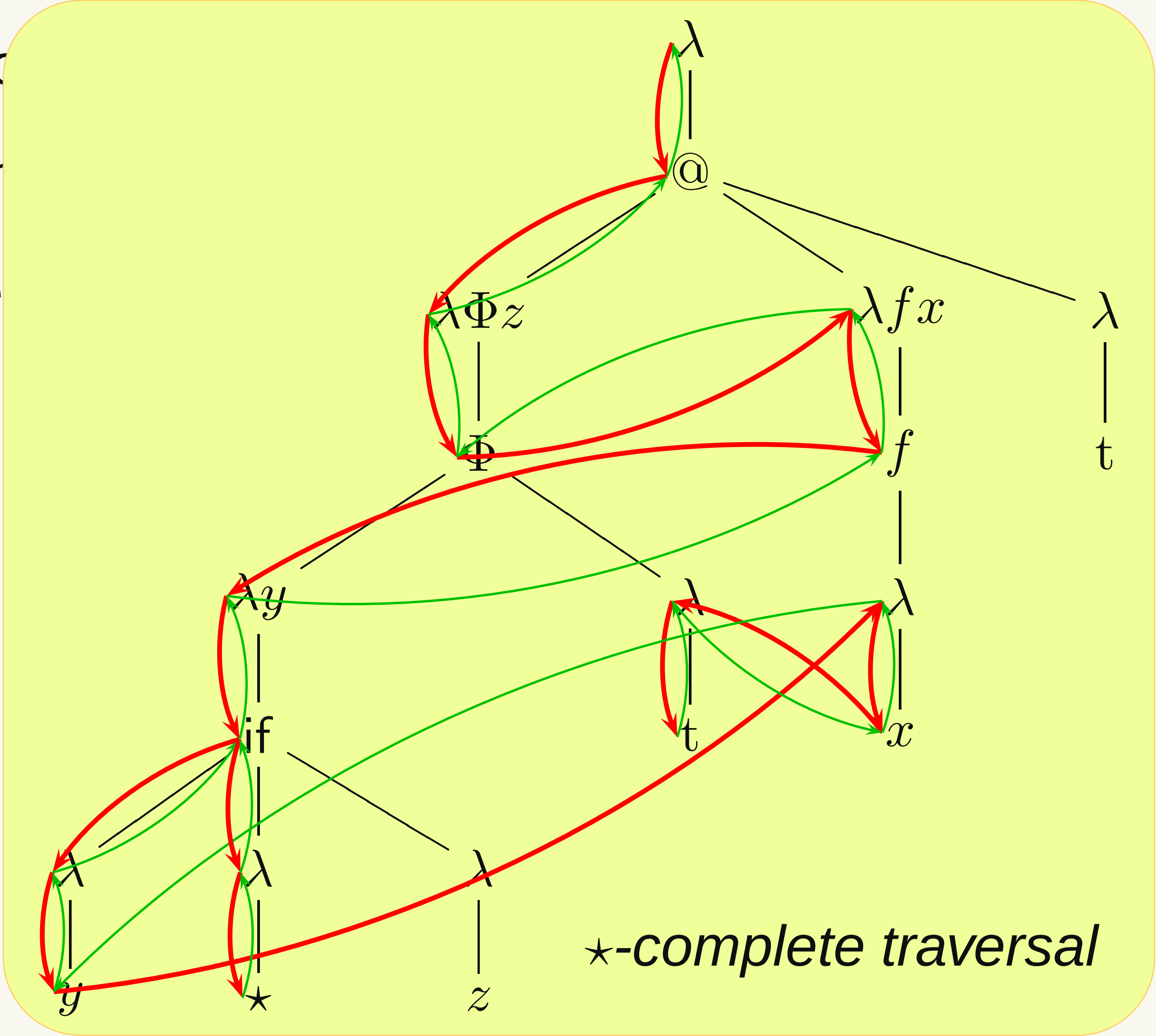
A traversal [Blum, Ong] o

- follows the flow of cor
- seen from the perspe



A traversal [Blum, Ong] c

- follows the flow of cor
- seen from the perspe



Traversals

A traversal [Blum, Ong] over a *full* 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.

For any $P : o$ and v, $P \twoheadrightarrow v$ iff there is a v-complete 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: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.

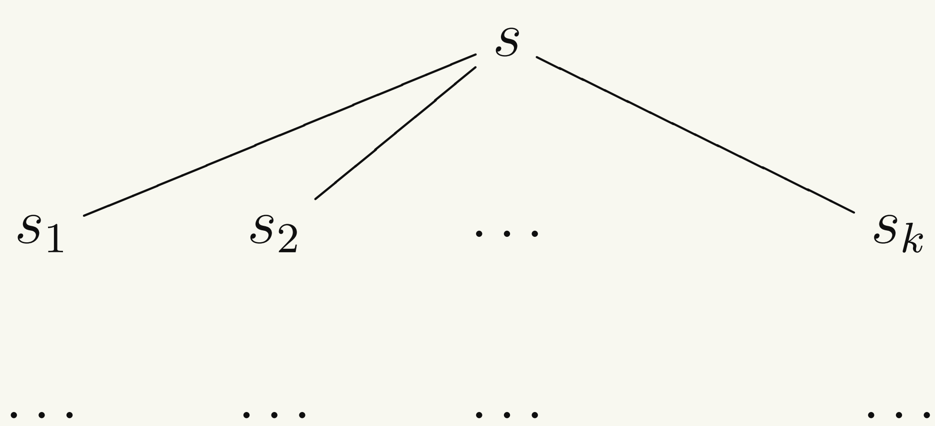
$s \in \Sigma$
$q \in Q$
$Q_1, \ldots, Q_k \subseteq Q$

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: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.

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

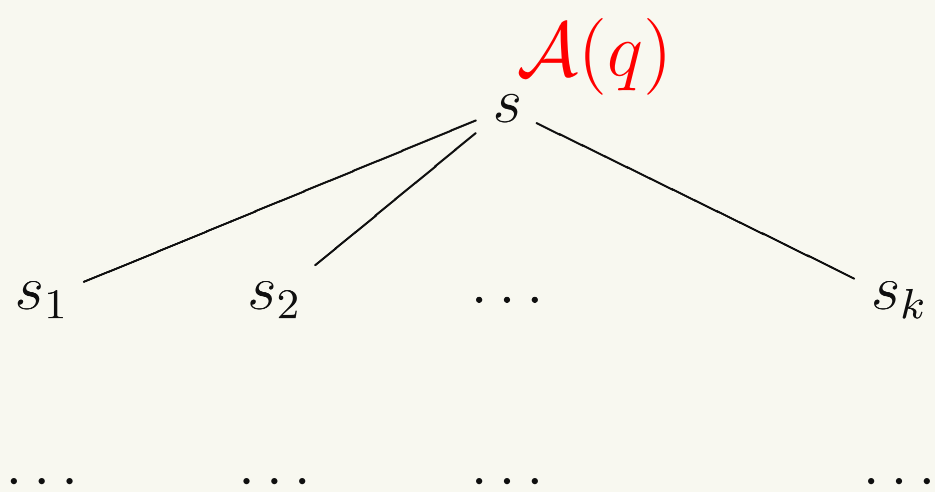


Alternating Tree Automata

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: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.

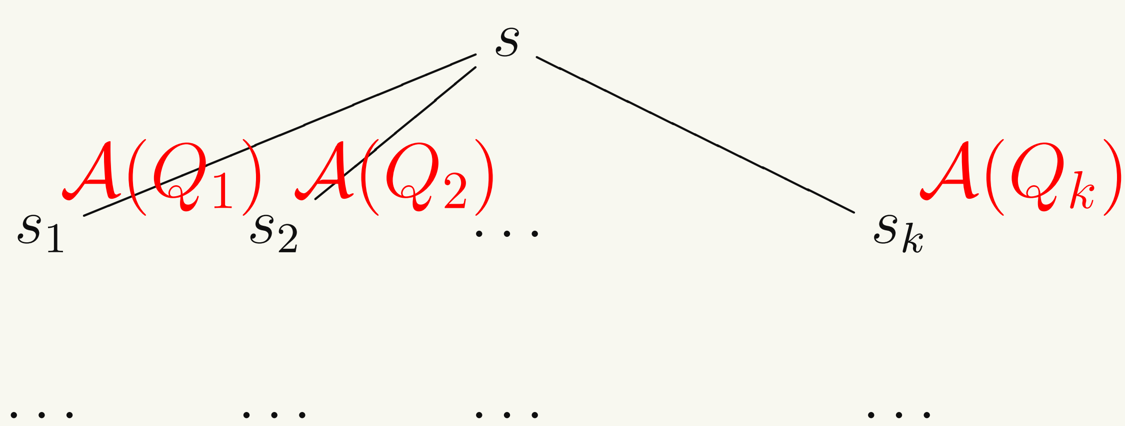
$s \in \Sigma$
$q \in Q$
$Q_1, \ldots, Q_k \subseteq Q$



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

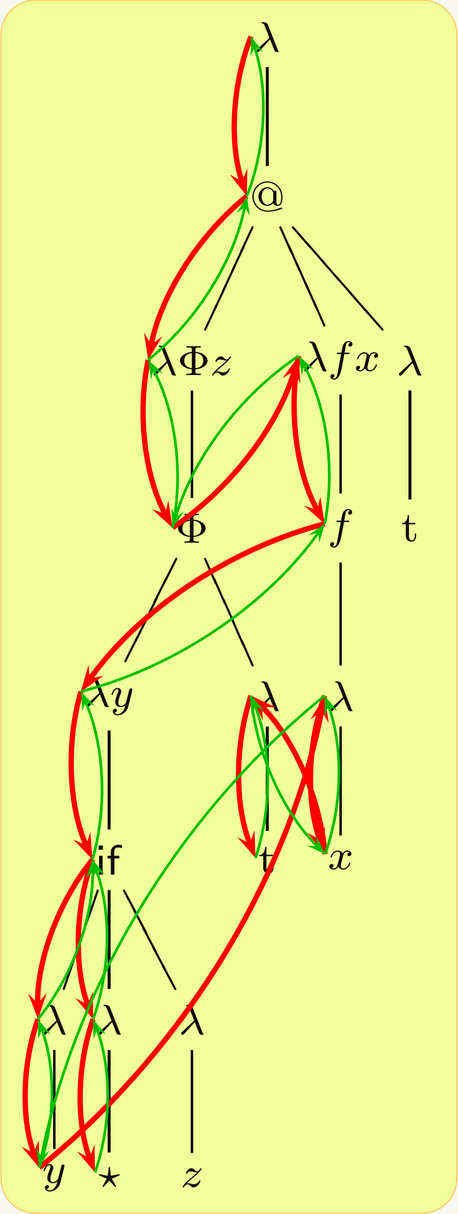
- Q is a finite set of states,
- Σ is a finite ranked alphabet,
- $q_0 \in Q$ is the initial state,
- Δ is a finite transition relation: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.

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



Traversal-simulating ATA’s

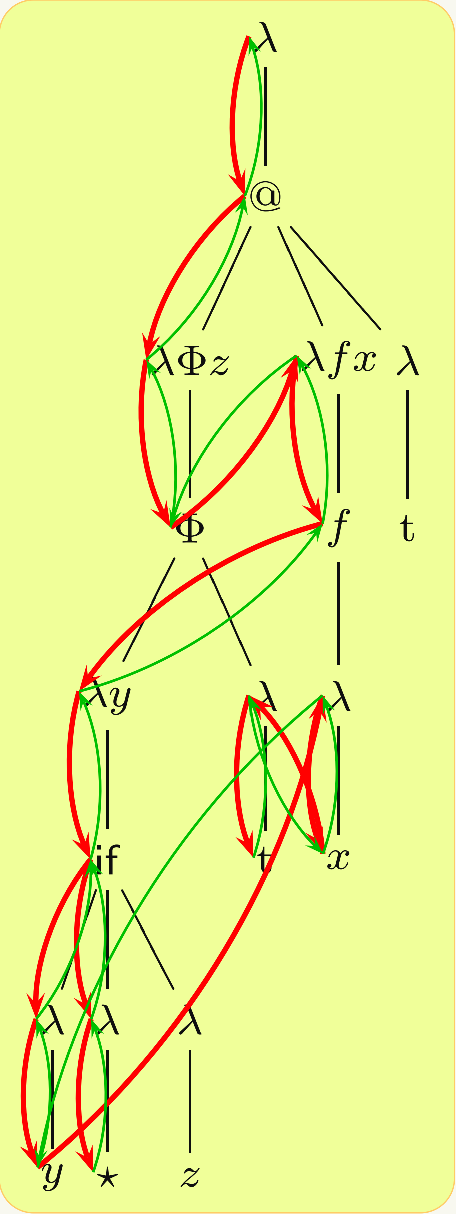
How can we simulate a complete traversal by an ATA?



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.



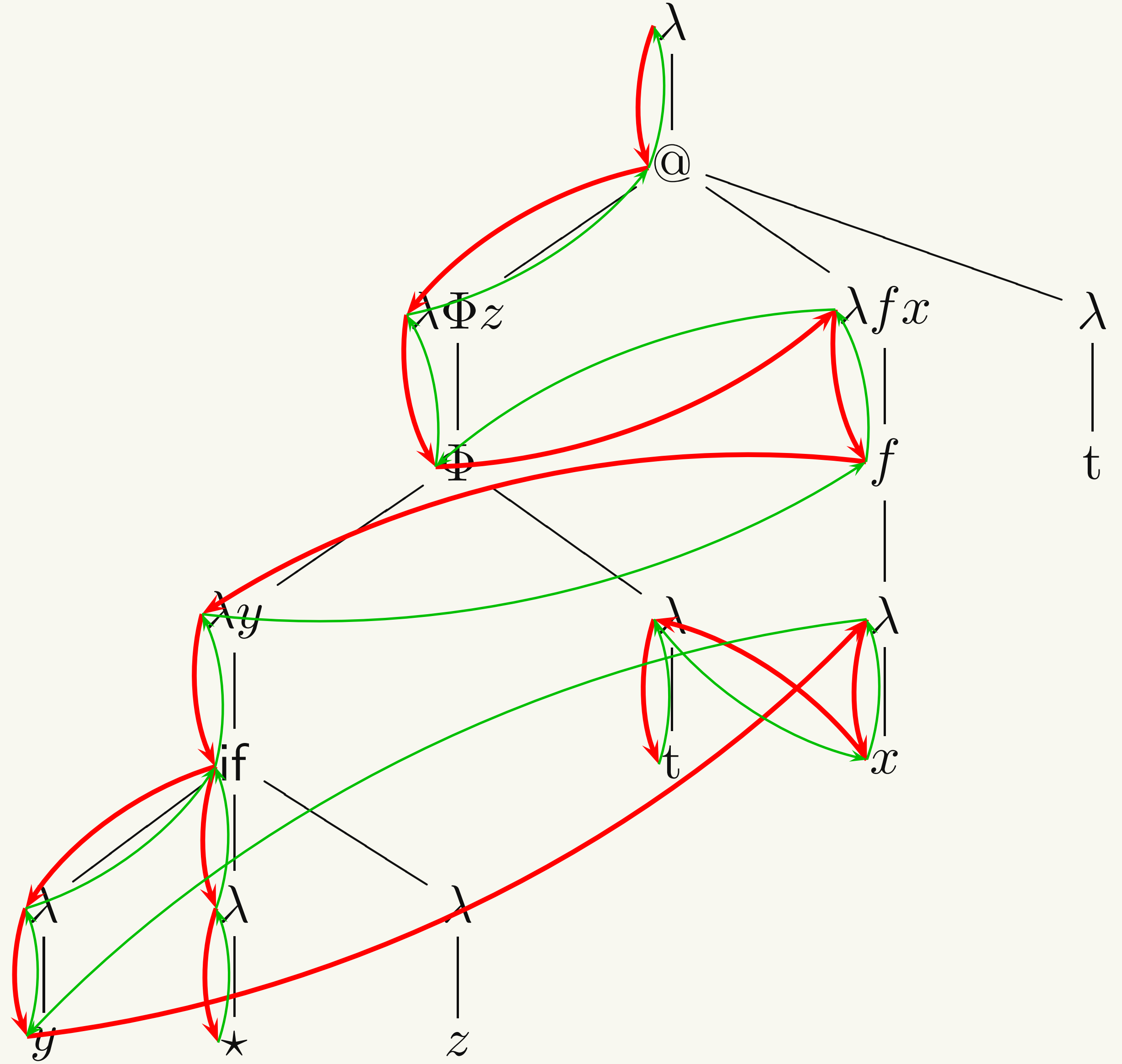
Variable profiles

Introduced by [Ong'06].

- $\mathbf{VP}_\Sigma(A_1, \ldots, A_n, o) := \mathit{Var}_\Sigma^A \times \mathit{Val} \times \mathcal{P}(\bigcup_{i=1}^n \mathbf{VP}_\Sigma(A_i))$
- Notation: (x, v), $(x, v \mid \pi_1, \ldots, \pi_n)$

Variable profiles

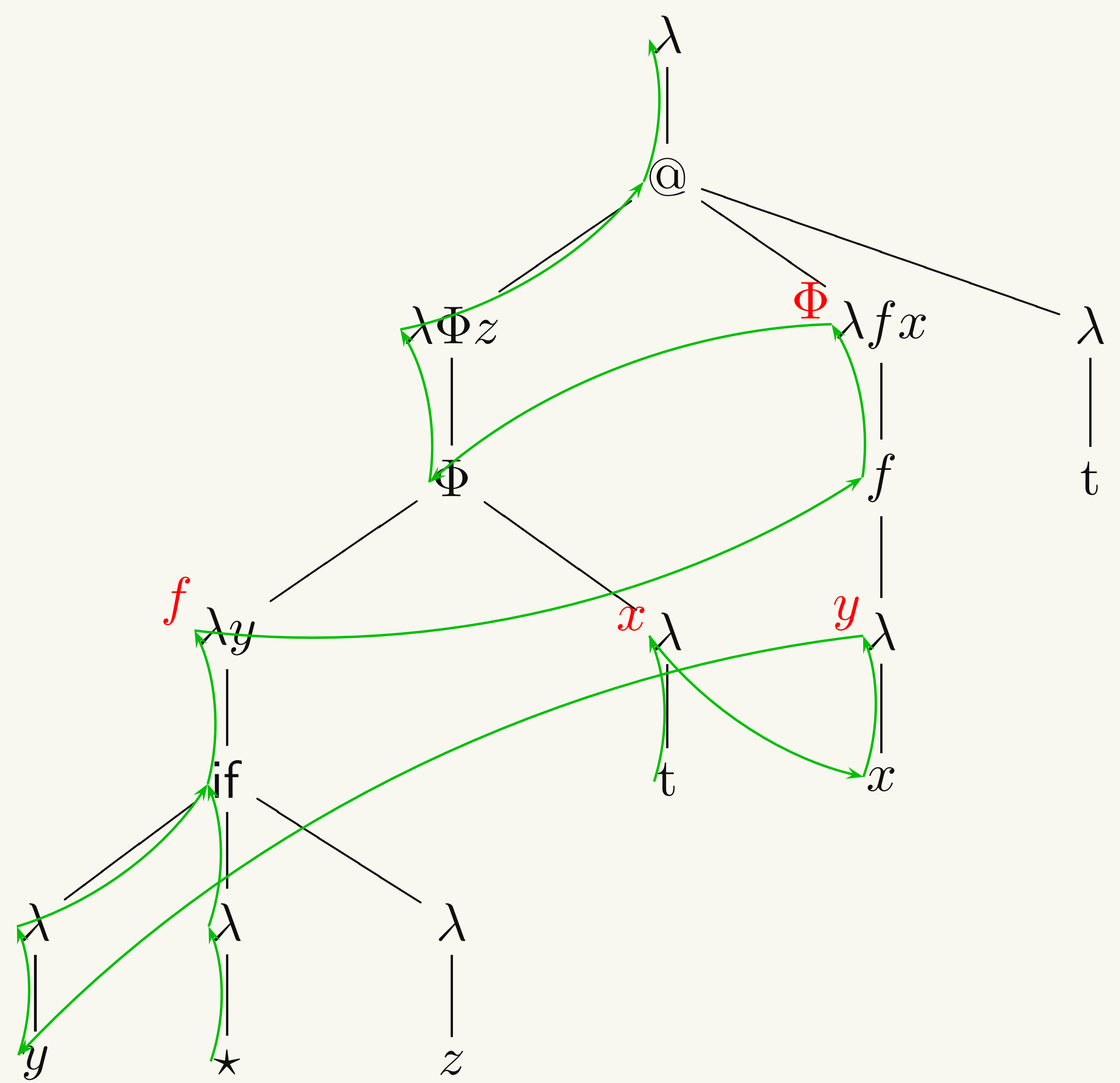
$\mathbf{VP}(A_1, \ldots, A_n, o) := \mathit{Var} \times \mathit{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \mathbf{VP}(A_i))$
Notation: (x, v), $(x, v \mid \pi_1, \ldots, \pi_n)$



Variable profiles

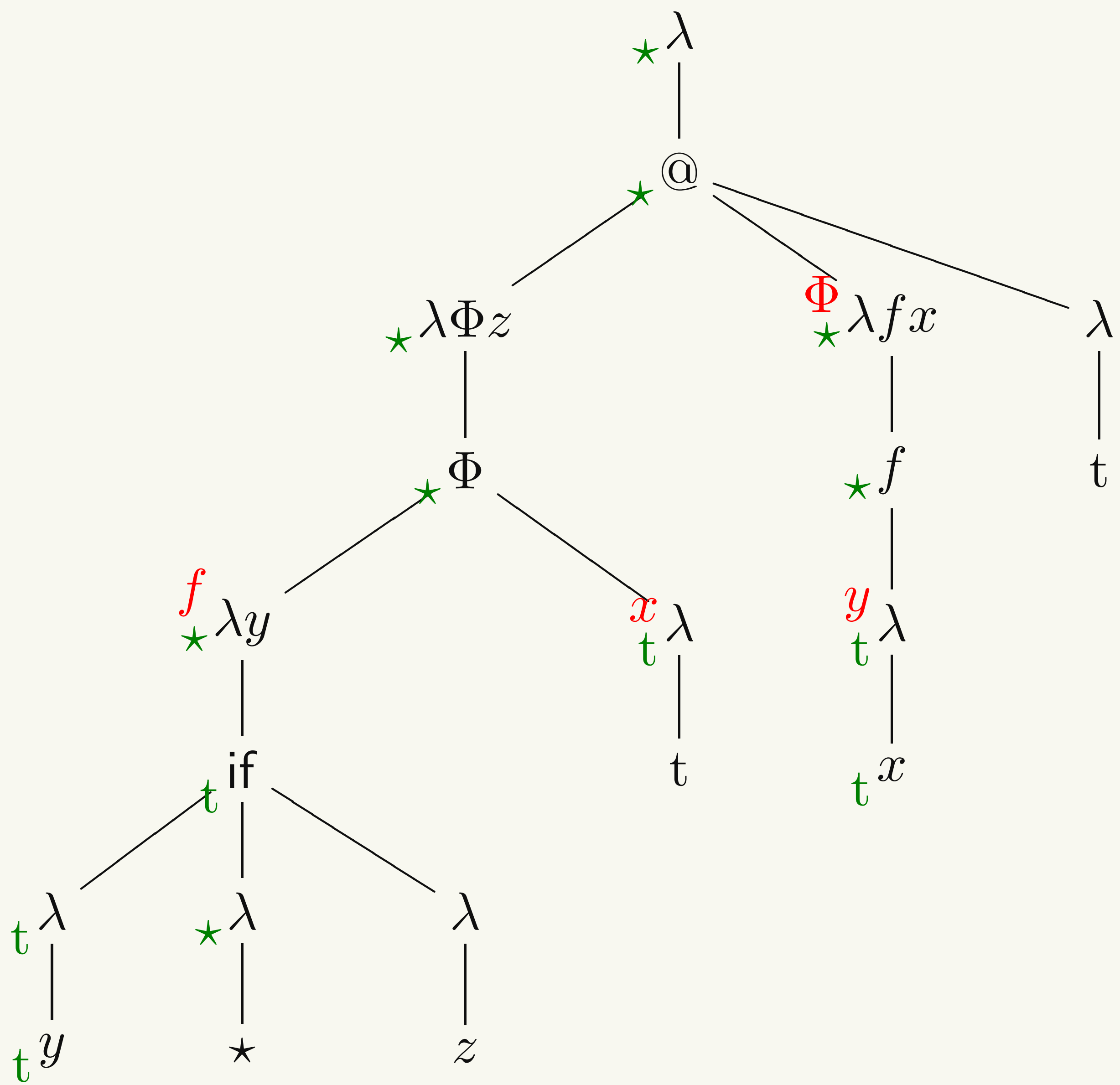
$\mathbf{VP}(A_1, \ldots, A_n, o) := \mathit{Var} \times \mathit{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \mathbf{VP}(A_i))$

Notation: (x, v), $(x, v \mid \pi_1, \ldots, \pi_n)$



Variable profiles

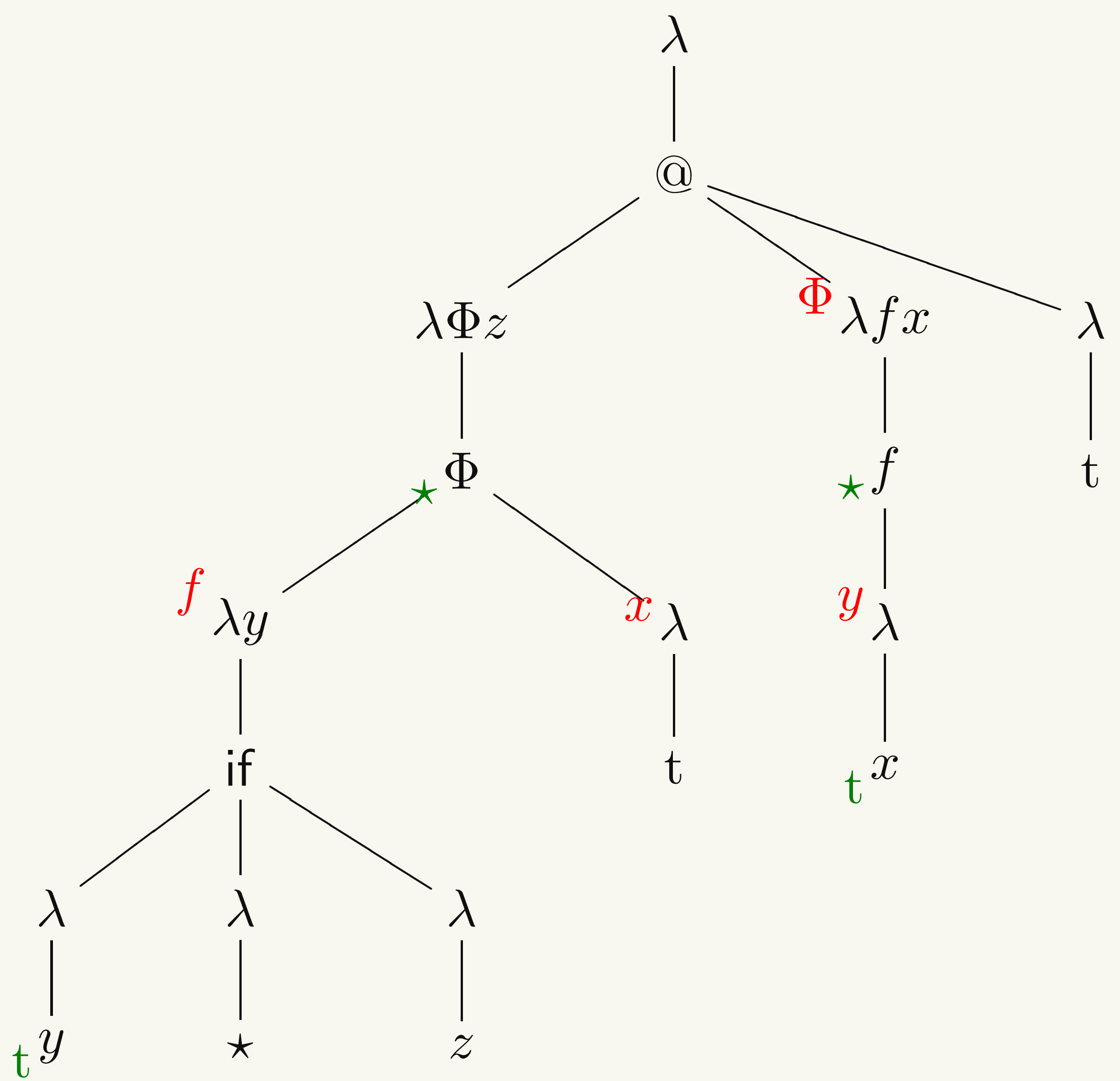
$\mathbf{VP}(A_1, \ldots, A_n, o) := \mathit{Var} \times \mathit{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \mathbf{VP}(A_i))$
Notation: (x, v), $(x, v \mid \pi_1, \ldots, \pi_n)$



Variable profiles

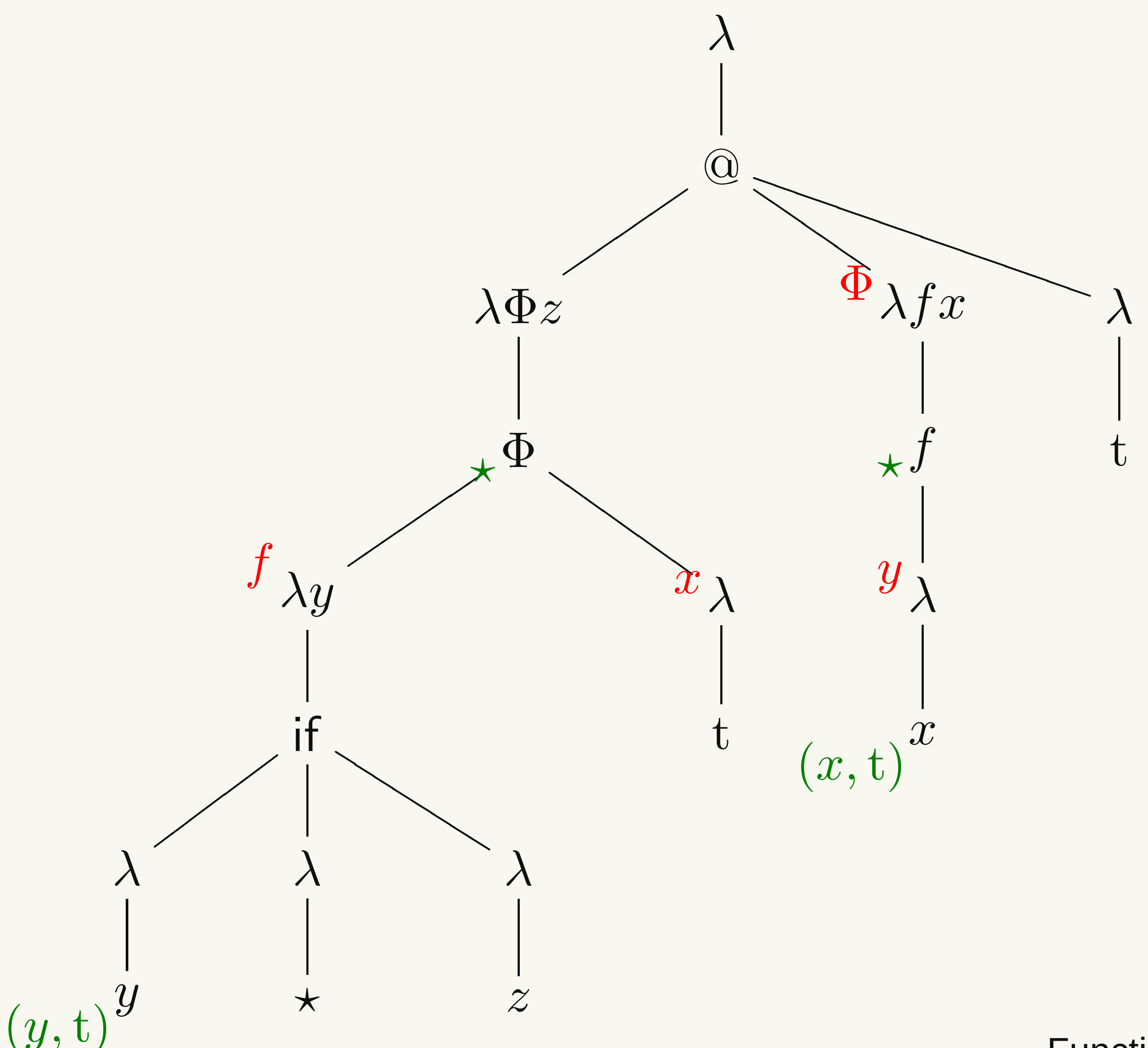
$\mathbf{VP}(A_1, \ldots, A_n, o) := \mathit{Var} \times \mathit{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \mathbf{VP}(A_i))$

Notation: (x, v), $(x, v \mid \pi_1, \ldots, \pi_n)$



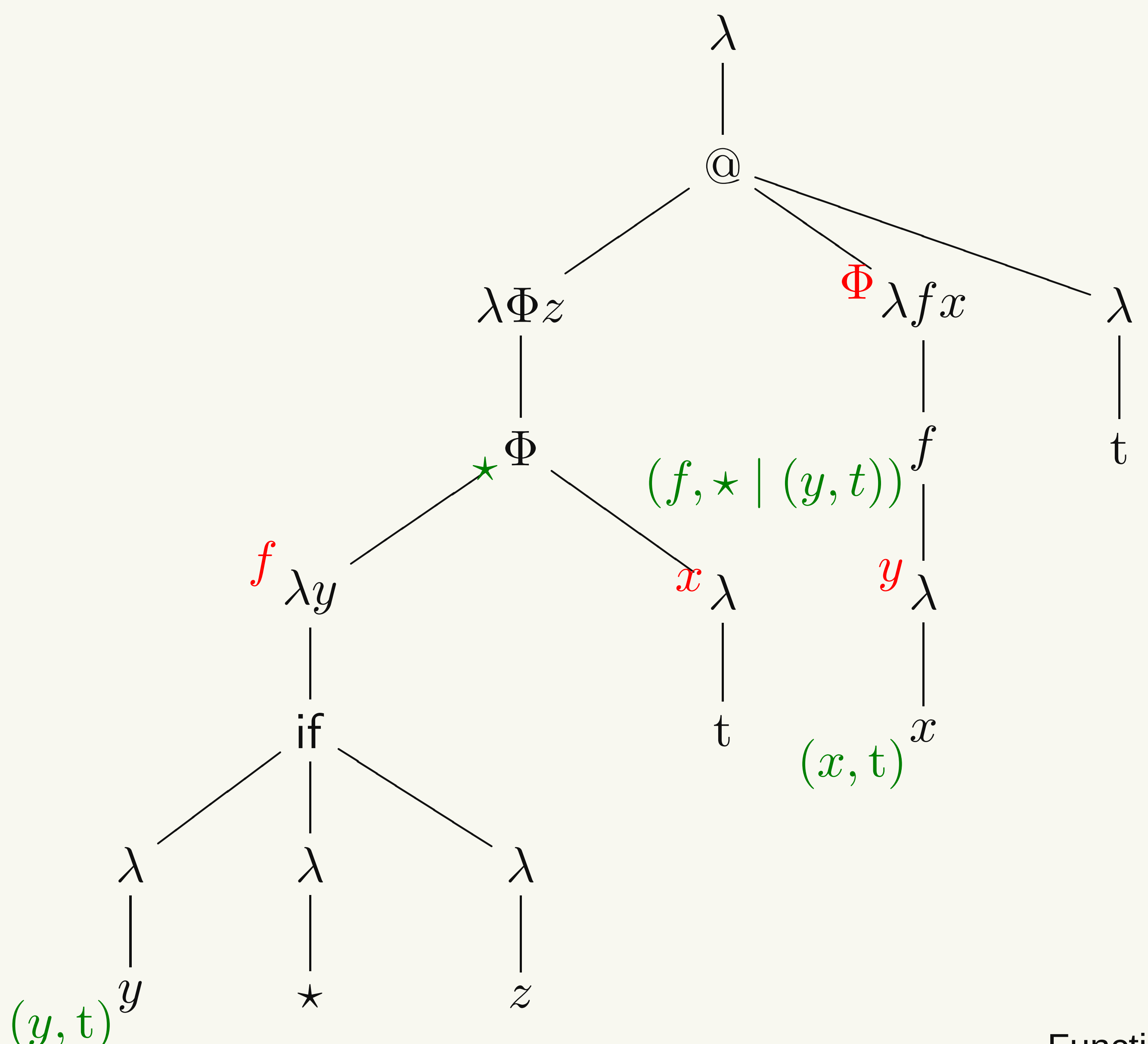
Variable profiles

$\mathbf{VP}(A_1, \ldots, A_n, o) := \mathit{Var} \times \mathit{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \mathbf{VP}(A_i))$
Notation: (x, v), $(x, v \mid \pi_1, \ldots, \pi_n)$



Variable profiles

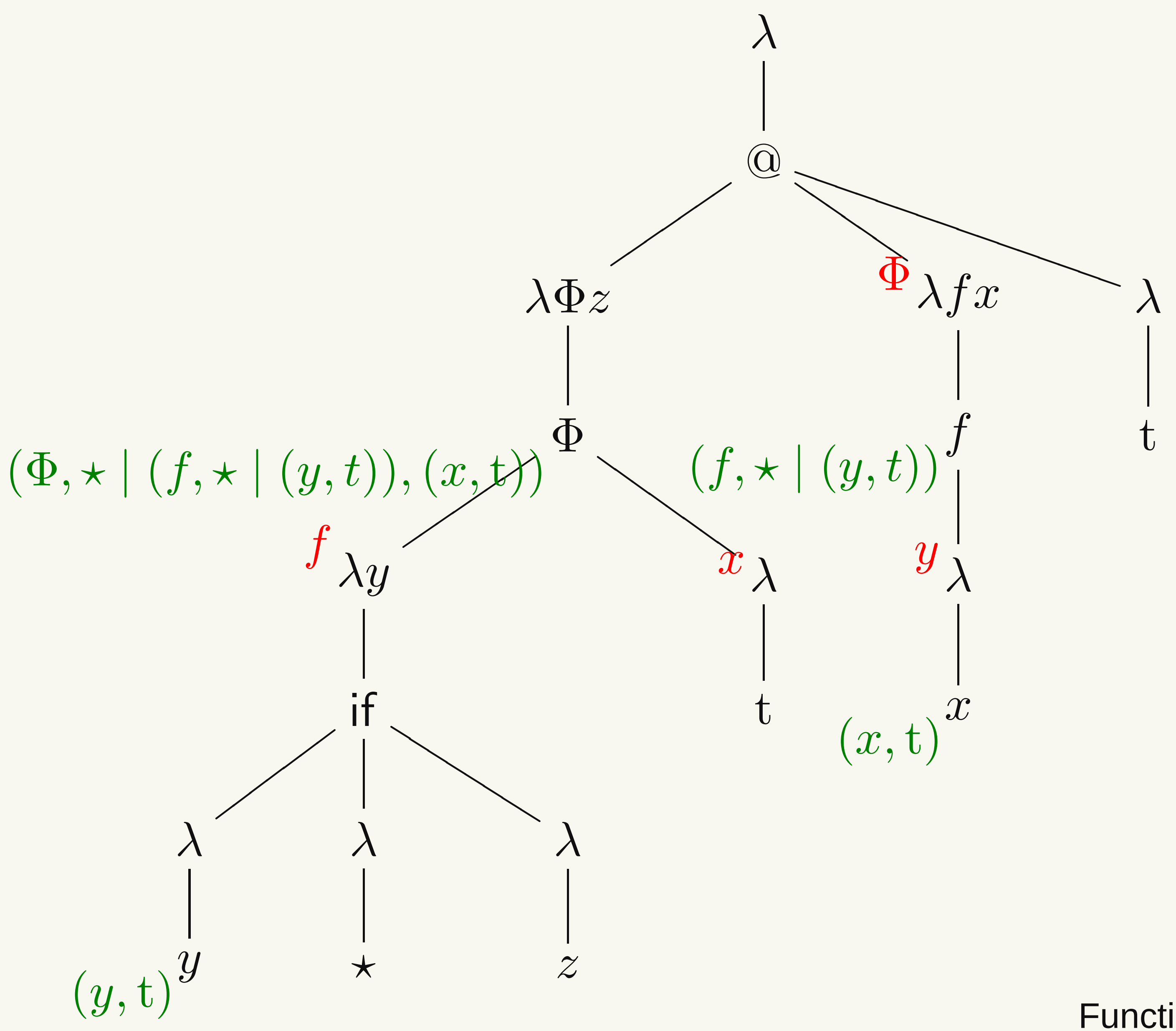
$\mathbf{VP}(A_1, \ldots, A_n, o) := \mathit{Var} \times \mathit{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \mathbf{VP}(A_i))$
Notation: (x, v), $(x, v \mid \pi_1, \ldots, \pi_n)$



Variable profiles

$\mathbf{VP}(A_1, \ldots, A_n, o) := \mathit{Var} \times \mathit{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \mathbf{VP}(A_i))$

Notation: (x, v), $(x, v \mid \pi_1, \ldots, \pi_n)$



ATA correspondence

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

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

ATA correspondence

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

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

$P \twoheadrightarrow v$ iff $\mathcal{A}_\Sigma$ accepts $\lambda(P)$ on initial state with *value* v.

Any tree accepted by $\mathcal{A}_\Sigma$ on *closed* initial state *represents* a closed $\mathrm{fPCF}^\star$-term.

Results

Theorem: $M : (A_1, \ldots, A_n, o) \in v$-$\textsc{Reach}\,[\mathrm{fPCF}^{\star}_{\Sigma}, \mathrm{fPCF}_{\Sigma}]$ iff there is a closed initial state q_0 with value v such that:

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

Results

Theorem: $M : (A_1, \ldots, A_n, o) \in v\text{-}\textsc{Reach}\,[\mathrm{fPCF}^\star_\Sigma, \mathrm{fPCF}_\Sigma]$ iff there is a closed initial state q_0 with value v such that:

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

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

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

Results

Theorem: $M : (A_1, \dots, A_n, o) \in v$-Reach $[\mathrm{fPCF}^{\star}_{\Sigma}, \mathrm{fPCF}_{\Sigma}]$ iff there is a closed initial state q_0 with value v such that:

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

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

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

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

Last slide

- 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.

Last slide

- 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: $\star$-$\textsc{Reach}\left[\mathrm{fPCF}^\star, \mathrm{fPCF}\right]$?