

Functional Reachability

Luke Ong and Nikos Tzevelekos

University of Oxford

Shonan, September 2011

What this talk is about

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 ?

What this talk is about

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 ?

What this talk is about

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.

- Reynolds ('70), Jones ('80), Shivers ('90),
many_more_people_in_this_room, . . .
Using game semantics: 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.

- Reynolds ('70), Jones ('80), Shivers ('90), many_more_people_in_this_room, . . .
Using game semantics: 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: (binary) PCF.

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

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

$$v ::= t \mid f$$

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

$$A, B ::= o \mid A \rightarrow B$$
$$v ::= t \mid f$$
$$M, N ::= v \mid x \mid \lambda x.M \mid MN \mid \text{if } M N_1 N_2 \mid \mathbf{Y}_A$$

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

$$v ::= t \mid f$$

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

$$\begin{array}{ll} (\lambda x.M)N \rightarrow M\{N/x\} & \text{if } t \rightarrow \lambda xy.x \\ \mathbf{Y}M \rightarrow M(\mathbf{Y}M) & \text{if } f \rightarrow \lambda xy.y \end{array}$$

$$M \rightarrow N \implies \mathbf{E}[M] \rightarrow \mathbf{E}[N]$$

$$\mathbf{E} ::= [-] \mid \mathbf{E} M \mid \text{if } \mathbf{E}$$

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

$$v ::= t \mid f$$

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

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

- Write (A_1, \dots, A_n, o) for $A_1 \rightarrow \dots \rightarrow A_n \rightarrow o$.
- Divergence definable, e.g. $\perp := \mathbf{Y}_o(\lambda x.x)$.

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

$$v ::= t \mid f$$

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

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

- Write (A_1, \dots, A_n, o) for $A_1 \rightarrow \dots \rightarrow A_n \rightarrow o$.
- Divergence definable, e.g. $\perp := \mathbf{Y}_o(\lambda x.x)$.
- *Finitary* restrictions (i.e. no \mathbf{Y}):

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

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

Reachability

Using a *Context Lemma* we can restate Reachability as:

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

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

Reachability

Using a *Context Lemma* we can restate Reachability as:

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

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

We can make things simpler

PCF-with-error: PCF^{*}

- Include an error constant: $o = \{t, f, \star\}$
- New rule: $\text{if } \star \rightarrow \lambda xy.\star$ (i.e. $E[\star] \twoheadrightarrow \star$)

PCF-with-error: PCF^{*}

- Include an error constant: $o = \{t, f, \star\}$
- New rule: $\text{if } \star \rightarrow \lambda xy.\star$ (i.e. $E[\star] \rightarrow \star$)

★-Reachability:

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

PCF-with-error: PCF[★]

- Include an error constant: $o = \{t, f, \star\}$
- New rule: $\text{if } \star \rightarrow \lambda xy.\star$ (i.e. $E[\star] \twoheadrightarrow \star$)

★-Reachability:

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

Reachability \cong ★-Reachability

Reach template

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

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

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

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

Reach template

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

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

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

E.g. $\left\{ \begin{array}{l} \star\text{-Reachability} = \star\text{-REACH} [\text{PCF}^{1\star}, \text{PCF}] \\ v\text{-REACH} [\mathcal{L}, \text{PCF}] = v\text{-REACH} [\mathcal{L}, \text{fPCF}] \end{array} \right.$

Reach template

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

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

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

E.g. $\left\{ \begin{array}{l} \star\text{-Reachability} = \star\text{-REACH} [\text{PCF}^{1\star}, \text{PCF}] \\ v\text{-REACH} [\mathcal{L}, \text{PCF}] = v\text{-REACH} [\mathcal{L}, \text{fPCF}] \end{array} \right.$

From [Loader]:

Observational equivalence in fPCF_\perp is undecidable.

therefore:

$t\text{-REACH} [\text{fPCF}_\perp, \text{fPCF}]$ is undecidable.

Undecidability

The following problems are undecidable.

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

Undecidability

The following problems are undecidable.

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

Not all is lost

- Reachability for *finitary* M ?
- \star -REACH [fPCF 1* , fPCF] ?
- \star -REACH [fPCF * , fPCF] ?

Our approach

We focus on $v\text{-REACH}$ [fPCF*, fPCF]

Computations of fPCF*-term $P : o$



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



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

Our approach

We focus on v -REACH [fPCF^{*}, fPCF]

Computations of fPCF^{*}-term $P : o$



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



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

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

Computation trees

Starting from a fPCF^{*}-term M ,

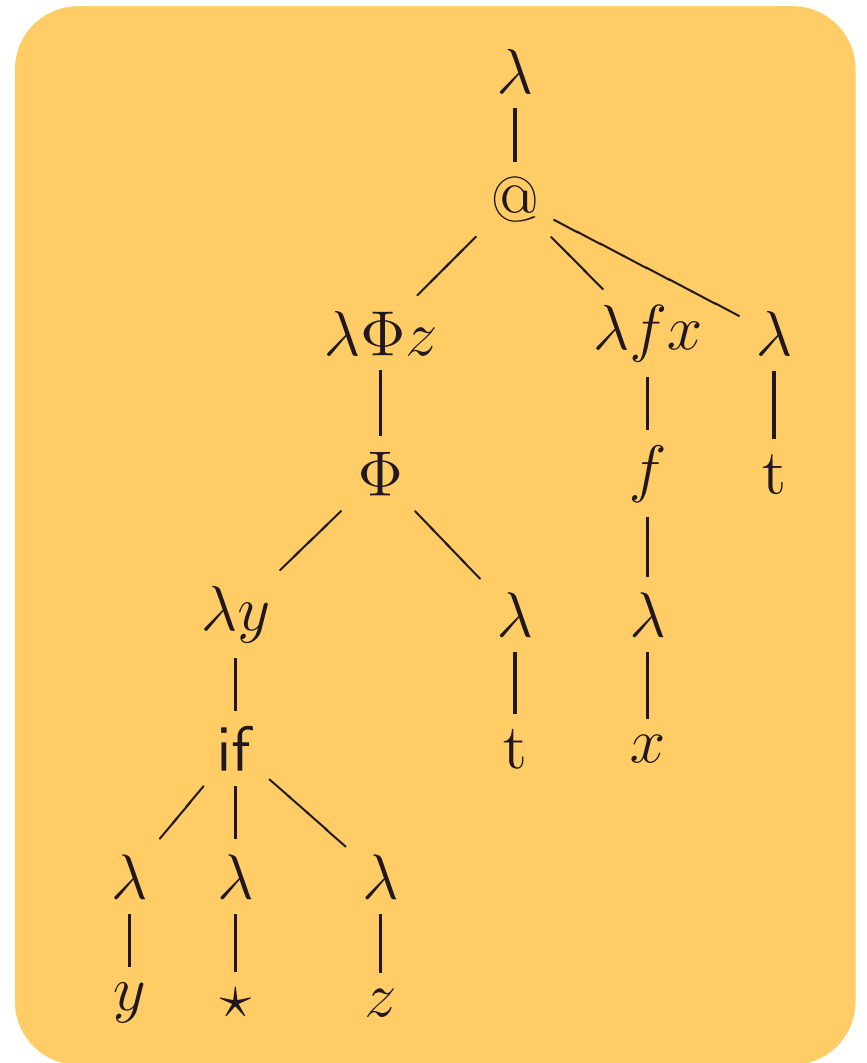
- take its η -long form,
- add application symbols ($@$),
- view the result as a tree, $\lambda(M)$.

Computation trees

Starting from a fPCF^{*}-term M ,

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

$(\lambda\Phi z. \Phi(\lambda y. \text{if } y \star z)t) (\lambda f x. f x) t \quad \mapsto$



Traversals

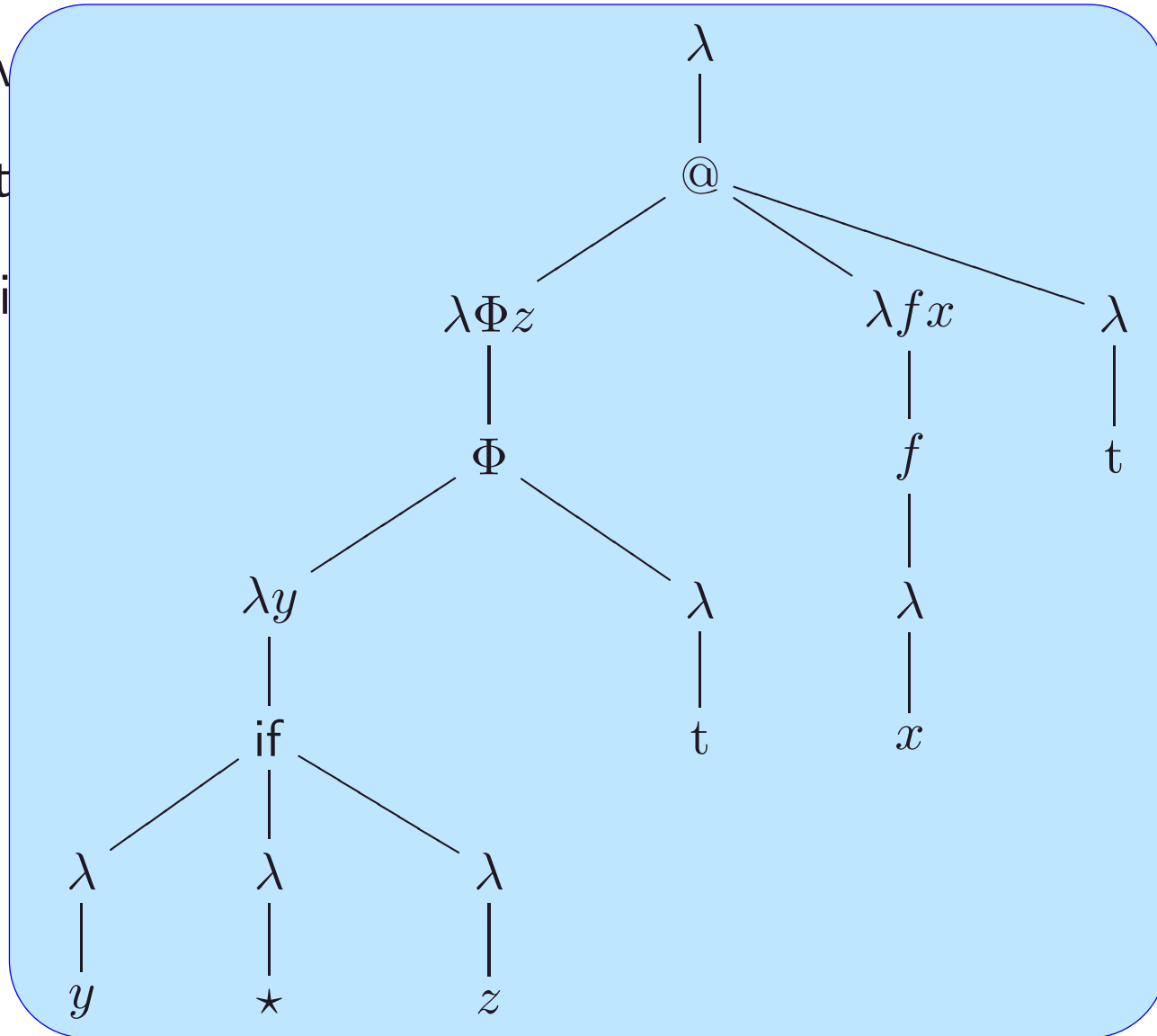
A **traversal** [Blum, Ong] over a *full* computation tree:

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

Traversals

A **traversal** [Blum, Ong] over

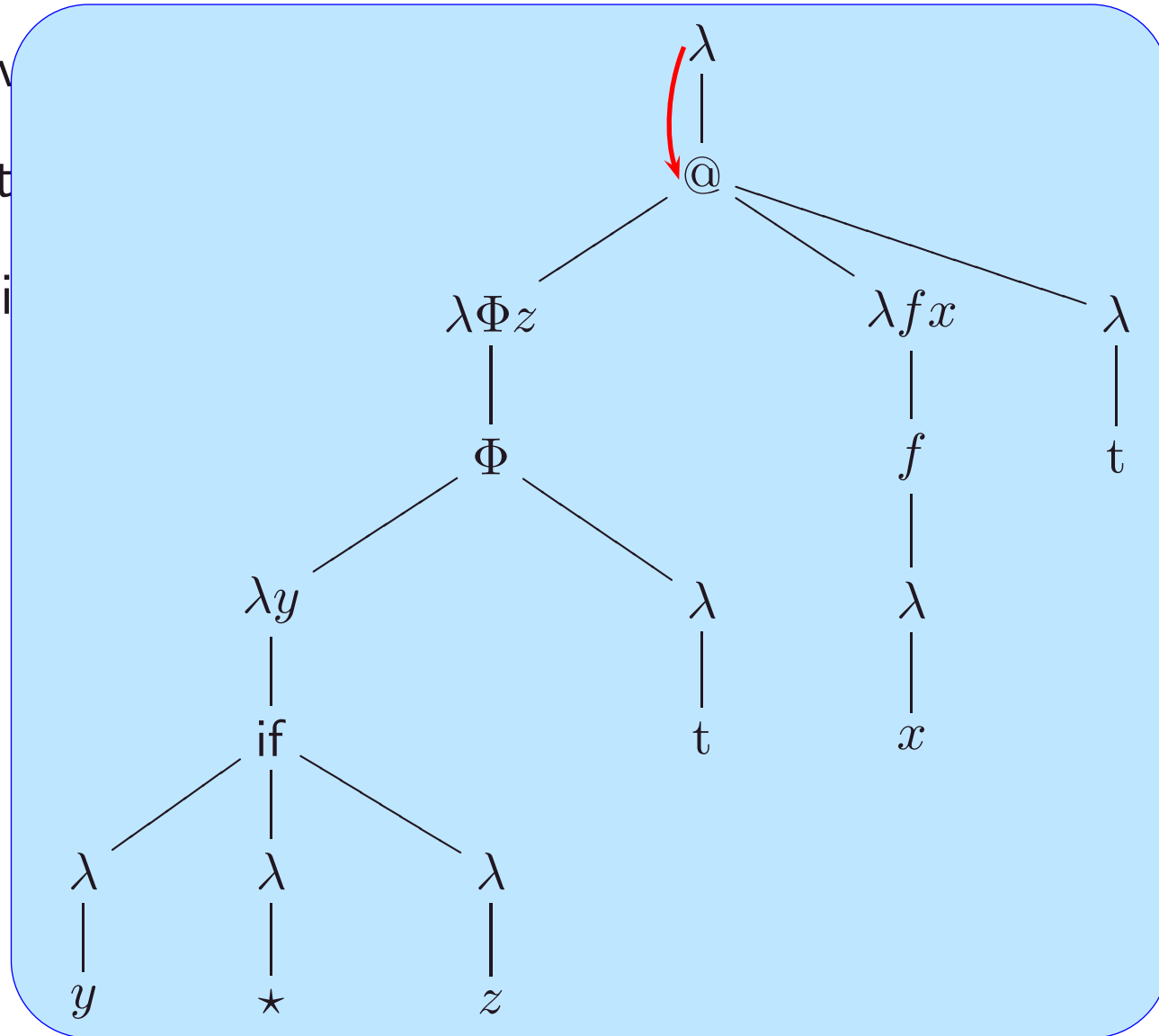
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

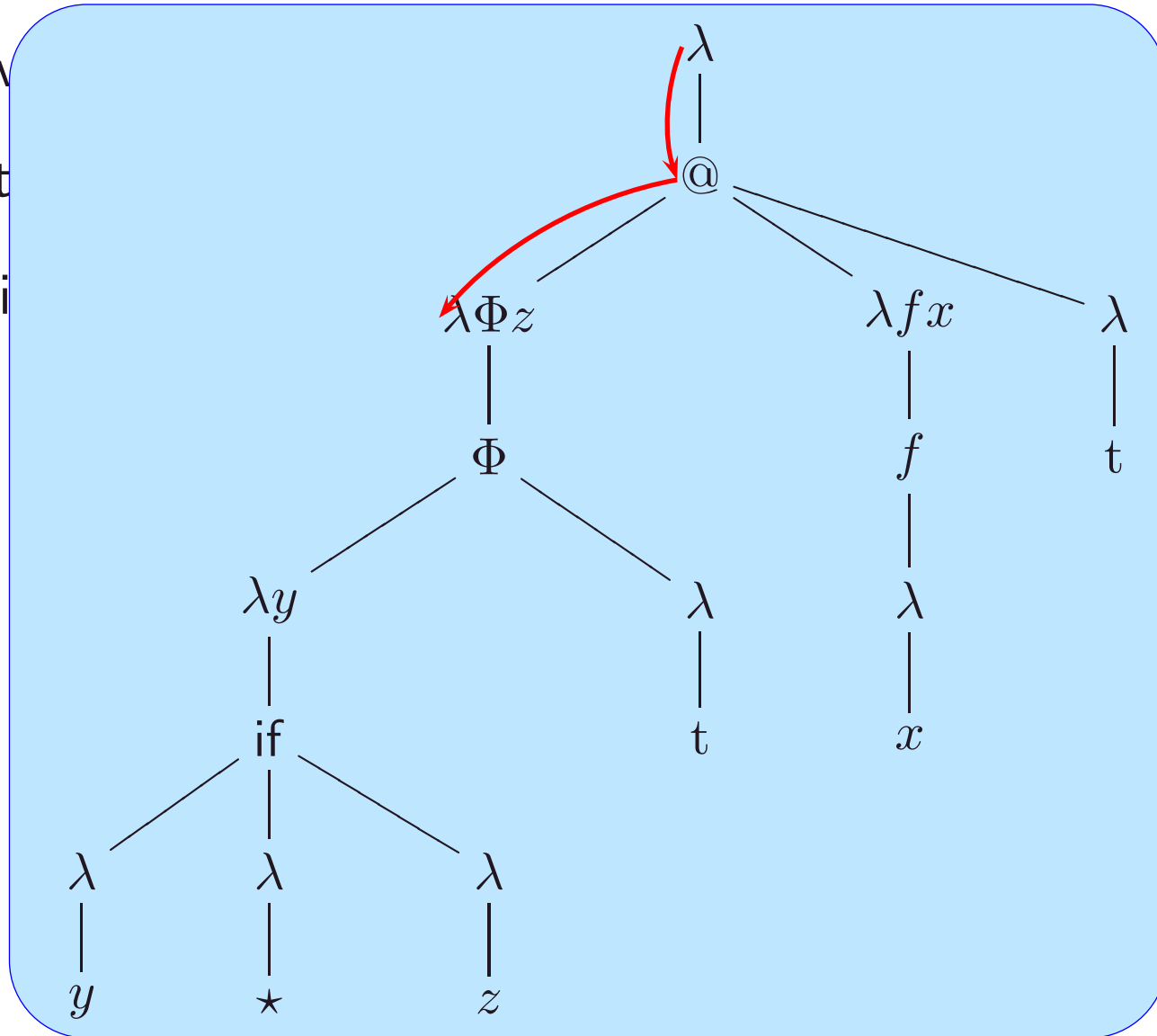
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

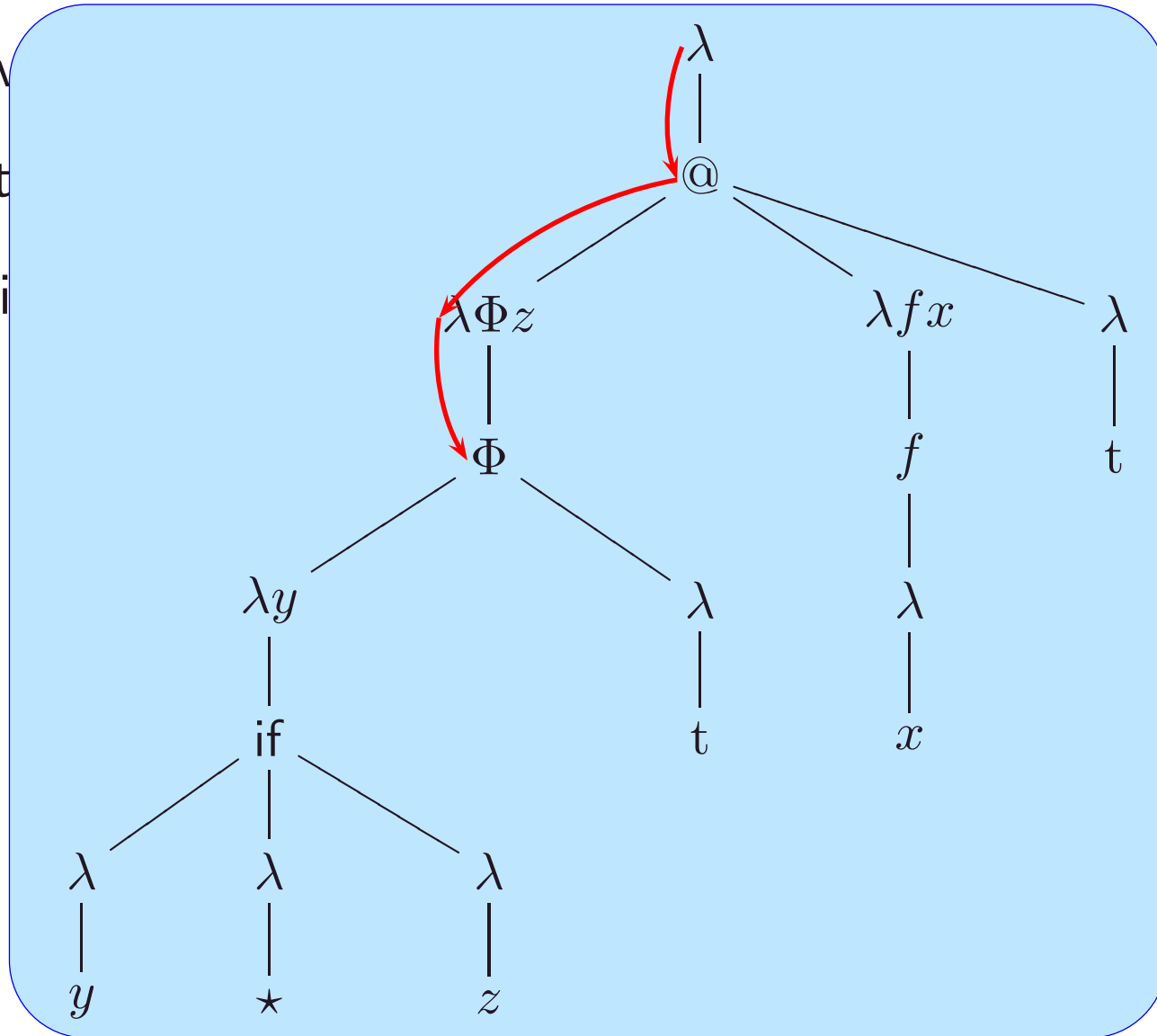
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

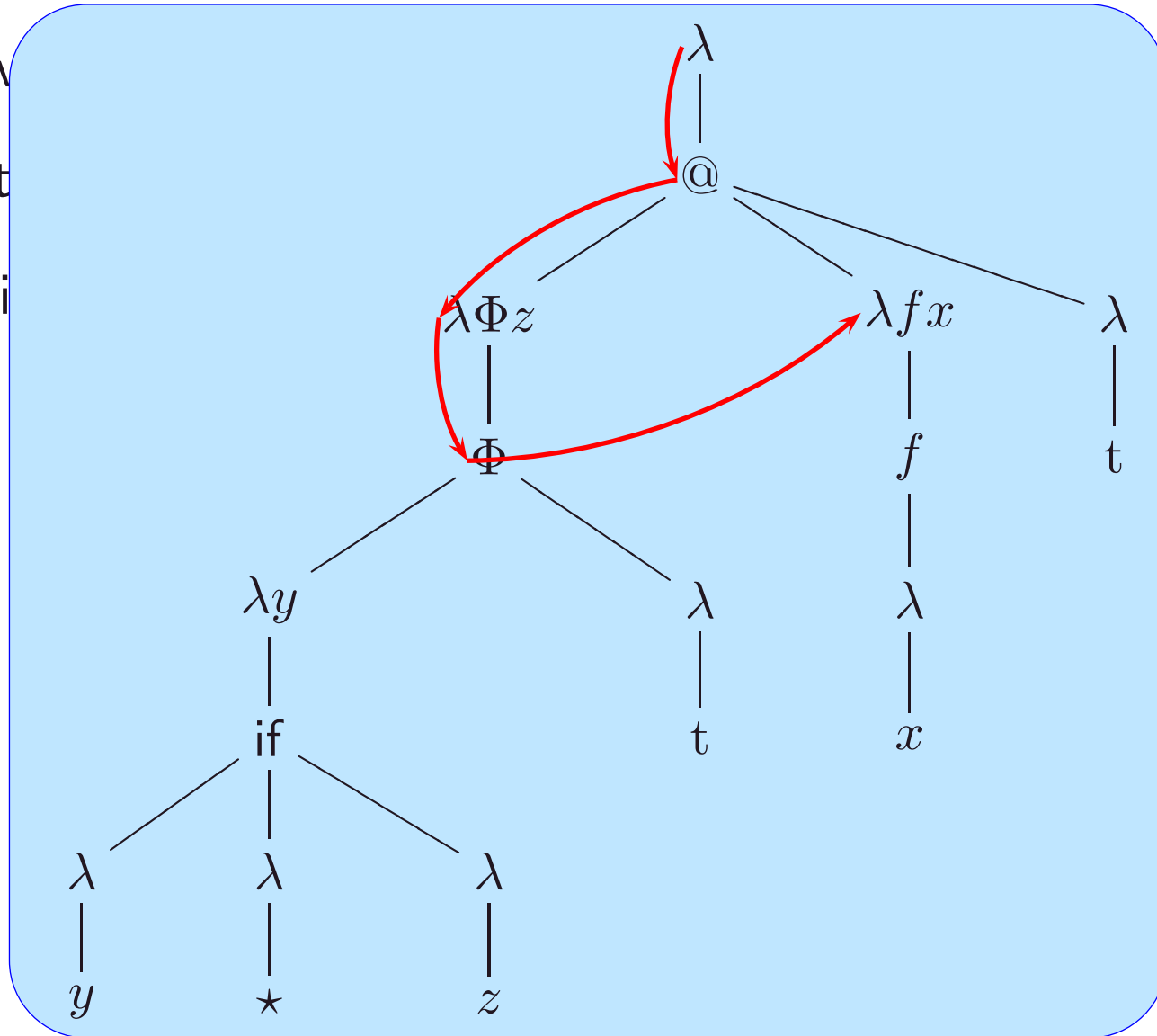
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

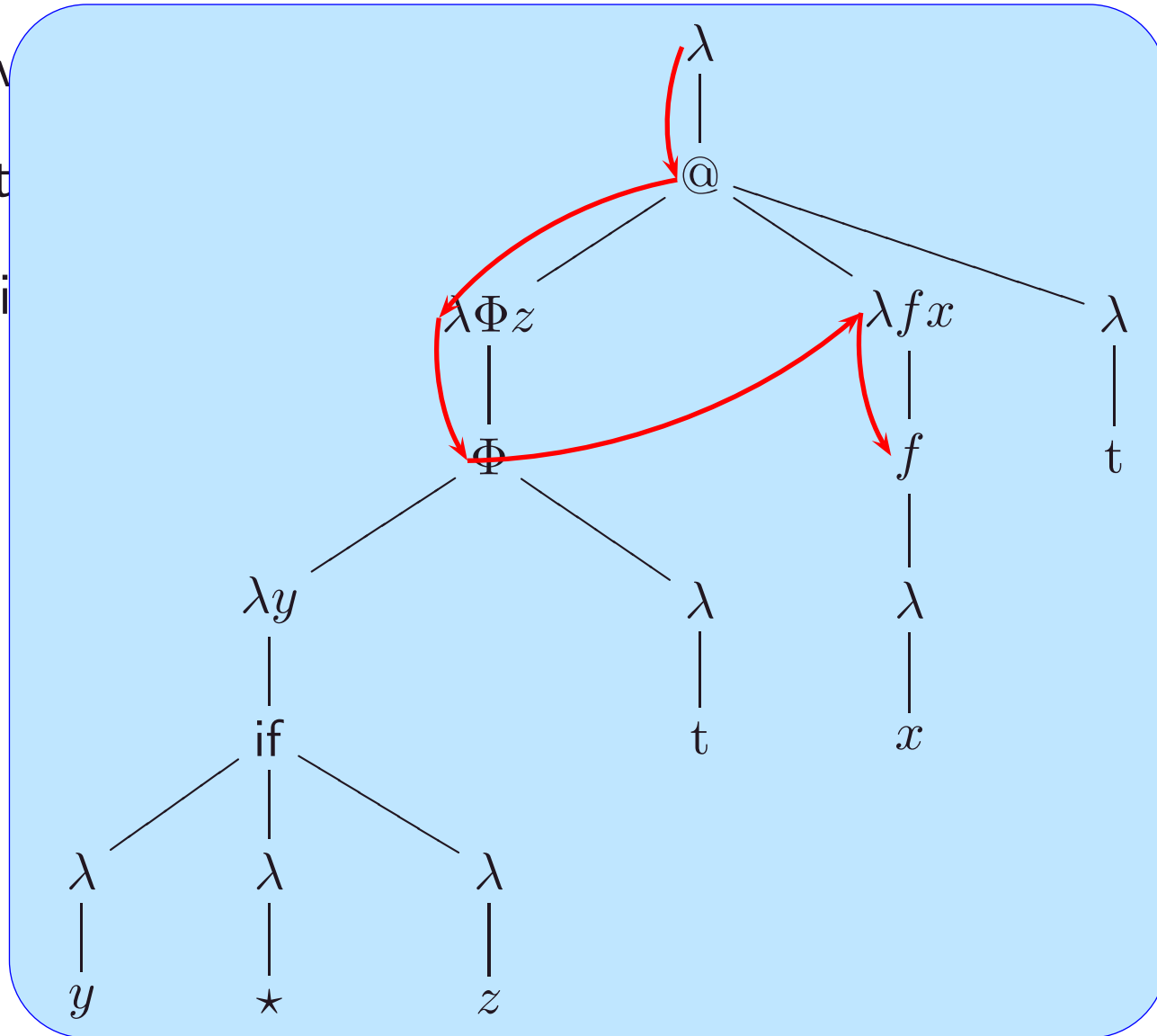
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

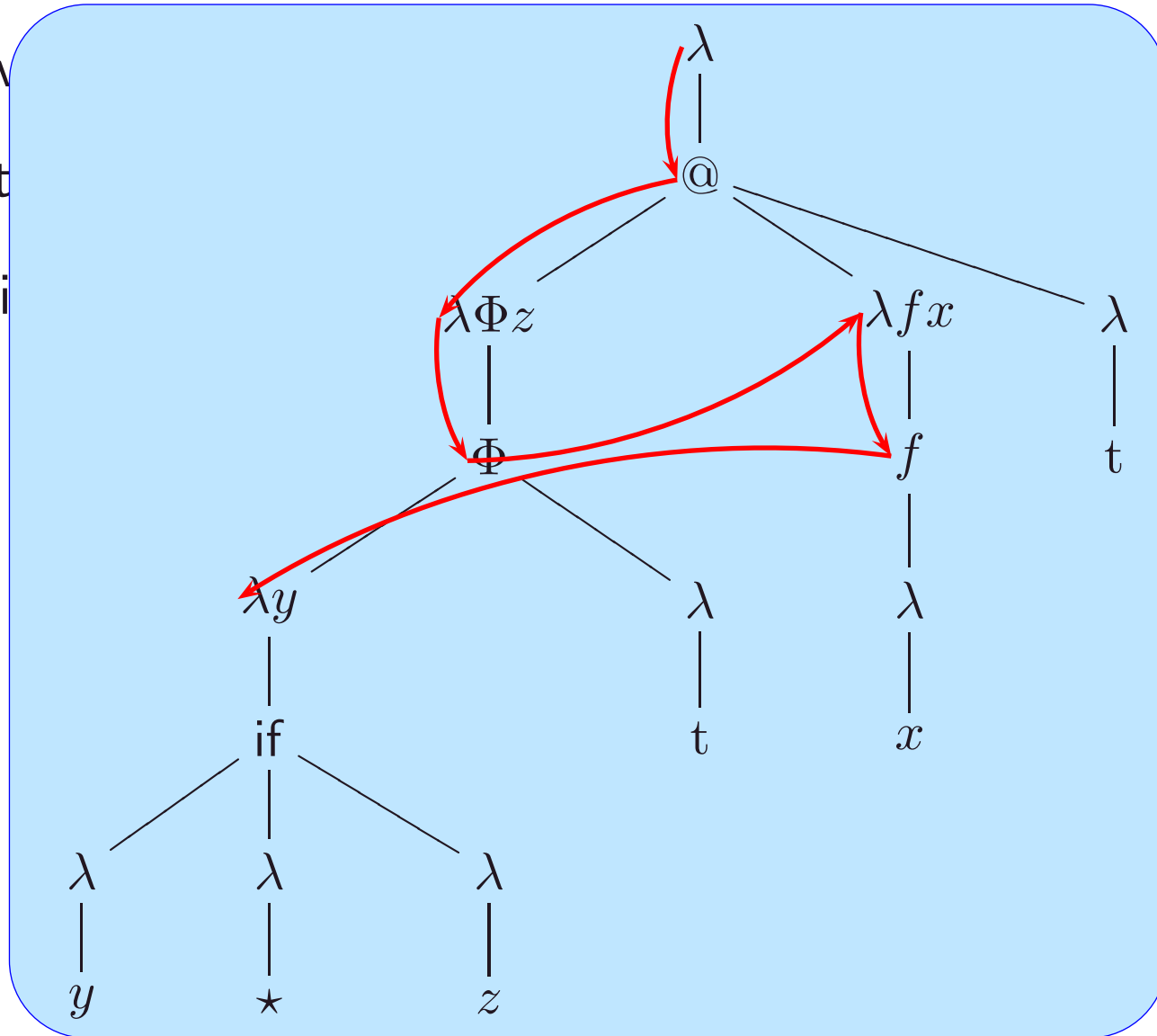
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

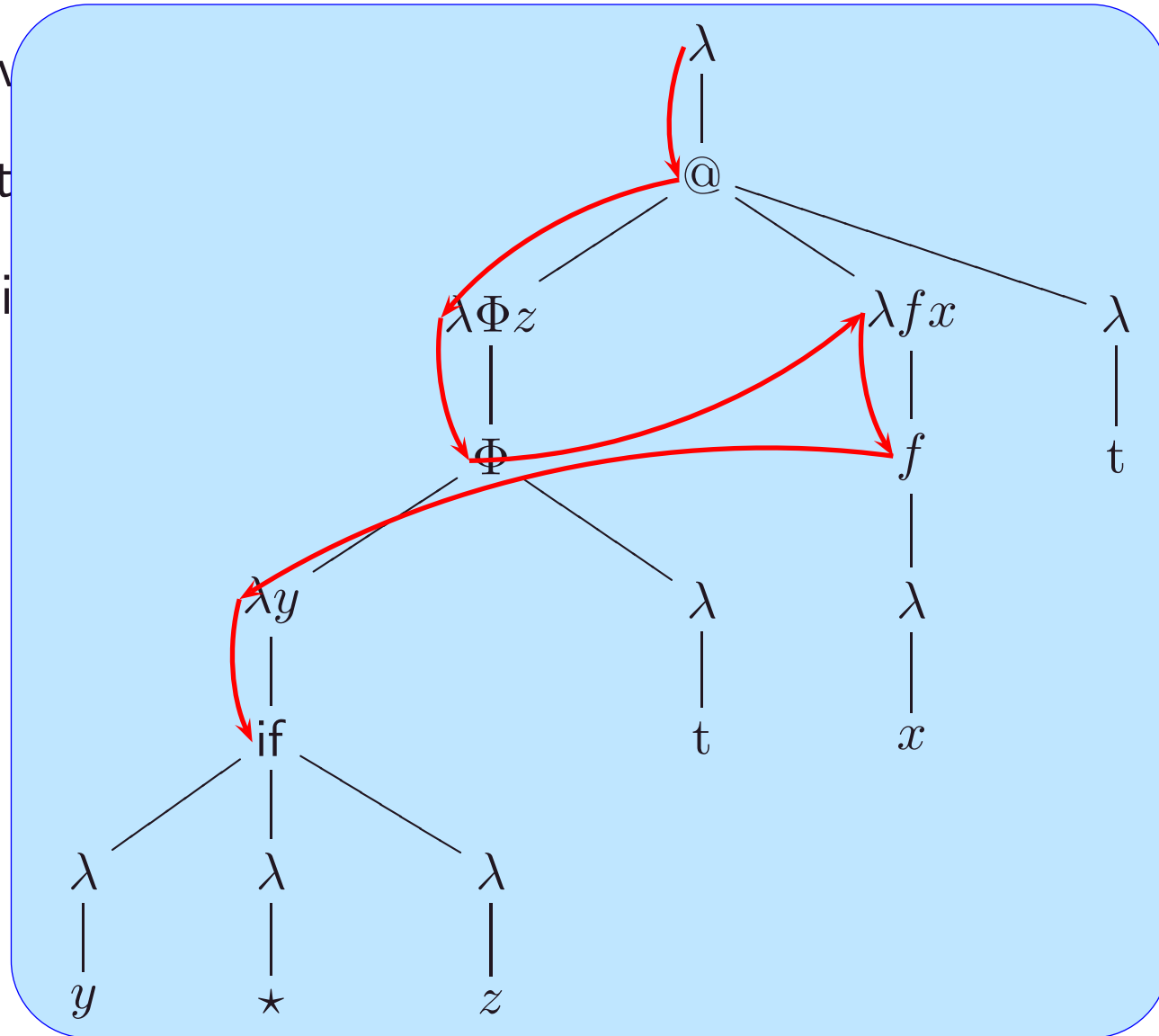
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

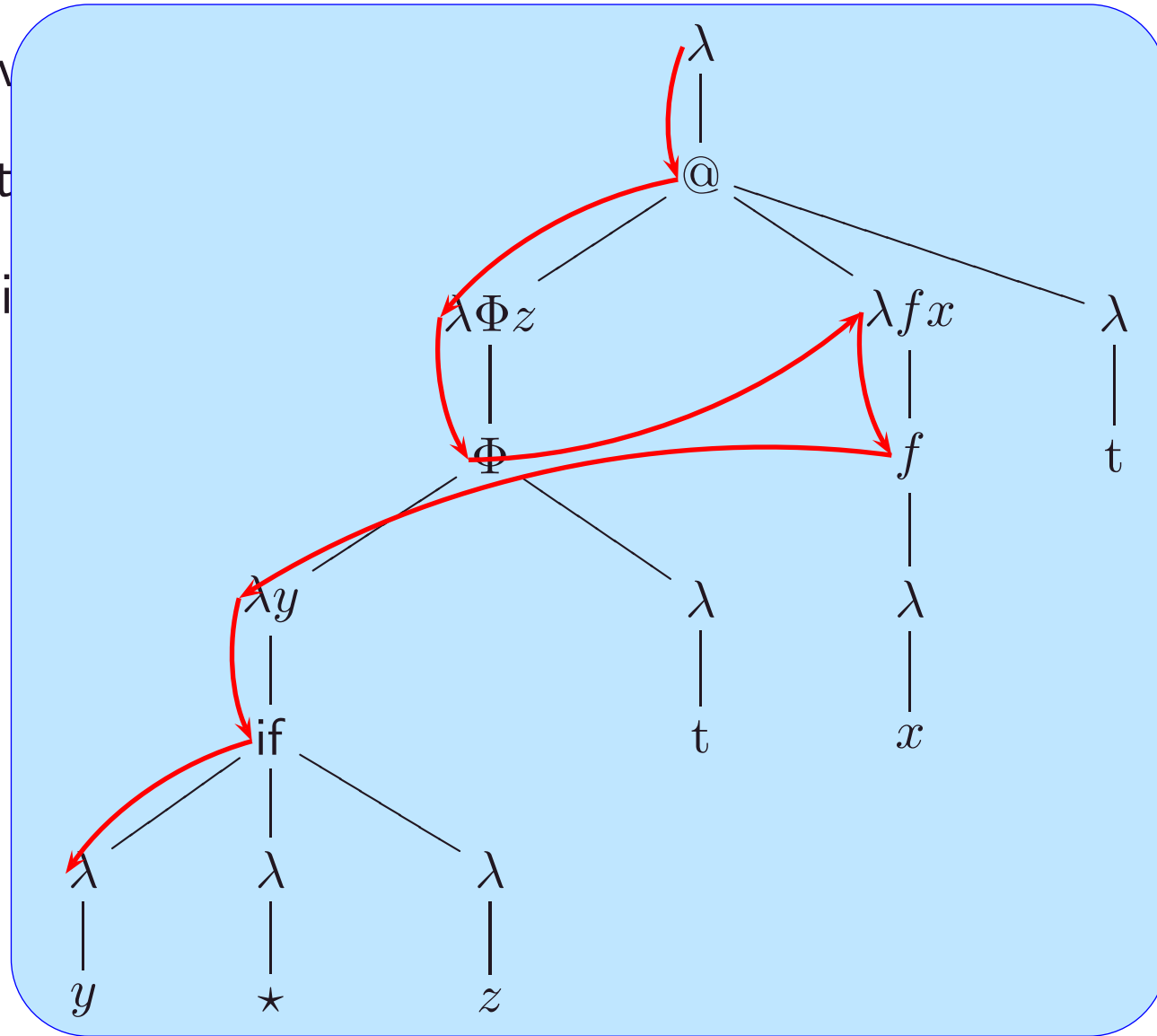
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

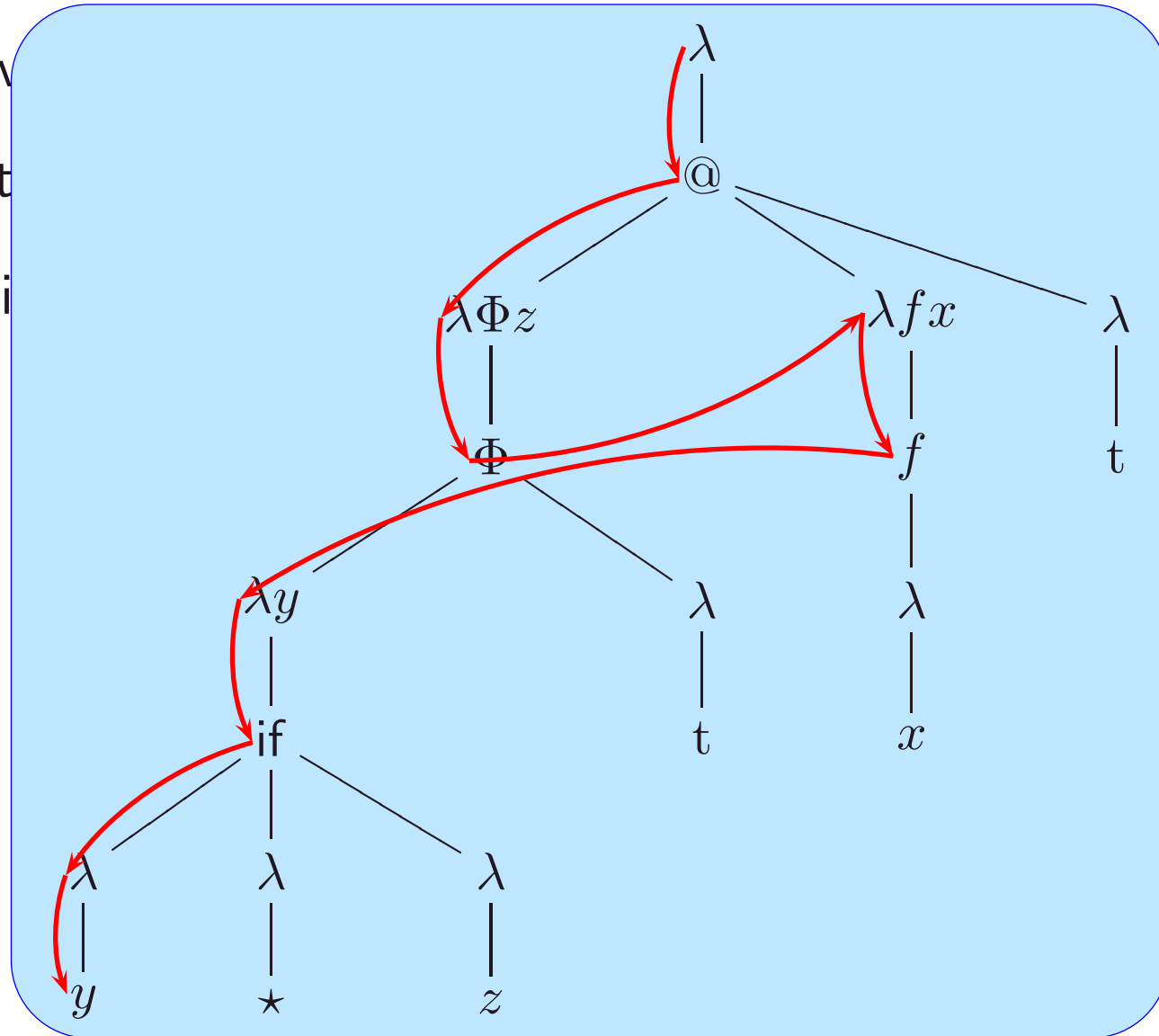
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

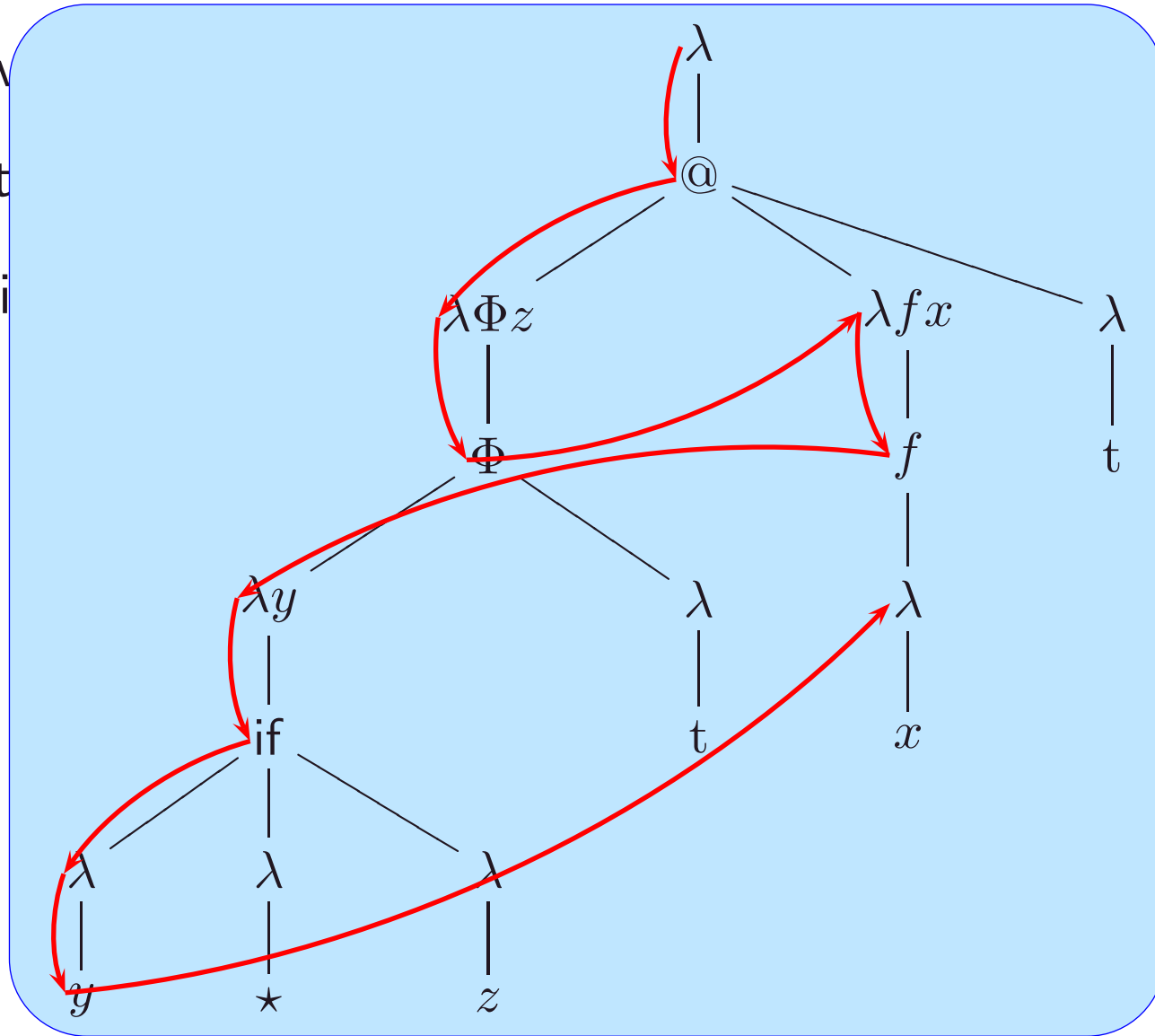
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

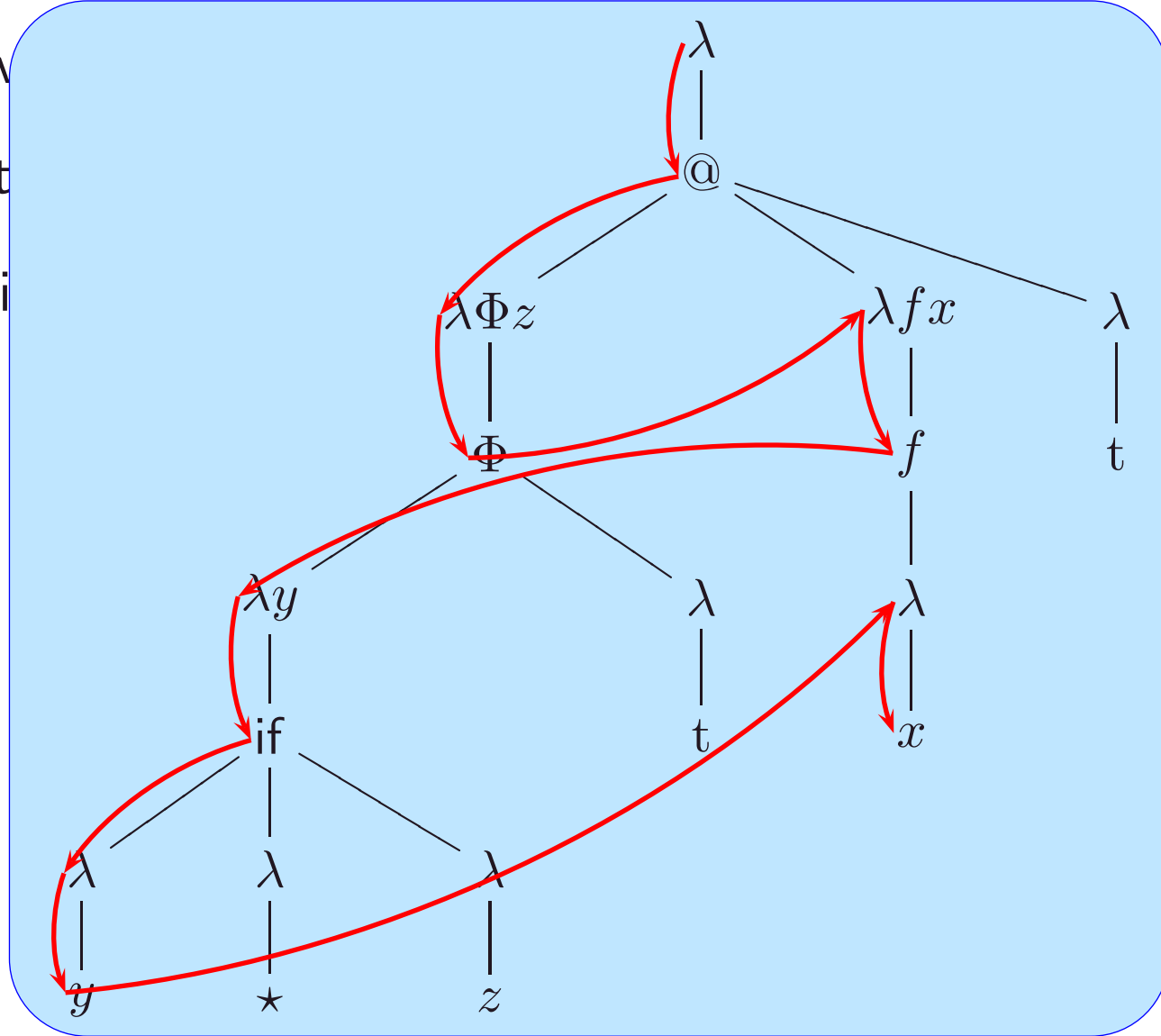
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

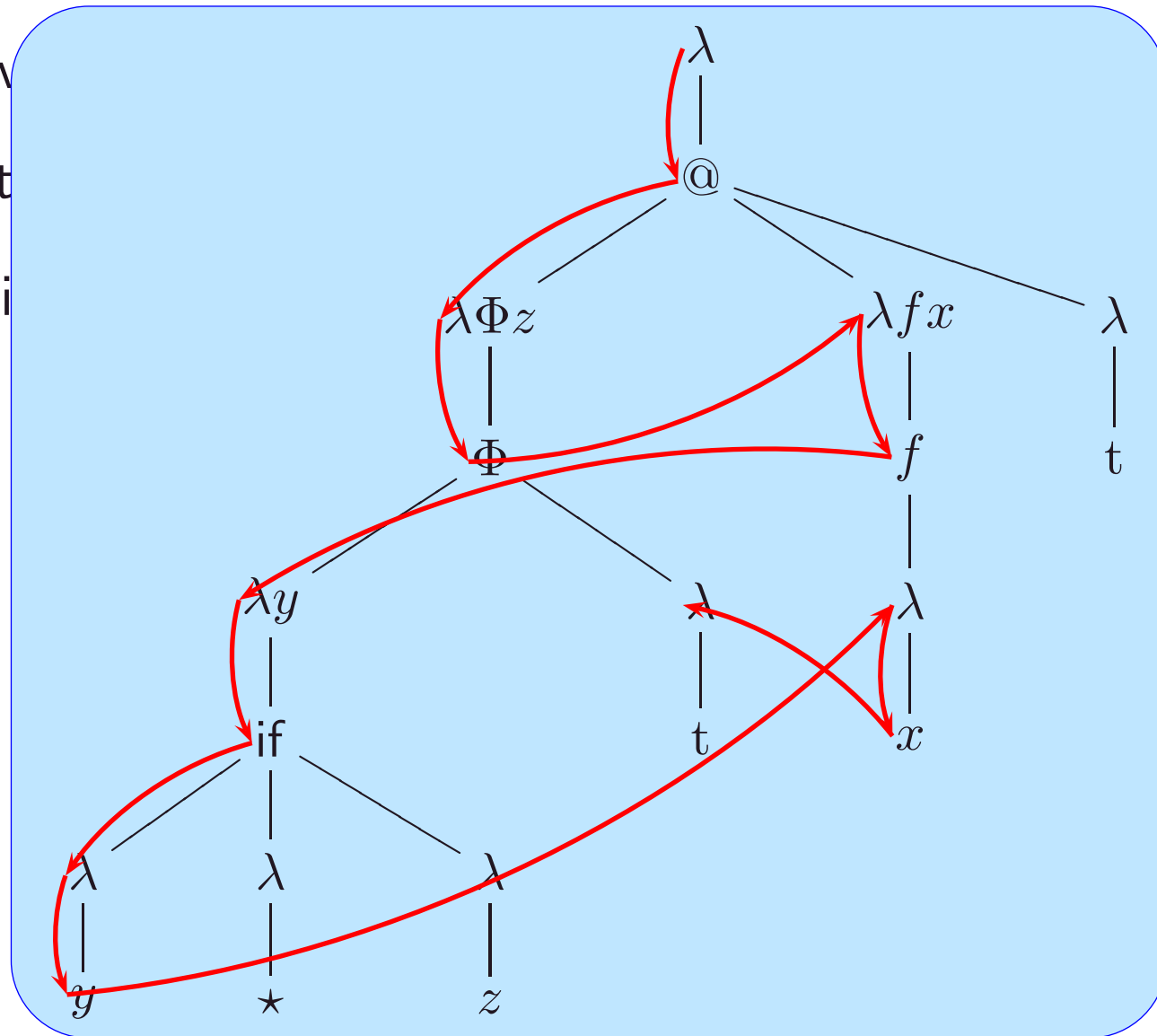
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

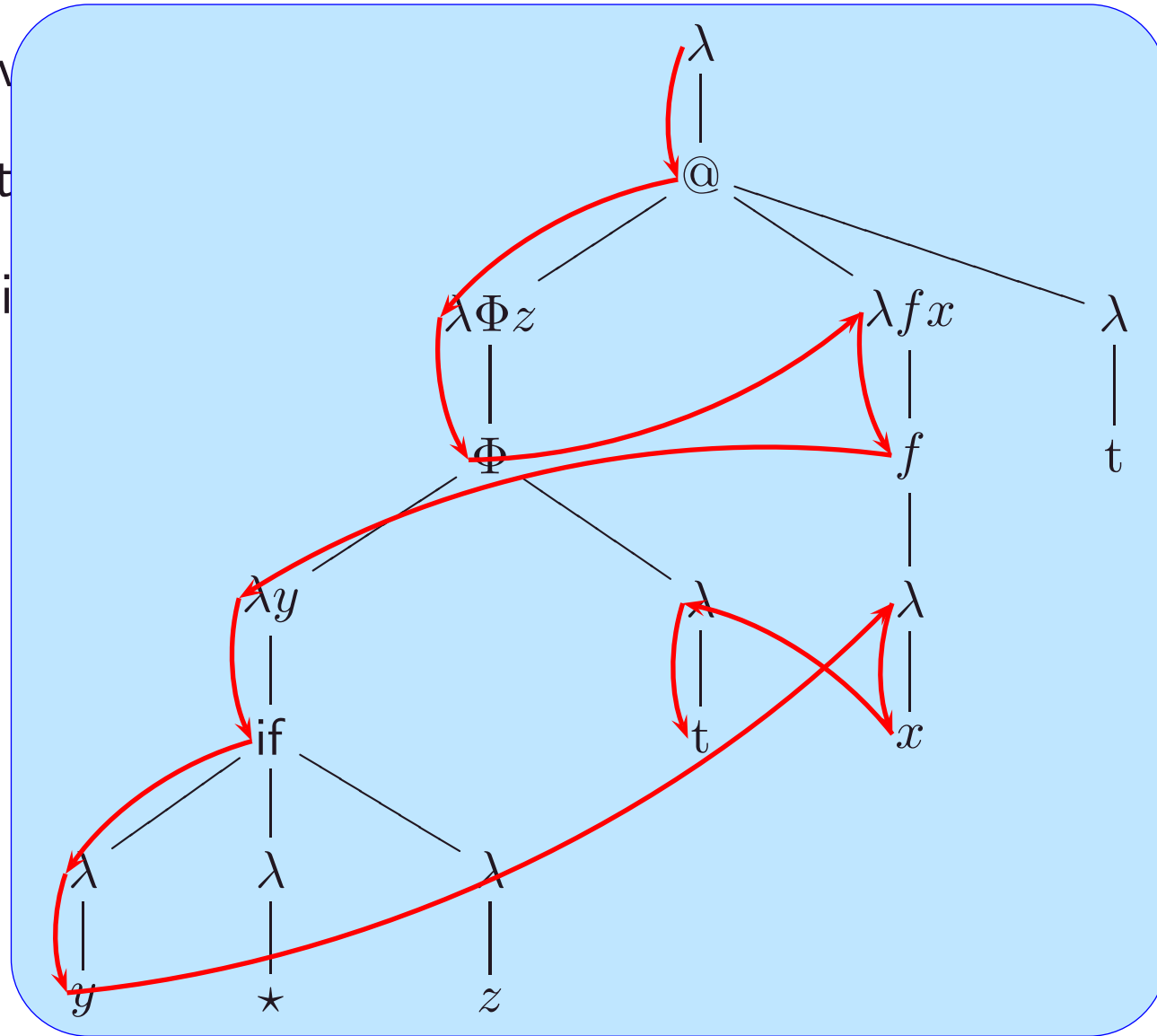
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

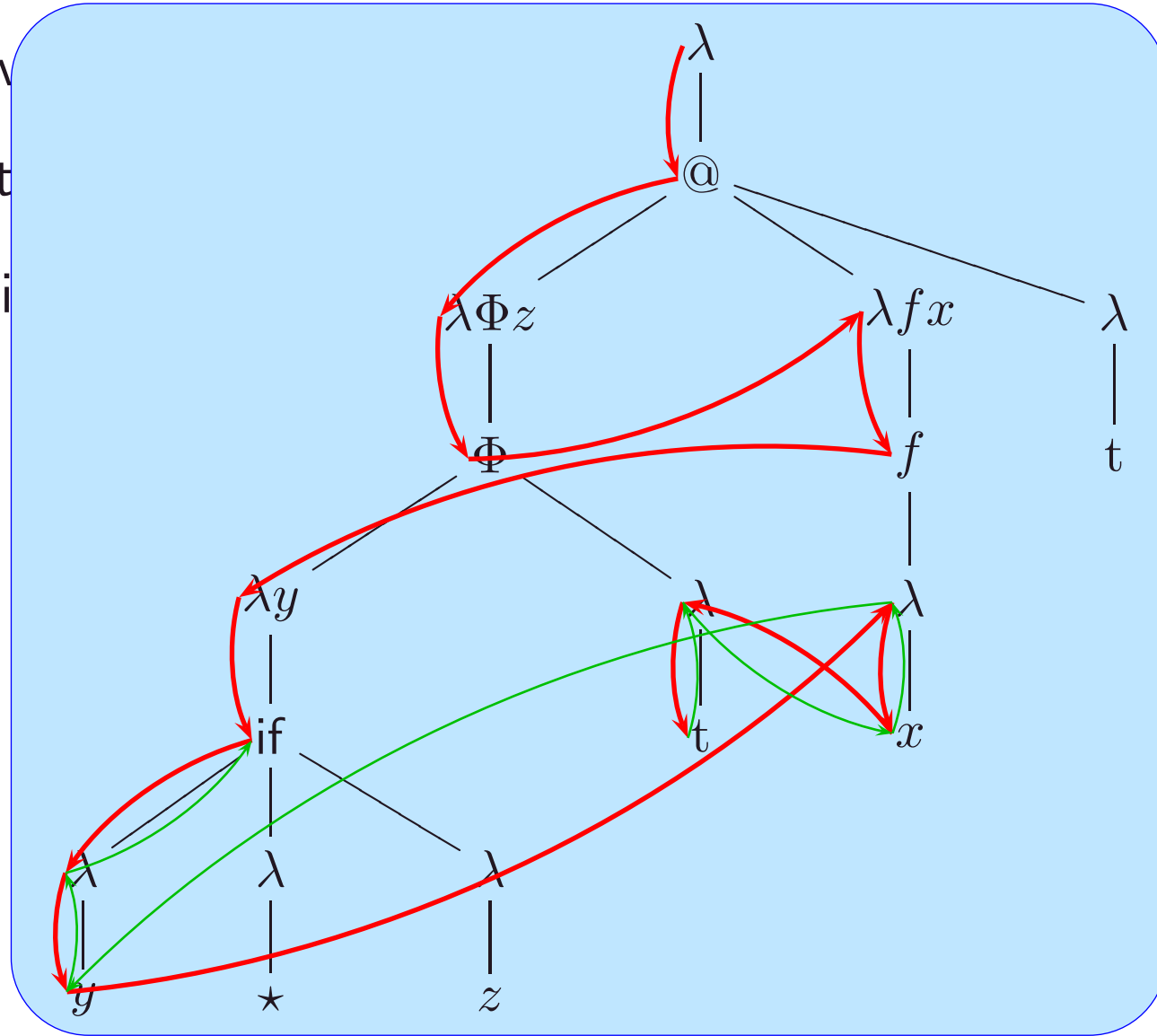
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

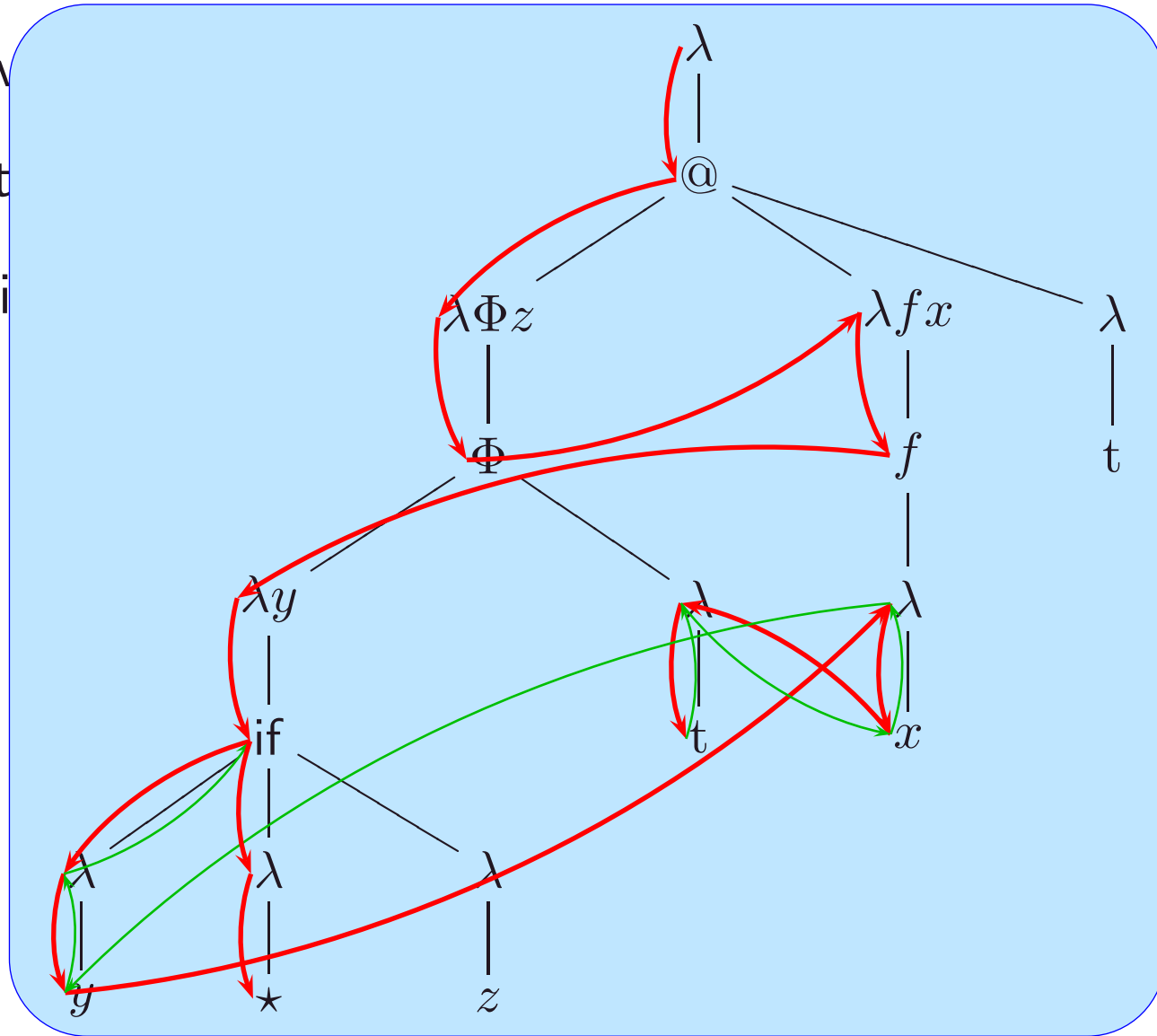
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

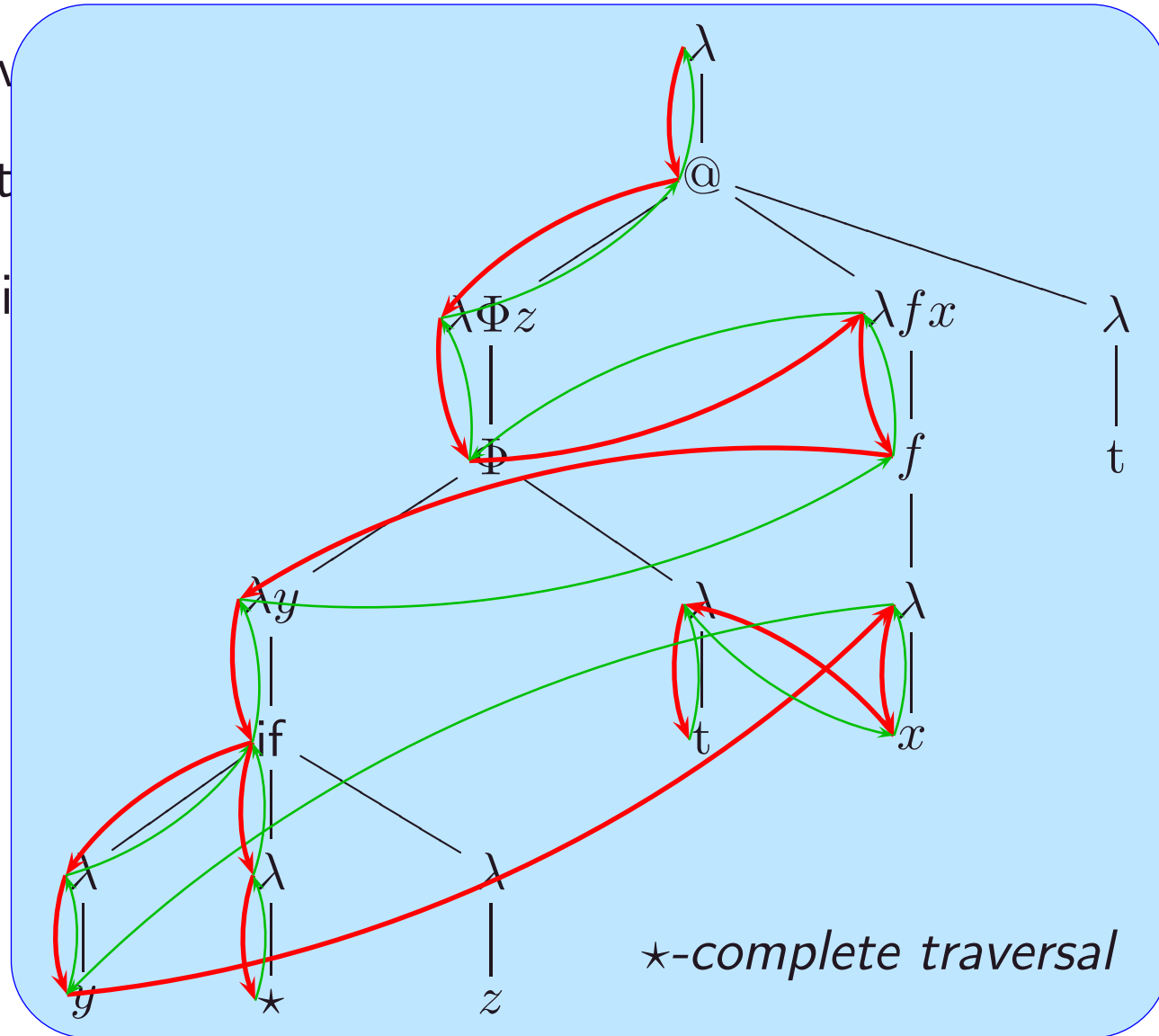
- follows the flow of control
- seen from the perspective



Traversals

A **traversal** [Blum, Ong] over

- follows the flow of control
- seen from the perspective



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

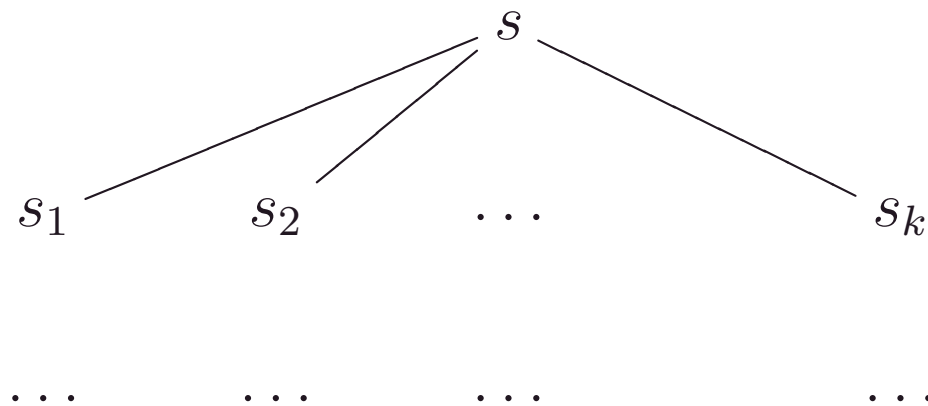
$$\begin{array}{l} s \in \Sigma \\ q \in Q \\ Q_1, \dots, Q_k \subseteq Q \end{array}$$

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

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

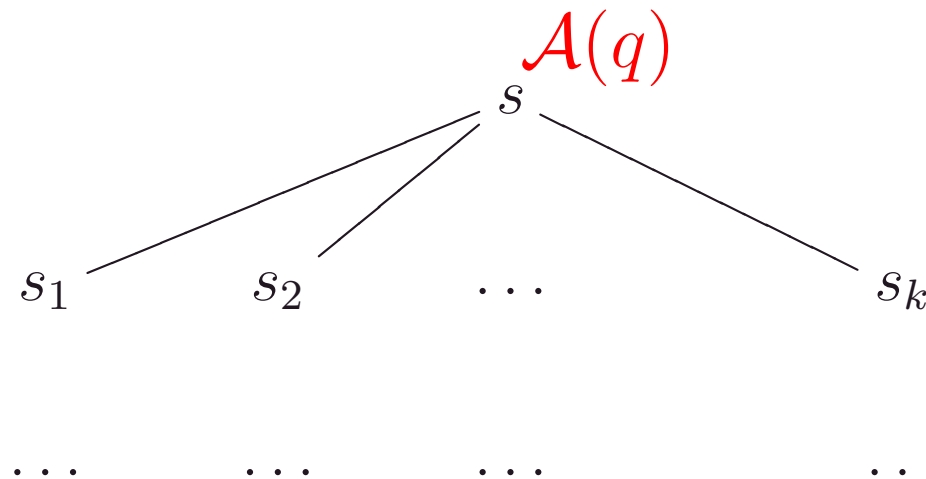


Alternating Tree Automata

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

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

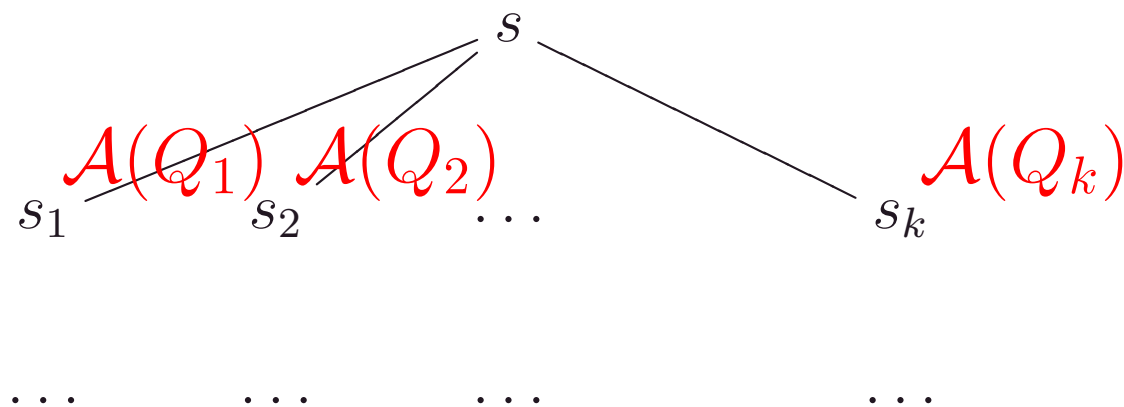


Alternating Tree Automata

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

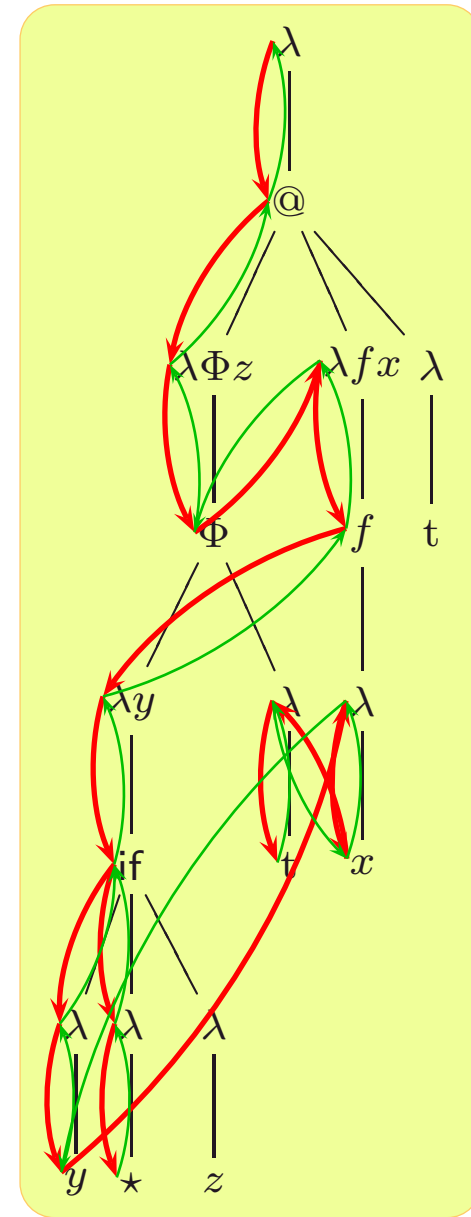
- Q is a finite set of states,
- Σ is a finite ranked alphabet,
- $q_0 \in Q$ is the initial state,
- Δ is a finite transition relation: $q \xrightarrow{s} (Q_1, \dots, Q_k)$.

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



Traversal-simulating ATA's

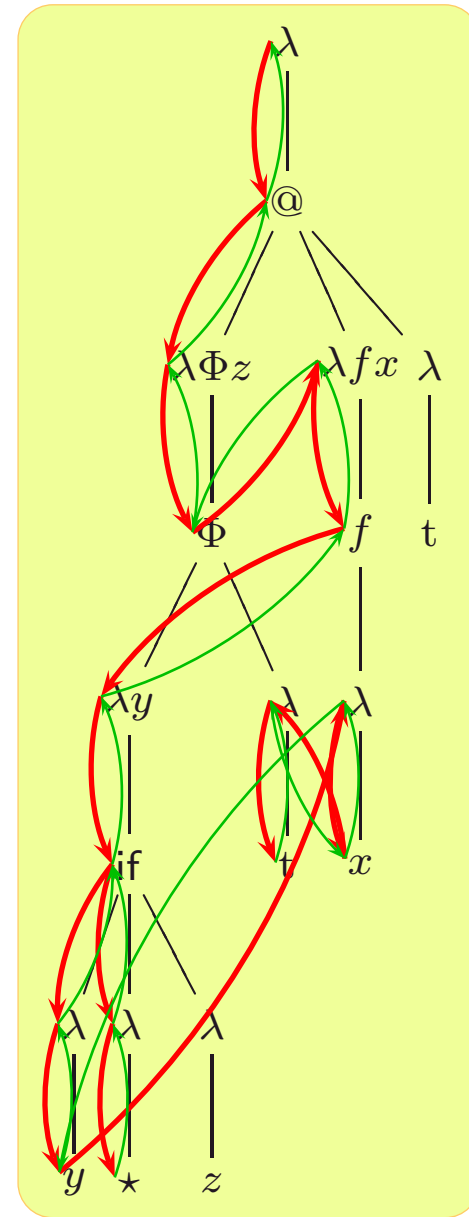
How can we simulate a complete traversal by an ATA?



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.



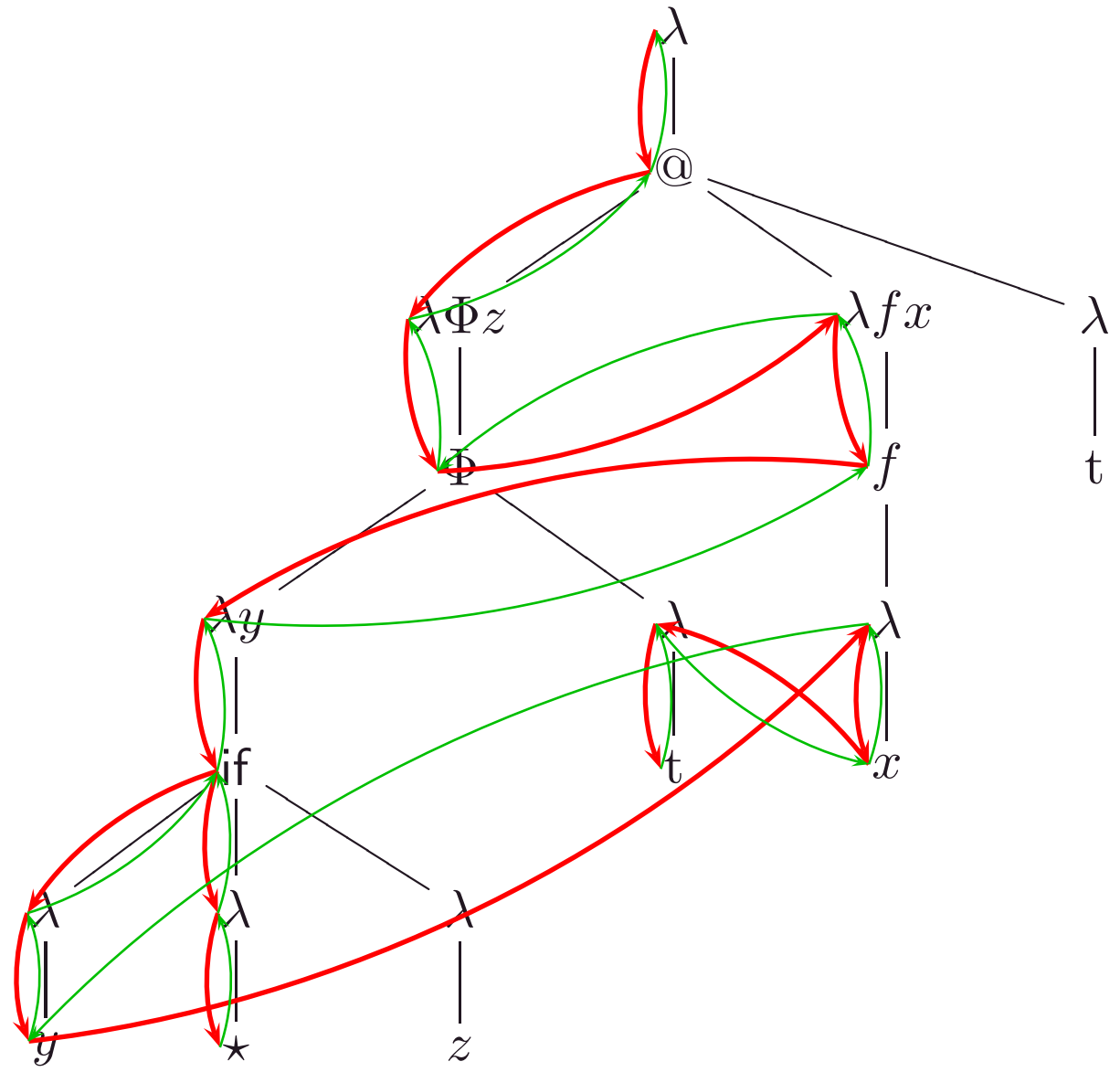
Variable profiles

Introduced in [Ong'06].

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

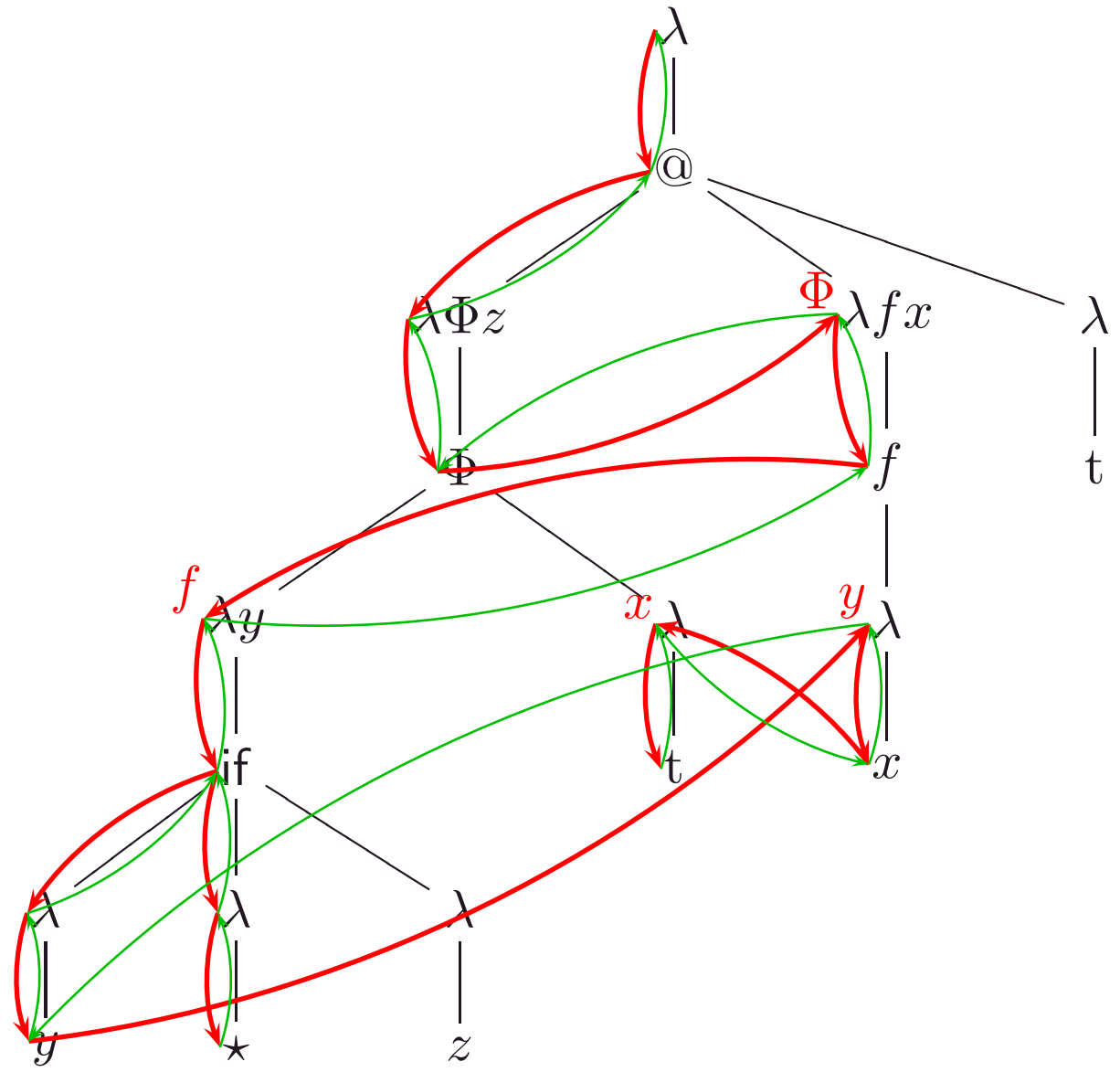
Variable profiles

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



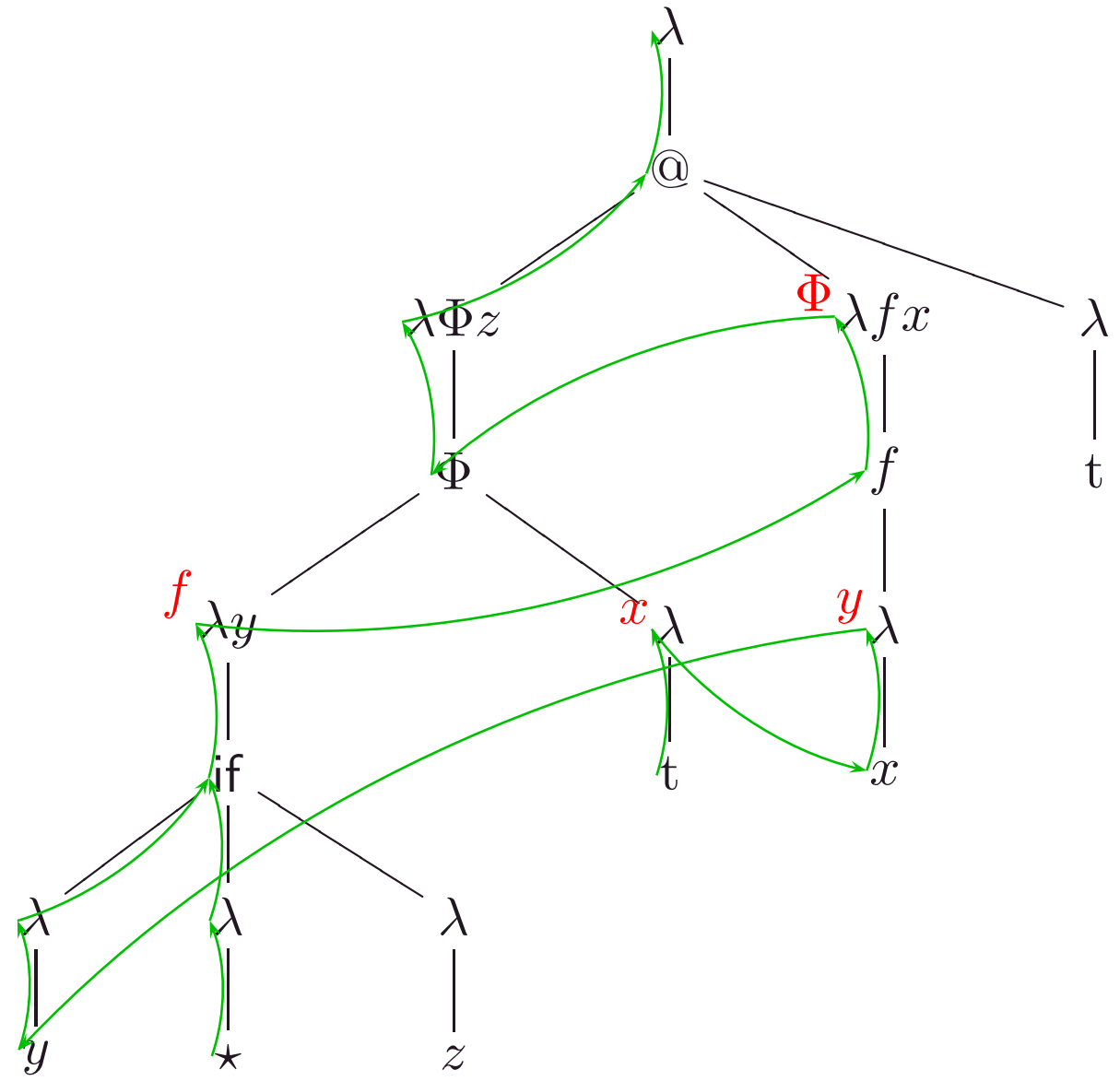
Variable profiles

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



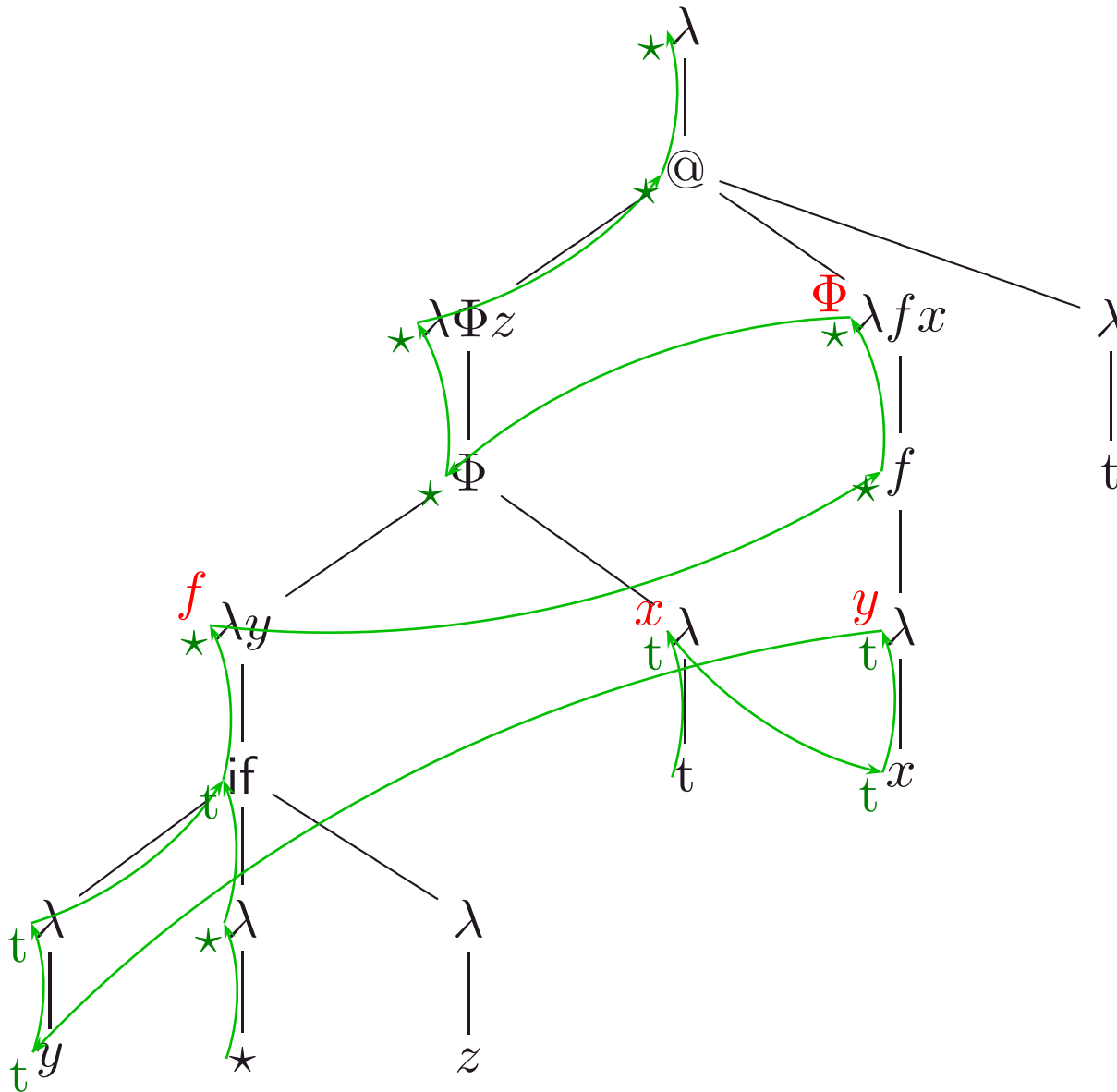
Variable profiles

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



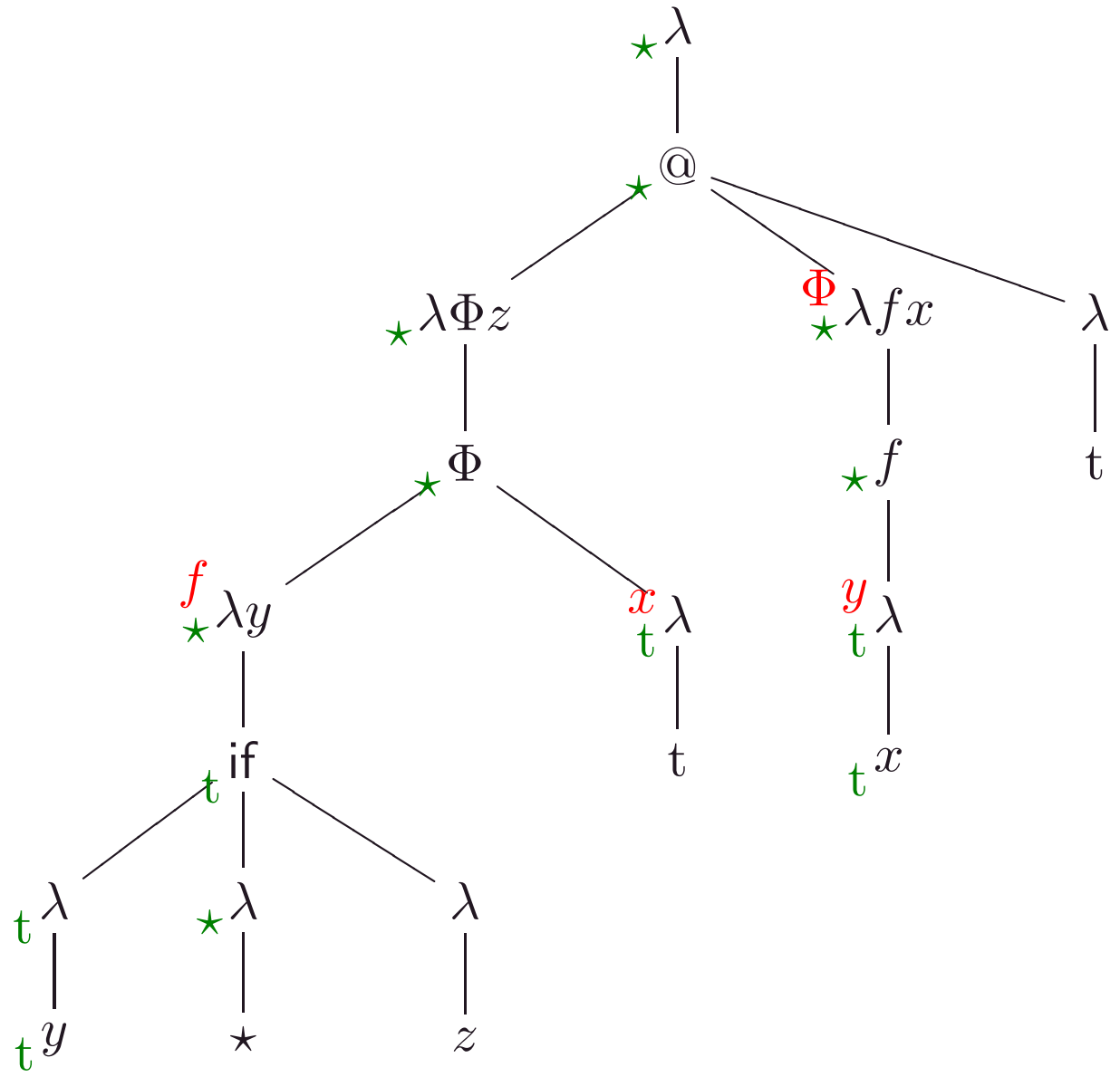
Variable profiles

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



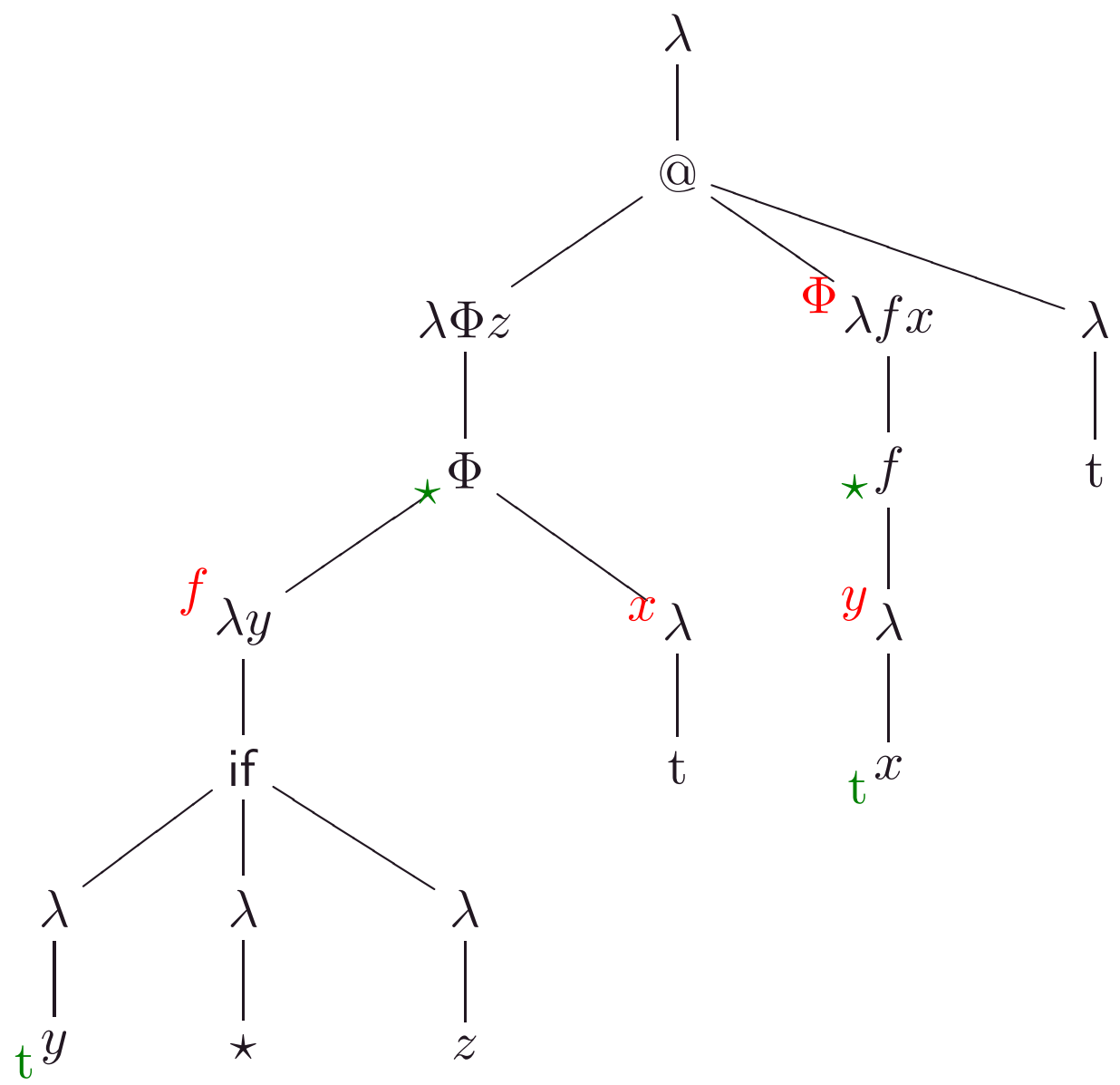
Variable profiles

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



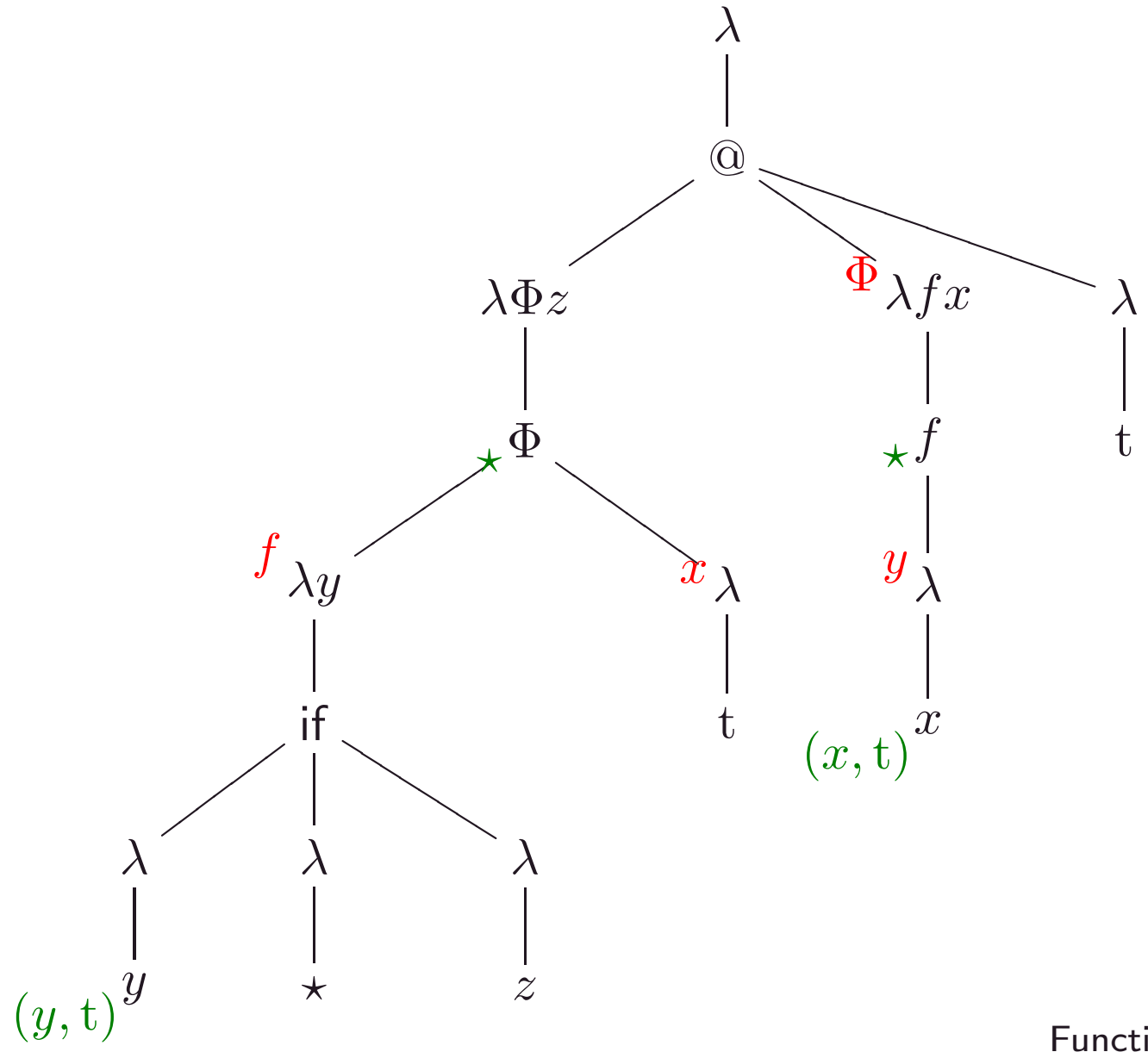
Variable profiles

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



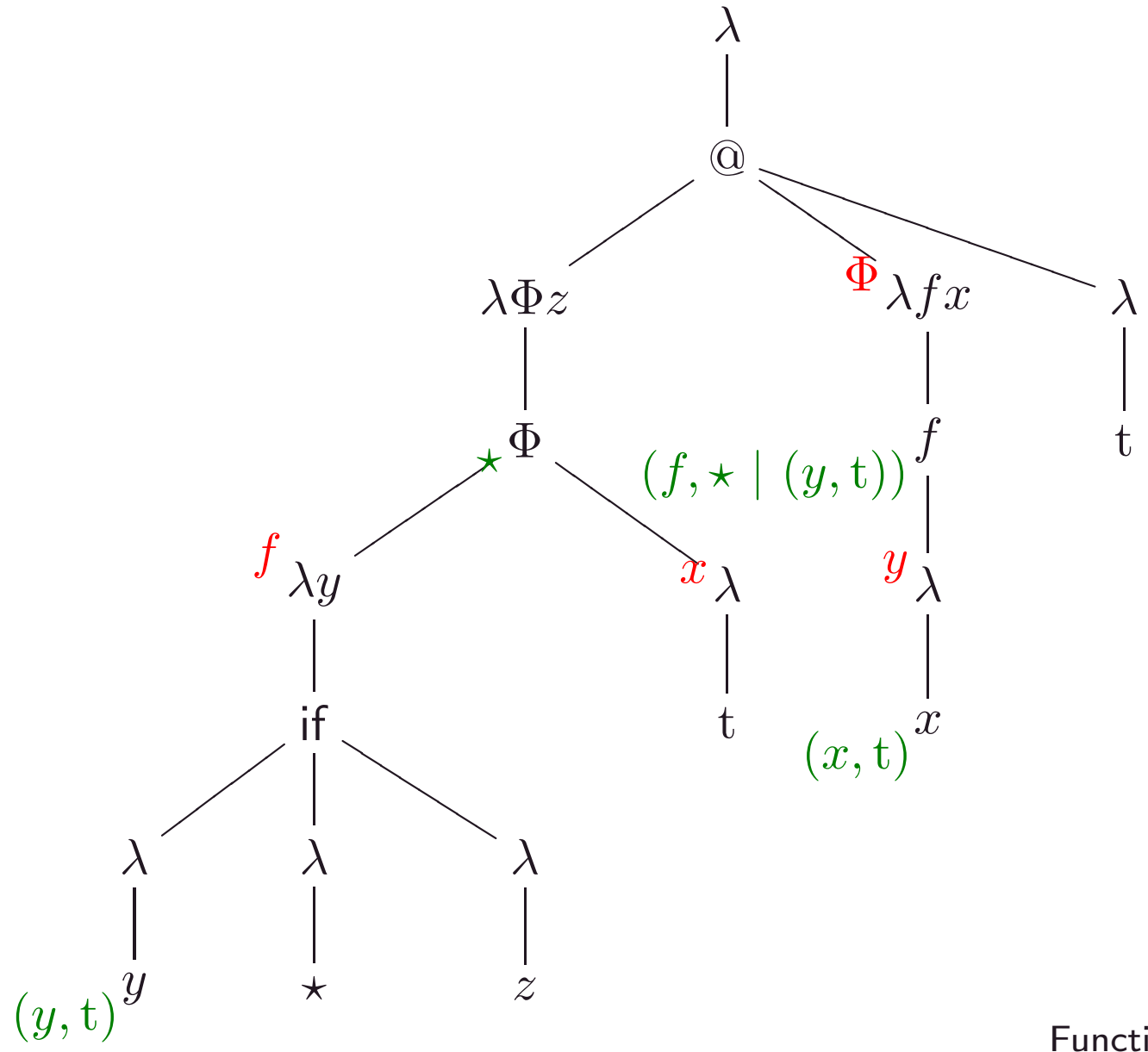
Variable profiles

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



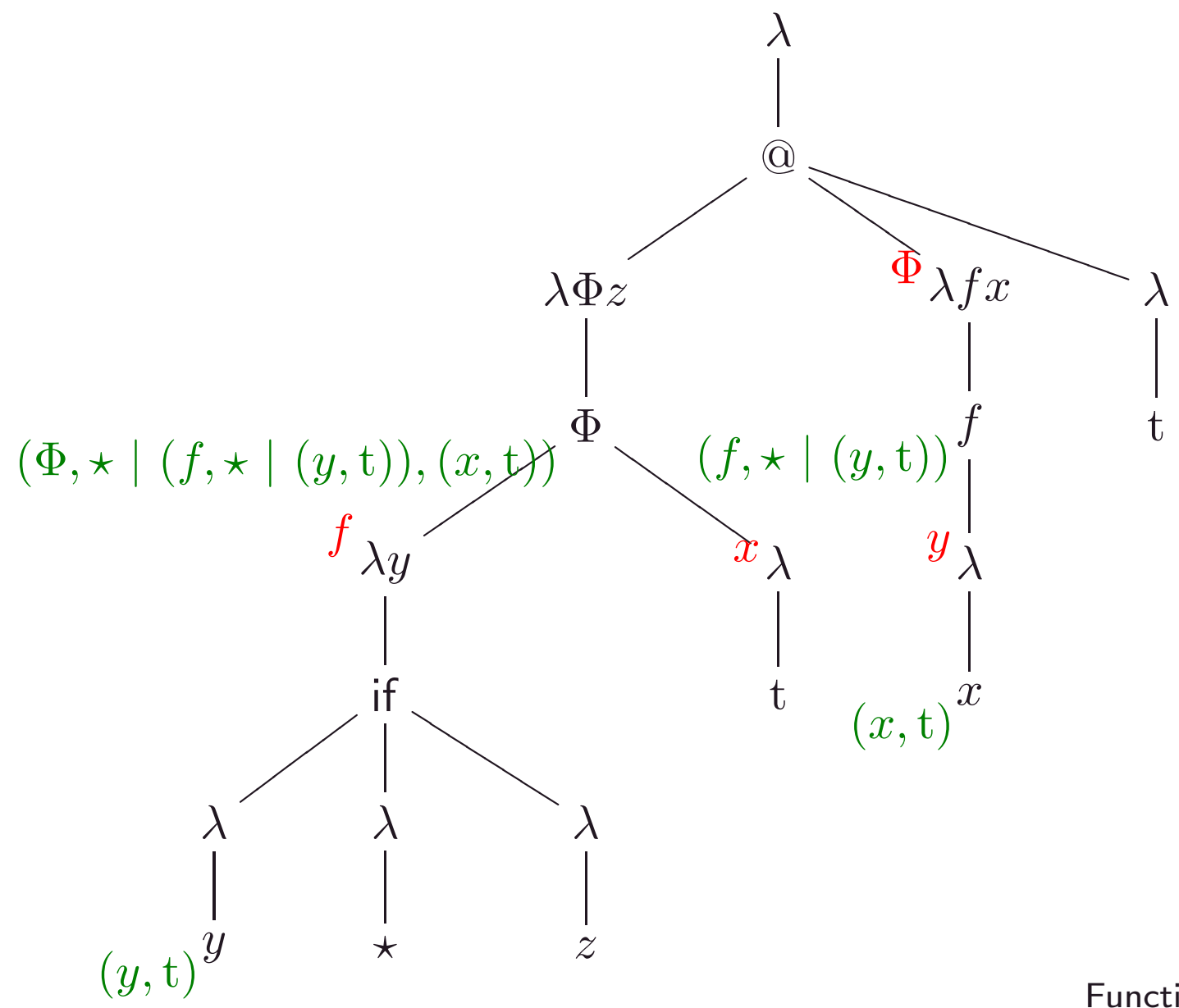
Variable profiles

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



Variable profiles

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



ATA correspondence

Given a finite fPCF^{*}-alphabet Σ , the states of the *traversal-simulating* ATA \mathcal{A}_Σ are:

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

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

Any tree accepted by \mathcal{A}_Σ on *closed* initial state *represents* a closed fPCF^{*}-term over Σ .

Results

Theorem: $M : (A_1, \dots, A_n, o) \in v\text{-REACH} [\text{fPCF}_\Sigma^*, \text{fPCF}_\Sigma]$

iff there is a closed initial state q_0 with value v such that:

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

Results

Theorem: $M : (A_1, \dots, A_n, o) \in v\text{-REACH} [\text{fPCF}_\Sigma^*, \text{fPCF}_\Sigma]$

iff there is a closed initial state q_0 with value v such that:

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

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

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

Results

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

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

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

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

- For the general case (also with \perp) 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.

- Conjecture: $\star\text{-REACH} [\text{fPCF}^*, \text{fPCF}] ?$
- Can this yield a (semantic) CFA?
- Extensions: expressivity, algorithms, abstractions.

- 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.

- Conjecture: $\star\text{-REACH} [\text{fPCF}^*, \text{fPCF}] ?$
- Can this yield a (semantic) CFA?
- Extensions: expressivity, algorithms, abstractions.