Inhabitation Machines: Determinism and Principality

Sandra Alves[1] Sabine Broda[2]

Computer Science Department
Faculty of Sciences, University of Porto, Portugal
[1] CRACS & [2] CMUP

NCMA 2017
Prague, Czech Republic

Overview

Overview

- Inhabitation problem in typed lambda calculus;

Overview

- Inhabitation problem in typed lambda calculus;
- corresponds to proof search intuitionistic logic;

Overview

- Inhabitation problem in typed lambda calculus;
- corresponds to proof search intuitionistic logic;
- different approaches over the years;

Overview

- Inhabitation problem in typed lambda calculus;
- corresponds to proof search intuitionistic logic;
- different approaches over the years;
  - backward inhabitation search; context-free grammars; formula-tree method; game semantics; etc.

Overview

- Inhabitation problem in typed lambda calculus;
- corresponds to proof search intuitionistic logic;
- different approaches over the years;
  - backward inhabitation search; context-free grammars; formula-tree method; game semantics; etc.
  - Schubert et al. (2015) use techniques from automata theory reproving PSPACE-completeness of the emptiness problem;

Overview

Overview

- inhabitation machines work locally in a non-deterministic way on lambda terms;

Overview

- inhabitation machines work locally in a non-deterministic way on lambda terms;
- our approach builds on this work combining it with the notion of pre-grammars;

Overview

- inhabitation machines work locally in a non-deterministic way on lambda terms;
- our approach builds on this work combining it with the notion of pre-grammars;
- the resulting inhabitation machines work deterministically;

Overview

- inhabitation machines work locally in a non-deterministic way on lambda terms;
- our approach builds on this work combining it with the notion of pre-grammars;
- the resulting inhabitation machines work deterministically;
- we extend the machines to address principal inhabitation.

Lambda Calculus

Lambda Calculus

- $x \in \mathcal{V} \quad \Rightarrow \quad x \in \Lambda$

Lambda Calculus

- $x \in \mathcal{V} \quad \Rightarrow \quad x \in \Lambda$
- $M, N \in \Lambda \quad \Rightarrow \quad MN \in \Lambda$

Lambda Calculus

- $x \in \mathcal{V} \quad \Rightarrow \quad x \in \Lambda$
- $M, N \in \Lambda \quad \Rightarrow \quad MN \in \Lambda$
- $x \in \mathcal{V}, M \in \Lambda \quad \Rightarrow \quad \lambda x.M \in \Lambda$

Lambda Calculus

- $x \in \mathcal{V} \quad \Rightarrow \quad x \in \Lambda$
- $M, N \in \Lambda \quad \Rightarrow \quad MN \in \Lambda$
- $x \in \mathcal{V}, M \in \Lambda \quad \Rightarrow \quad \lambda x.M \in \Lambda$

Associativity to the left, writing $MN_1 \cdots N_n$ instead of $(((MN_1)N_2)\cdots)$

Lambda Calculus

- $x \in \mathcal{V} \quad \Rightarrow \quad x \in \Lambda$
- $M, N \in \Lambda \quad \Rightarrow \quad MN \in \Lambda$
- $x \in \mathcal{V}, M \in \Lambda \quad \Rightarrow \quad \lambda x.M \in \Lambda$

Associativity to the left, writing $MN_1 \cdots N_n$ instead of $(((MN_1)N_2)\cdots)$

Examples: $\lambda xy.x(\lambda z.y)y$, $\lambda x.x$, ...

Simply Typed Lambda Calculus

Simply Typed Lambda Calculus

The set of simple types $\mathcal{T}$

- if a is a type variable, then $a \in \mathcal{T}$;
- $\alpha, \beta \in \mathcal{T} \quad \Rightarrow \quad (\alpha \to \beta) \in \mathcal{T}$;
- associativity to the right:
  $\alpha_1 \to \alpha_2 \to \cdots \to \alpha_n$ instead of $(\alpha_1 \to (\alpha_2 \to (\cdots \to \alpha_n)))$.

Simply Typed Lambda Calculus

The set of simple types $\mathcal{T}$

- if a is a type variable, then $a \in \mathcal{T}$;
- $\alpha, \beta \in \mathcal{T} \quad \Rightarrow \quad (\alpha \to \beta) \in \mathcal{T}$;
- associativity to the right:
  $\alpha_1 \to \alpha_2 \to \cdots \to \alpha_n$ instead of $(\alpha_1 \to (\alpha_2 \to (\cdots \to \alpha_n)))$.

Every type α can be uniquely written as $\alpha = \alpha_1 \to \ldots \to \alpha_n \to a$, where a is a type-variable and $n \geq 0$.

Simply Typed Lambda Calculus

The set of simple types $\mathcal{T}$

- if a is a type variable, then $a \in \mathcal{T}$;
- $\alpha, \beta \in \mathcal{T} \quad \Rightarrow \quad (\alpha \to \beta) \in \mathcal{T}$;
- associativity to the right:
  $\alpha_1 \to \alpha_2 \to \cdots \to \alpha_n$ instead of $(\alpha_1 \to (\alpha_2 \to (\cdots \to \alpha_n)))$.

Every type α can be uniquely written as $\alpha = \alpha_1 \to \ldots \to \alpha_n \to a$, where a is a type-variable and $n \geq 0$.

$\alpha_1, \ldots, \alpha_n$ are the arguments of α and a is its tail.

Simply Typed Lambda Calculus

The set of simple types $\mathcal{T}$

- if a is a type variable, then $a \in \mathcal{T}$;
- $\alpha, \beta \in \mathcal{T} \quad \Rightarrow \quad (\alpha \to \beta) \in \mathcal{T}$;
- associativity to the right:
  $\alpha_1 \to \alpha_2 \to \cdots \to \alpha_n$ instead of $(\alpha_1 \to (\alpha_2 \to (\cdots \to \alpha_n)))$.

Every type α can be uniquely written as $\alpha = \alpha_1 \to \ldots \to \alpha_n \to a$, where a is a type-variable and $n \geq 0$.

$\alpha_1, \ldots, \alpha_n$ are the arguments of α and a is its tail.

Example $\alpha = ((o \to o) \to o \to o) \to o \to o$

Simply Typed Lambda Calculus

Simply Typed Lambda Calculus

A **context Γ** is a finite set of *typed term variables* of the form x^α.

Simply Typed Lambda Calculus

A **context Γ** is a finite set of *typed term variables* of the form x^{α}.

$\Lambda^{\Gamma}(\alpha)$ **- terms of type α in the context Γ**

Simply Typed Lambda Calculus

A **context Γ** is a finite set of *typed term variables* of the form x^α.

$\Lambda^\Gamma(\alpha)$ **- terms of type α in the context Γ**

- $x^\alpha \in \Gamma \quad \Rightarrow \quad x^\alpha \in \Lambda^\Gamma(\alpha)$;
- $M \in \Lambda^\Gamma(\alpha \to \beta) \quad N \in \Lambda^\Gamma(\alpha) \quad \Rightarrow \quad MN \in \Lambda^\Gamma(\beta)$;
- $M \in \Lambda^{\Gamma \cup \{x^\alpha\}}(\beta) \quad \Rightarrow \quad \lambda x^\alpha . M \in \Lambda^\Gamma(\alpha \to \beta)$.

Simply Typed Lambda Calculus

A **context Γ** is a finite set of *typed term variables* of the form x^α.

$\Lambda^\Gamma(\alpha)$ **- terms of type α in the context Γ**

- $x^\alpha \in \Gamma \quad \Rightarrow \quad x^\alpha \in \Lambda^\Gamma(\alpha)$;
- $M \in \Lambda^\Gamma(\alpha \to \beta) \quad N \in \Lambda^\Gamma(\alpha) \quad \Rightarrow \quad MN \in \Lambda^\Gamma(\beta)$;
- $M \in \Lambda^{\Gamma \cup \{x^\alpha\}}(\beta) \quad \Rightarrow \quad \lambda x^\alpha . M \in \Lambda^\Gamma(\alpha \to \beta)$.

$\Lambda^\emptyset(((o \to o) \to o \to o) \to o \to o)$:

Simply Typed Lambda Calculus

A **context Γ** is a finite set of *typed term variables* of the form x^α.

$\Lambda^\Gamma(\alpha)$ **- terms of type α in the context Γ**

- $x^\alpha \in \Gamma \quad \Rightarrow \quad x^\alpha \in \Lambda^\Gamma(\alpha)$;
- $M \in \Lambda^\Gamma(\alpha \to \beta) \quad N \in \Lambda^\Gamma(\alpha) \quad \Rightarrow \quad MN \in \Lambda^\Gamma(\beta)$;
- $M \in \Lambda^{\Gamma \cup \{x^\alpha\}}(\beta) \quad \Rightarrow \quad \lambda x^\alpha . M \in \Lambda^\Gamma(\alpha \to \beta)$.

$\Lambda^\emptyset(((o \to o) \to o \to o) \to o \to o)$:

$\lambda x^{\tau_1} y^o . x^{\tau_1} (\lambda z^o . y^o) y^o$

$\lambda x^{\tau_1} . x^{\tau_1} (\lambda y^o . y^o)$

$\lambda x^{\tau_1} y^o . x^{\tau_1} (\lambda z^o . z^o)(x^{\tau_1} (\lambda u^o . y^o) y^o)$

$\lambda x^{\tau_1} y^o . y^o$

Simply Typed Lambda Calculus

A **context Γ** is a finite set of *typed term variables* of the form x^α.

$\Lambda^\Gamma(\alpha)$ **- terms of type α in the context Γ**

- $x^\alpha \in \Gamma \quad \Rightarrow \quad x^\alpha \in \Lambda^\Gamma(\alpha)$;
- $M \in \Lambda^\Gamma(\alpha \to \beta) \quad N \in \Lambda^\Gamma(\alpha) \quad \Rightarrow \quad MN \in \Lambda^\Gamma(\beta)$;
- $M \in \Lambda^{\Gamma \cup \{x^\alpha\}}(\beta) \quad \Rightarrow \quad \lambda x^\alpha.M \in \Lambda^\Gamma(\alpha \to \beta)$.

$\Lambda^\emptyset(((o \to o) \to o \to o) \to o \to o)$:

$\lambda xy.x(\lambda z.y)y$

$\lambda x.x(\lambda y.y)$

$\lambda xy.x(\lambda z.z)(x(\lambda u.y)y)$

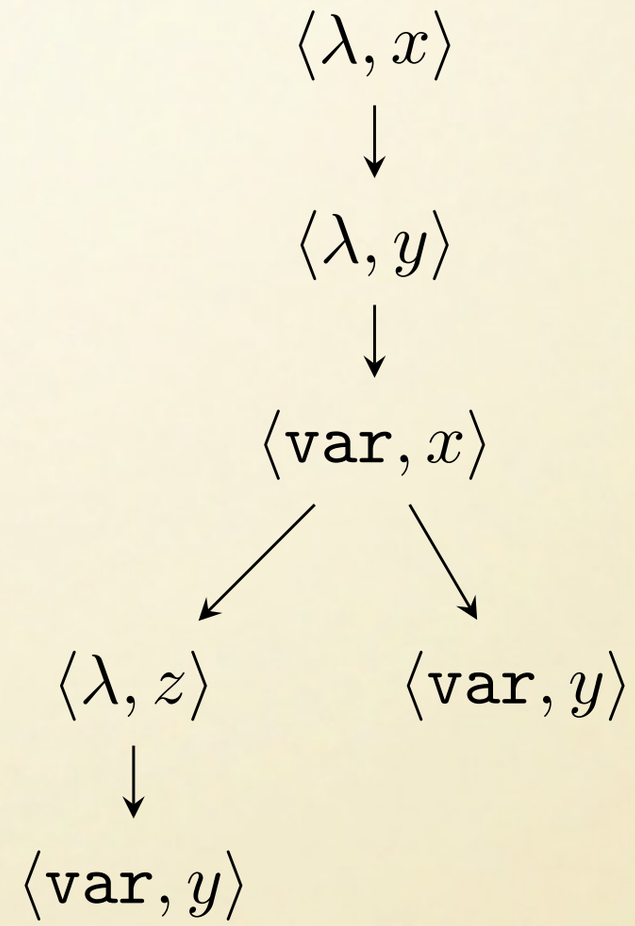
$\lambda xy.y$

Tree Representation of Terms

$\lambda xy.x(\lambda z.y)y$

Tree Representation of Terms

$\lambda xy.x(\lambda z.y)y$



Inhabitation Machines

Inhabitation Machines

$$\mathcal{A} = \langle \Sigma, N, Q, q_I, \mathcal{R}, \delta \rangle$$

Inhabitation Machines

$$\mathcal{A} = \langle \Sigma, N, Q, q_I, \mathcal{R}, \delta \rangle$$

$\Sigma = \{\lambda, \texttt{var}\}$

Inhabitation Machines

$$\mathcal{A} = \langle \Sigma, N, Q, q_I, \mathcal{R}, \delta \rangle$$

$\Sigma = \{\lambda, \texttt{var}\}$

N - infinite set of data elements

Inhabitation Machines

$$\mathcal{A} = \langle \Sigma, N, Q, q_I, \mathcal{R}, \delta \rangle$$

$\Sigma = \{\lambda, \texttt{var}\}$

N - infinite set of data elements

Q - finite set of states

Inhabitation Machines

$$\mathcal{A} = \langle \Sigma, N, Q, q_I, \mathcal{R}, \delta \rangle$$

$\Sigma = \{\lambda, \texttt{var}\}$

N - infinite set of data elements

Q - finite set of states $\qquad\qquad q_I \in Q$

Inhabitation Machines

$$\mathcal{A} = \langle \Sigma, N, Q, q_I, \mathcal{R}, \delta \rangle$$

$\Sigma = \{\lambda, \texttt{var}\}$

N - infinite set of data elements

Q - finite set of states $\qquad q_I \in Q$

$\mathcal{R}$ - finite set of register names

Inhabitation Machines

$$\mathcal{A} = \langle \Sigma, N, Q, q_I, \mathcal{R}, \delta \rangle$$

$\Sigma = \{\lambda, \texttt{var}\}$

N - infinite set of data elements

Q - finite set of states $\qquad q_I \in Q$

$\mathcal{R}$ - finite set of register names

$\delta \subseteq \Sigma \times Q \times (\mathcal{R} \cup \{\varnothing\}) \times \vec{Q} \times (\mathcal{R} \cup \{\varnothing\})$ - the transition relation

Inhabitation Machines

$$\mathcal{A} = \langle \Sigma, N, Q, q_I, \mathcal{R}, \delta \rangle$$

$\Sigma = \{\lambda, \texttt{var}\}$

N - infinite set of data elements

Q - finite set of states $\qquad q_I \in Q$

$\mathcal{R}$ - finite set of register names

$\delta \subseteq \Sigma \times Q \times (\mathcal{R} \cup \{\varnothing\}) \times \vec{Q} \times (\mathcal{R} \cup \{\varnothing\})$ - the transition relation

$$a, q, \texttt{r} \rightsquigarrow q_1, \ldots, q_n, \texttt{r}'$$

Operational Semantics

Operational Semantics

- $\mathcal{A}$ traverses tree representations of λ-terms in normal form

Operational Semantics

- $\mathcal{A}$ traverses tree representations of λ-terms in normal form
- during a run variable names (elements of N) are stored in the registers

Operational Semantics

- $\mathcal{A}$ traverses tree representations of λ-terms in normal form
- during a run variable names (elements of N) are stored in the registers
- a configuration of $\mathcal{A}$ in a tree $\mathbf{t}$ is a finite sequence of tuples of the form $\mathtt{c}_i = (\mathbf{t}_i, q_i, \mathtt{R}_i)$

Operational Semantics

- $\mathcal{A}$ traverses tree representations of λ-terms in normal form
- during a run variable names (elements of N) are stored in the registers
- a configuration of $\mathcal{A}$ in a tree $\mathbf{t}$ is a finite sequence of tuples of the form $\mathtt{c}_i = (\mathbf{t}_i, q_i, \mathtt{R}_i)$
- the initial configuration is $\mathtt{c} = (\mathbf{t}, q_I, \mathtt{R}_\varnothing)$

Operational Semantics

Operational Semantics

A non-empty sequence $\mathtt{c}_1, \mathtt{c}_2, \ldots, \mathtt{c}_n$ with $\mathtt{c}_1 = (\mathbf{t}_1, q_i, \mathtt{R}_1)$ transitions to $\mathtt{c}^1, \ldots, \mathtt{c}^k, \mathtt{c}_2, \ldots, \mathtt{c}_n$ if one of the following applies:

Operational Semantics

A non-empty sequence $\mathtt{c}_1, \mathtt{c}_2, \ldots, \mathtt{c}_n$ with $\mathtt{c}_1 = (\mathbf{t}_1, q_i, \mathtt{R}_1)$ transitions to $\mathtt{c}^1, \ldots, \mathtt{c}^k, \mathtt{c}_2, \ldots, \mathtt{c}_n$ if one of the following applies:

- $\mathbf{t}_1$ has root $\langle \lambda, x \rangle$ and a single subtree $\mathbf{t}'$, there is some rule $\lambda, q_i, \varnothing \rightsquigarrow q^1, \mathtt{r}$ in δ, $k = 1$, and $\mathtt{c}^1 = (\mathbf{t}', q^1, \mathtt{R}')$, where $\mathtt{R}'$ is obtained from $\mathtt{R}_1$ by updating register $\mathtt{r}$ with the addition of variable x

Operational Semantics

A non-empty sequence $\mathtt{c}_1, \mathtt{c}_2, \ldots, \mathtt{c}_n$ with $\mathtt{c}_1 = (\mathbf{t}_1, q_i, \mathtt{R}_1)$ transitions to $\mathtt{c}^1, \ldots, \mathtt{c}^k, \mathtt{c}_2, \ldots, \mathtt{c}_n$ if one of the following applies:

- $\mathbf{t}_1$ has root $\langle \lambda, x \rangle$ and a single subtree $\mathbf{t}'$, there is some rule $\lambda, q_i, \varnothing \rightsquigarrow q^1, \mathtt{r}$ in δ, $k = 1$, and $\mathtt{c}^1 = (\mathbf{t}', q^1, \mathtt{R}')$, where $\mathtt{R}'$ is obtained from $\mathtt{R}_1$ by updating register $\mathtt{r}$ with the addition of variable x
- $\mathbf{t}_1$ has root $\langle \mathtt{var}, x \rangle$ and $k \geq 0$ subtrees $\mathbf{t}^1, \ldots, \mathbf{t}^k$, there is some rule $\mathtt{var}, q_i, \mathtt{r} \rightsquigarrow q^1, \ldots, q^k, \varnothing$ in δ, $x \in \mathtt{cont}(\mathtt{R}_1, \mathtt{r})$, and for $1 \leq j \leq k$ one has $\mathtt{c}^j = (\mathbf{t}^j, q^j, \mathtt{R}_1)$

Operational Semantics

A non-empty sequence $\mathtt{c}_1, \mathtt{c}_2, \ldots, \mathtt{c}_n$ with $\mathtt{c}_1 = (\mathbf{t}_1, q_i, \mathtt{R}_1)$ transitions to $\mathtt{c}^1, \ldots, \mathtt{c}^k, \mathtt{c}_2, \ldots, \mathtt{c}_n$ if one of the following applies:

- $\mathbf{t}_1$ has root $\langle \lambda, x \rangle$ and a single subtree $\mathbf{t}'$, there is some rule $\lambda, q_i, \varnothing \rightsquigarrow q^1, \mathtt{r}$ in δ, $k = 1$, and $\mathtt{c}^1 = (\mathbf{t}', q^1, \mathtt{R}')$, where $\mathtt{R}'$ is obtained from $\mathtt{R}_1$ by updating register $\mathtt{r}$ with the addition of variable x

- $\mathbf{t}_1$ has root $\langle \mathtt{var}, x \rangle$ and $k \geq 0$ subtrees $\mathbf{t}^1, \ldots, \mathbf{t}^k$, there is some rule $\mathtt{var}, q_i, \mathtt{r} \rightsquigarrow q^1, \ldots, q^k, \varnothing$ in δ, $x \in \mathtt{cont}(\mathtt{R}_1, \mathtt{r})$, and for $1 \leq j \leq k$ one has $\mathtt{c}^j = (\mathbf{t}^j, q^j, \mathtt{R}_1)$

- The empty configuration sequence represents success

Operational Semantics

A non-empty sequence $\mathtt{c}_1, \mathtt{c}_2, \ldots, \mathtt{c}_n$ with $\mathtt{c}_1 = (\mathbf{t}_1, q_i, \mathtt{R}_1)$ transitions to $\mathtt{c}^1, \ldots, \mathtt{c}^k, \mathtt{c}_2, \ldots, \mathtt{c}_n$ if one of the following applies:

- $\mathbf{t}_1$ has root $\langle \lambda, x \rangle$ and a single subtree $\mathbf{t}'$, there is some rule $\lambda, q_i, \varnothing \rightsquigarrow q^1, \mathtt{r}$ in δ, $k = 1$, and $\mathtt{c}^1 = (\mathbf{t}', q^1, \mathtt{R}')$, where $\mathtt{R}'$ is obtained from $\mathtt{R}_1$ by updating register $\mathtt{r}$ with the addition of variable x
- $\mathbf{t}_1$ has root $\langle \mathtt{var}, x \rangle$ and $k \geq 0$ subtrees $\mathbf{t}^1, \ldots, \mathbf{t}^k$, there is some rule $\mathtt{var}, q_i, \mathtt{r} \rightsquigarrow q^1, \ldots, q^k, \varnothing$ in δ, $x \in \mathtt{cont}(\mathtt{R}_1, \mathtt{r})$, and for $1 \leq j \leq k$ one has $\mathtt{c}^j = (\mathbf{t}^j, q^j, \mathtt{R}_1)$
- The empty configuration sequence represents success
- $\mathcal{A}$ accepts a tree $\mathbf{t}$ if there is a run from the initial configuration to the empty sequence

Operational Semantics

A non-empty sequence $\mathtt{c}_1, \mathtt{c}_2, \ldots, \mathtt{c}_n$ with $\mathtt{c}_1 = (\mathbf{t}_1, q_i, \mathtt{R}_1)$ transitions to $\mathtt{c}^1, \ldots, \mathtt{c}^k, \mathtt{c}_2, \ldots, \mathtt{c}_n$ if one of the following applies:

- $\mathbf{t}_1$ has root $\langle \lambda, x \rangle$ and a single subtree $\mathbf{t}'$, there is some rule $\lambda, q_i, \varnothing \rightsquigarrow q^1, \mathtt{r}$ in δ, $k = 1$, and $\mathtt{c}^1 = (\mathbf{t}', q^1, \mathtt{R}')$, where $\mathtt{R}'$ is obtained from $\mathtt{R}_1$ by updating register $\mathtt{r}$ with the addition of variable x
- $\mathbf{t}_1$ has root $\langle \mathtt{var}, x \rangle$ and $k \geq 0$ subtrees $\mathbf{t}^1, \ldots, \mathbf{t}^k$, there is some rule $\mathtt{var}, q_i, \mathtt{r} \rightsquigarrow q^1, \ldots, q^k, \varnothing$ in δ, $x \in \mathtt{cont}(\mathtt{R}_1, \mathtt{r})$, and for $1 \leq j \leq k$ one has $\mathtt{c}^j = (\mathbf{t}^j, q^j, \mathtt{R}_1)$

- The empty configuration sequence represents success
- $\mathcal{A}$ accepts a tree $\mathbf{t}$ if there is a run from the initial configuration to the empty sequence
- the set of normal terms M such that $\mathcal{A}$ accepts $\mathbf{t}^M$ is denoted by $\mathcal{L}(\mathcal{A})$

Pre-grammars

Pre-grammars

- given type α its pre-grammar $\texttt{pre}(\alpha)$ is based on set $\texttt{OccT}(\alpha)$

Pre-grammars

- given type α its pre-grammar $\texttt{pre}(\alpha)$ is based on set $\texttt{OccT}(\alpha)$
- $\texttt{OccT}(\alpha)$ contains for each occurrence of a subtype β a tuple (β, n, l), where $n \in \mathbb{N}$, and $l \in \{\texttt{var}\} \cup \{\, n \to m \mid n, m \in \mathbb{N} \,\}$

Pre-grammars

- given type α its pre-grammar $\texttt{pre}(\alpha)$ is based on set $\texttt{OccT}(\alpha)$
- $\texttt{OccT}(\alpha)$ contains for each occurrence of a subtype β a tuple (β, n, l), where $n \in \mathbb{N}$, and $l \in \{\texttt{var}\} \cup \{\, n \to m \mid n, m \in \mathbb{N} \,\}$
- $\texttt{N}(\alpha) = \{\, n \mid (\beta, n, l) \in \texttt{OccT}(\alpha) \,\}$

Pre-grammars

- given type α its pre-grammar $\texttt{pre}(\alpha)$ is based on set $\texttt{OccT}(\alpha)$
- $\texttt{OccT}(\alpha)$ contains for each occurrence of a subtype β a tuple (β, n, l), where $n \in \mathbb{N}$, and $l \in \{\texttt{var}\} \cup \{\, n \to m \mid n, m \in \mathbb{N} \,\}$
- $\texttt{N}(\alpha) = \{\, n \mid (\beta, n, l) \in \texttt{OccT}(\alpha) \,\}$
- elements of $\texttt{N}(\alpha)$ may be superscripted negatively/positively

Pre-grammars

- given type α its pre-grammar $\texttt{pre}(\alpha)$ is based on set $\texttt{OccT}(\alpha)$
- $\texttt{OccT}(\alpha)$ contains for each occurrence of a subtype β a tuple (β, n, l), where $n \in \mathbb{N}$, and $l \in \{\texttt{var}\} \cup \{\, n \to m \mid n, m \in \mathbb{N} \,\}$
- $\texttt{N}(\alpha) = \{\, n \mid (\beta, n, l) \in \texttt{OccT}(\alpha) \,\}$
- elements of $\texttt{N}(\alpha)$ may be superscripted negatively/positively
- $T(\alpha) \subseteq \texttt{N}(\alpha) \times \texttt{N}(\alpha)$ defined by $(k, n) \in T(\alpha)$ iff $(\beta, n, m \to k) \in \texttt{OccT}(\alpha)$

Pre-grammars

- given type α its pre-grammar $\texttt{pre}(\alpha)$ is based on set $\texttt{OccT}(\alpha)$
- $\texttt{OccT}(\alpha)$ contains for each occurrence of a subtype β a tuple (β, n, l), where $n \in \mathbb{N}$, and $l \in \{\texttt{var}\} \cup \{\, n \to m \mid n, m \in \mathbb{N} \,\}$
- $\texttt{N}(\alpha) = \{\, n \mid (\beta, n, l) \in \texttt{OccT}(\alpha) \,\}$
- elements of $\texttt{N}(\alpha)$ may be superscripted negatively/positively
- $T(\alpha) \subseteq \texttt{N}(\alpha) \times \texttt{N}(\alpha)$ defined by $(k, n) \in T(\alpha)$ iff $(\beta, n, m \to k) \in \texttt{OccT}(\alpha)$
- $n \equiv_{\texttt{OccT}} m$, if and only if they correspond to the same subtype

Pre-grammars

Pre-grammars

$$\alpha = ((o \to o) \to o \to o) \to o \to o$$

Pre-grammars

$\alpha = ((o \to o) \to o \to o) \to o \to o$

β	n	l
o	0	`var`
o	1	`var`
o	2	`var`
o	3	`var`
o	4	`var`
o	5	`var`
$o \to o$	6	$0 \to 1$
$o \to o$	7	$2 \to 3$
$o \to o$	8	$4 \to 5$
$(o \to o) \to o \to o$	9	$6 \to 7$
α	10	$9 \to 8$

Pre-grammars

$\alpha = ((o \to o) \to o \to o) \to o \to o$

equivalence classes of $\equiv_{\texttt{OccT}}$:

β	n	l
o	0	`var`
o	1	`var`
o	2	`var`
o	3	`var`
o	4	`var`
o	5	`var`
$o \to o$	6	$0 \to 1$
$o \to o$	7	$2 \to 3$
$o \to o$	8	$4 \to 5$
$(o \to o) \to o \to o$	9	$6 \to 7$
α	10	$9 \to 8$

Pre-grammars

$\alpha = ((o \to o) \to o \to o) \to o \to o$

equivalence classes of $\equiv_{\texttt{OccT}}$:

$\{10^+\}$ $\{9^-\}$ $\{6^+, 7, 8^+\}$
$\{0^-, 1^+, 2^+, 3, 4^-, 5^+\}$

β	n	l
o	0	`var`
o	1	`var`
o	2	`var`
o	3	`var`
o	4	`var`
o	5	`var`
$o \to o$	6	$0 \to 1$
$o \to o$	7	$2 \to 3$
$o \to o$	8	$4 \to 5$
$(o \to o) \to o \to o$	9	$6 \to 7$
α	10	$9 \to 8$

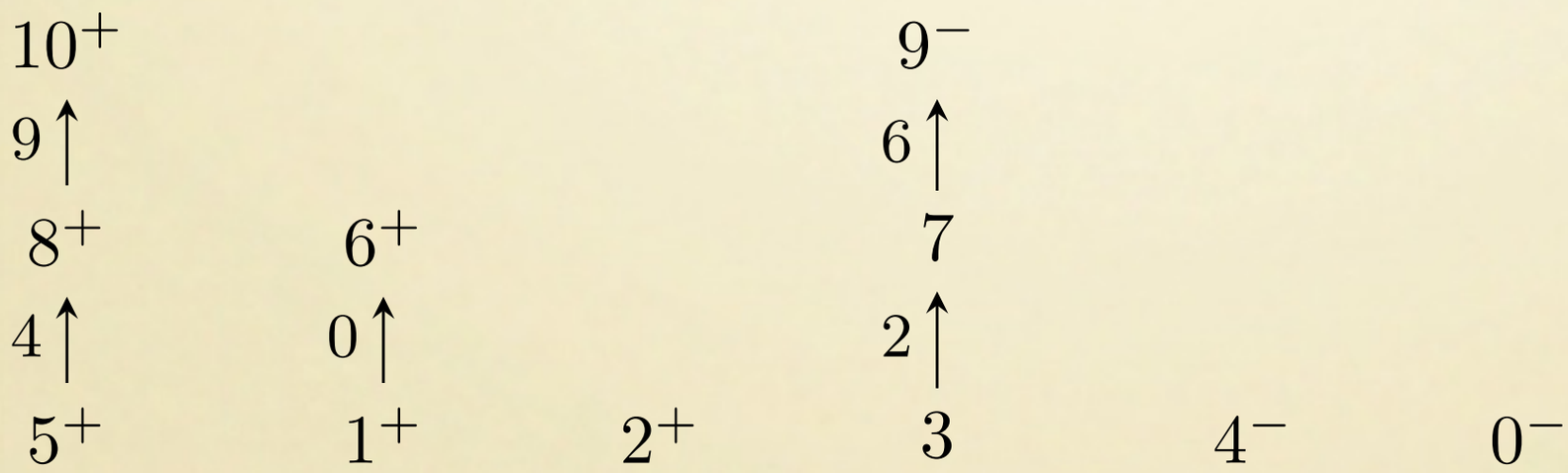
Pre-grammars

$\alpha = ((o \to o) \to o \to o) \to o \to o$

β	n	l
o	0	`var`
o	1	`var`
o	2	`var`
o	3	`var`
o	4	`var`
o	5	`var`
$o \to o$	6	$0 \to 1$
$o \to o$	7	$2 \to 3$
$o \to o$	8	$4 \to 5$
$(o \to o) \to o \to o$	9	$6 \to 7$
α	10	$9 \to 8$

equivalence classes of $\equiv_{\texttt{OccT}}$:

$\{10^+\}$ $\{9^-\}$ $\{6^+, 7, 8^+\}$
$\{0^-, 1^+, 2^+, 3, 4^-, 5^+\}$



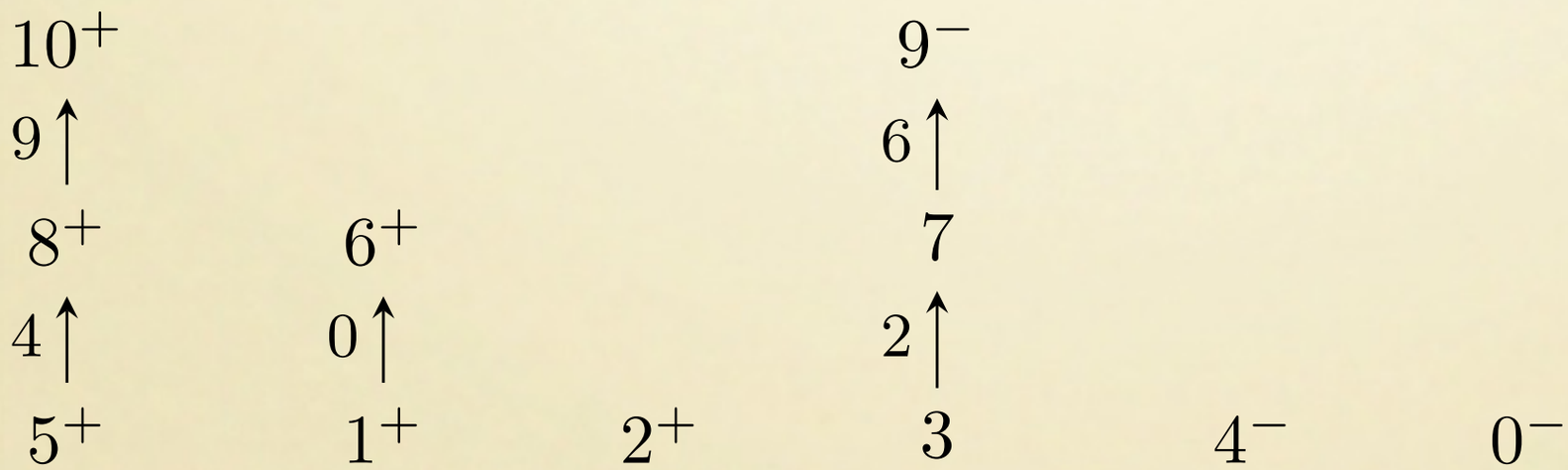
Pre-grammars

$\alpha = ((o \to o) \to o \to o) \to o \to o$

equivalence classes of $\equiv_{\texttt{OccT}}$:

$\{10^+\}\ \{9^-\}\ \{6^+, 7, 8^+\}$
$\{0^-, 1^+, 2^+, 3, 4^-, 5^+\}$

$$
\begin{array}{lcl}
10 & := & \lambda 9.8 \\
8 & := & \lambda 4.5 \mid 9\ 6 \\
6 & := & \lambda 0.1 \mid 9\ 6 \\
5 & := & 9\ 6\ 2 \mid 4 \mid 0 \\
2 & := & 9\ 6 \mid 4 \mid 0 \\
1 & := & 9\ 6\ 2 \mid 4 \mid 0
\end{array}
$$



Inhabitation

Inhabitation

$$
\begin{array}{lll}
10 & := & \lambda 9.8 \\
8 & := & \lambda 4.5 \mid 9\ 6 \\
6 & := & \lambda 0.1 \mid 9\ 6 \\
5 & := & 9\ 6\ 2 \mid 4 \mid 0 \\
2 & := & 9\ 6 \mid 4 \mid 0 \\
1 & := & 9\ 6\ 2 \mid 4 \mid 0
\end{array}
$$

Inhabitation

$$
\begin{array}{lll}
10 & := & \lambda 9.8 \\
8 & := & \lambda 4.5 \mid 9\ 6 \\
6 & := & \lambda 0.1 \mid 9\ 6 \\
5 & := & 9\ 6\ 2 \mid 4 \mid 0 \\
2 & := & 9\ 6 \mid 4 \mid 0 \\
1 & := & 9\ 6\ 2 \mid 4 \mid 0
\end{array}
$$

$$A_\alpha = \langle \{\lambda, \mathtt{var}\}, \mathcal{V}, Q, q_{10}, \mathcal{R}, \delta \rangle$$

Inhabitation

$$
\begin{array}{lll}
10 & := & \lambda 9.8 \\
8 & := & \lambda 4.5 \mid 9\ 6 \\
6 & := & \lambda 0.1 \mid 9\ 6 \\
5 & := & 9\ 6\ 2 \mid 4 \mid 0 \\
2 & := & 9\ 6 \mid 4 \mid 0 \\
1 & := & 9\ 6\ 2 \mid 4 \mid 0
\end{array}
$$

$$A_\alpha = \langle \{\lambda, \mathtt{var}\}, \mathcal{V}, Q, q_{10}, \mathcal{R}, \delta \rangle$$

$$Q = \{q_1, q_2, q_5, q_6, q_8, q_{10}\}$$

Inhabitation

$$
\begin{array}{lll}
10 & := & \lambda 9.8 \\
8 & := & \lambda 4.5 \mid 9\ 6 \\
6 & := & \lambda 0.1 \mid 9\ 6 \\
5 & := & 9\ 6\ 2 \mid 4 \mid 0 \\
2 & := & 9\ 6 \mid 4 \mid 0 \\
1 & := & 9\ 6\ 2 \mid 4 \mid 0
\end{array}
$$

$$A_\alpha = \langle \{\lambda, \mathtt{var}\}, \mathcal{V}, Q, q_{10}, \mathcal{R}, \delta \rangle$$

$$Q = \{q_1, q_2, q_5, q_6, q_8, q_{10}\}$$

$$\mathcal{R} = \{\mathtt{r}_9, \mathtt{r}_4, \mathtt{r}_0\}$$

Inhabitation

$$
\begin{array}{lcl}
10 & := & \lambda 9.8 \\
8 & := & \lambda 4.5 \mid 9\ 6 \\
6 & := & \lambda 0.1 \mid 9\ 6 \\
5 & := & 9\ 6\ 2 \mid 4 \mid 0 \\
2 & := & 9\ 6 \mid 4 \mid 0 \\
1 & := & 9\ 6\ 2 \mid 4 \mid 0
\end{array}
$$

$$A_\alpha = \langle \{\lambda, \mathtt{var}\}, \mathcal{V}, Q, q_{10}, \mathcal{R}, \delta \rangle$$

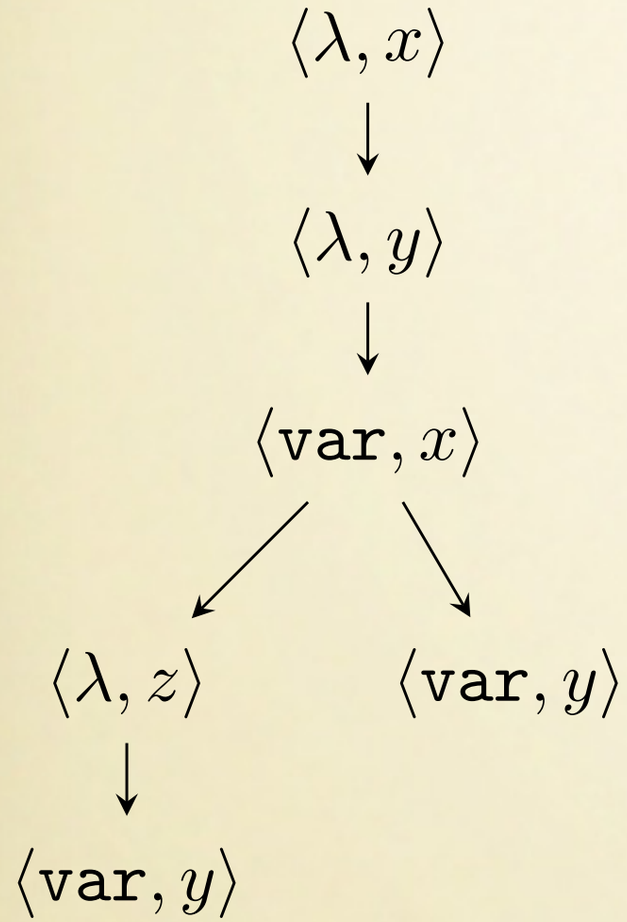
$$Q = \{q_1, q_2, q_5, q_6, q_8, q_{10}\}$$

$$\mathcal{R} = \{\mathtt{r}_9, \mathtt{r}_4, \mathtt{r}_0\}$$

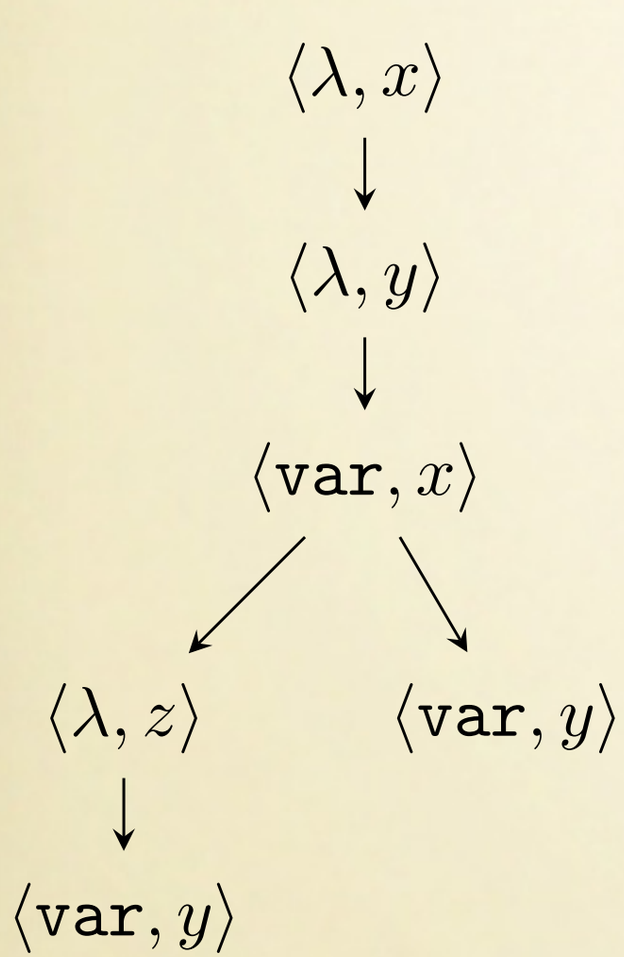
$$
\begin{array}{lcl}
\lambda, q_{10}, \varnothing & \rightsquigarrow & q_8, \mathtt{r}_9 \\
\lambda, q_8, \varnothing & \rightsquigarrow & q_5, \mathtt{r}_4 \\
\mathtt{var}, q_8, \mathtt{r}_9 & \rightsquigarrow & q_6, \varnothing \\
\lambda, q_6, \varnothing & \rightsquigarrow & q_1, \mathtt{r}_0 \\
\mathtt{var}, q_6, \mathtt{r}_9 & \rightsquigarrow & q_6, \varnothing \\
\mathtt{var}, q_5, \mathtt{r}_9 & \rightsquigarrow & q_6, q_2, \varnothing \\
\mathtt{var}, q_5, \mathtt{r}_4 & \rightsquigarrow & \varnothing \\
\mathtt{var}, q_5, \mathtt{r}_0 & \rightsquigarrow & \varnothing \\
\mathtt{var}, q_2, \mathtt{r}_9 & \rightsquigarrow & q_6, q_2, \varnothing \\
\mathtt{var}, q_2, \mathtt{r}_4 & \rightsquigarrow & \varnothing \\
\mathtt{var}, q_2, \mathtt{r}_0 & \rightsquigarrow & \varnothing \\
\mathtt{var}, q_1, \mathtt{r}_9 & \rightsquigarrow & q_6, q_2, \varnothing \\
\mathtt{var}, q_1, \mathtt{r}_4 & \rightsquigarrow & \varnothing \\
\mathtt{var}, q_1, \mathtt{r}_0 & \rightsquigarrow & \varnothing
\end{array}
$$

Inhabitation

Inhabitation

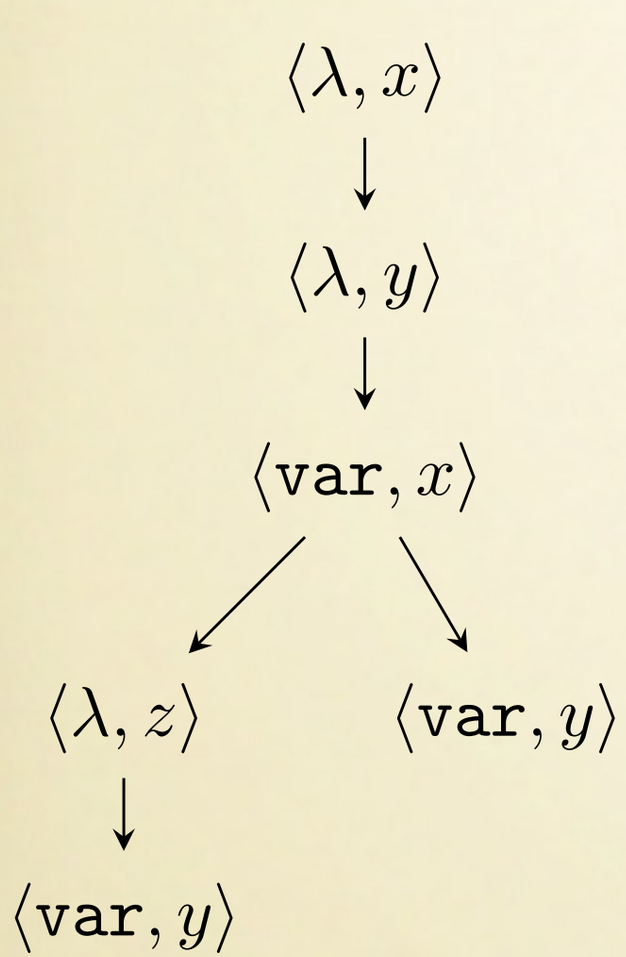


Inhabitation



$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$

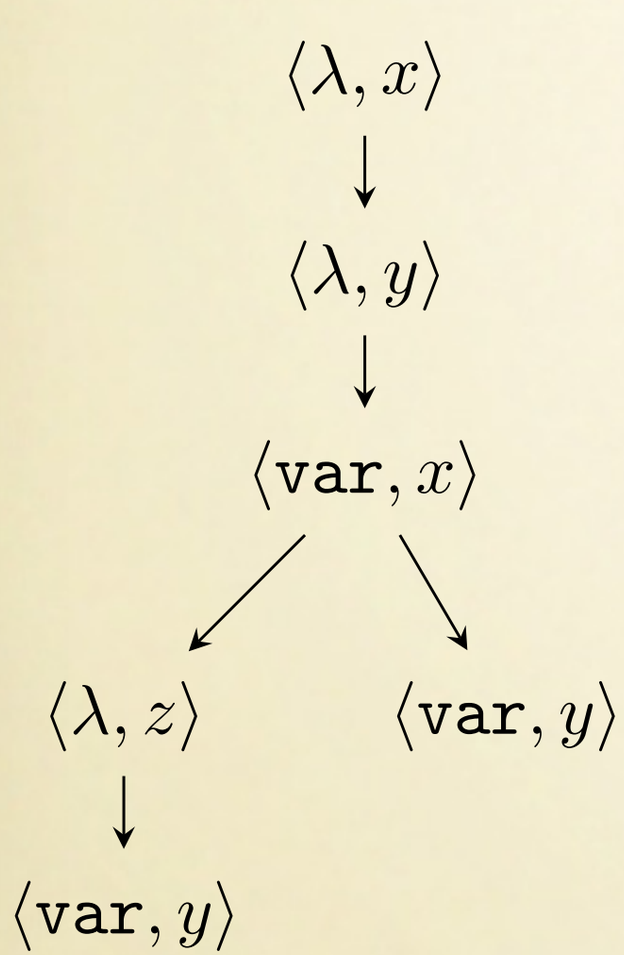
Inhabitation



$$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$$

$$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$$

Inhabitation

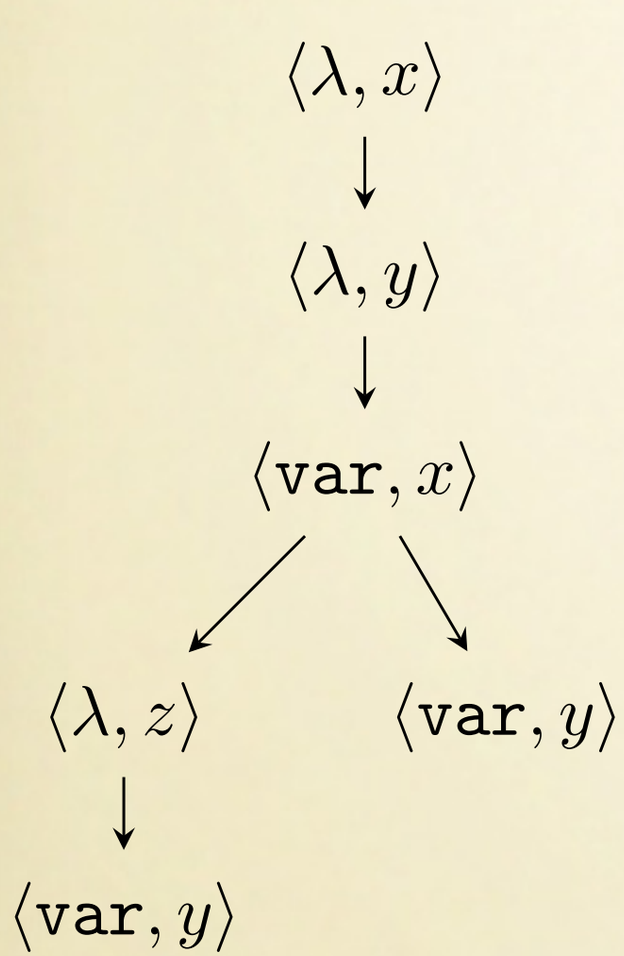


$$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$$

$$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$$

$$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$$

Inhabitation



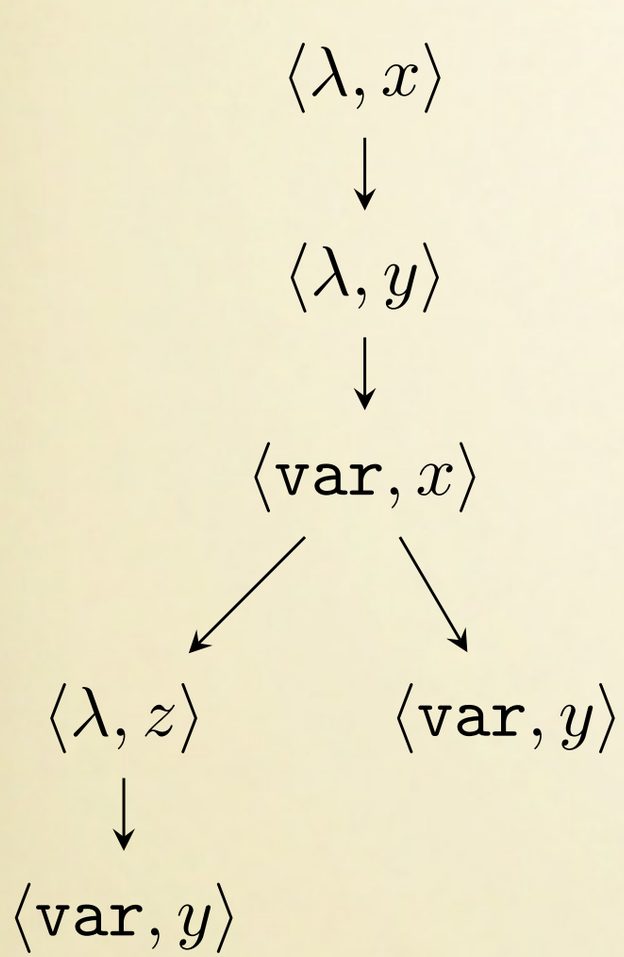
$$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$$

$$\lambda, q_8, \varnothing \rightsquigarrow q_5, \mathtt{r}_4$$

$$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$$

$$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$$

Inhabitation



$$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$$

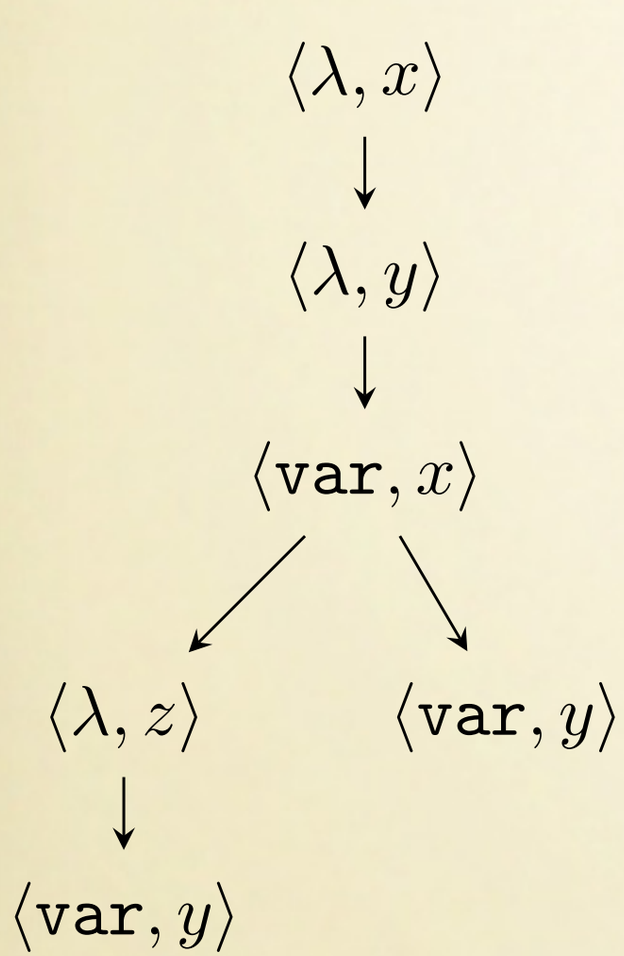
$$\lambda, q_8, \varnothing \rightsquigarrow q_5, \mathtt{r}_4$$

$$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$$

$$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$$

$$(q_5, [\{x\}_9, \{y\}_4, \emptyset_0])$$

Inhabitation



$$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$$

$$\lambda, q_8, \varnothing \rightsquigarrow q_5, \mathtt{r}_4$$

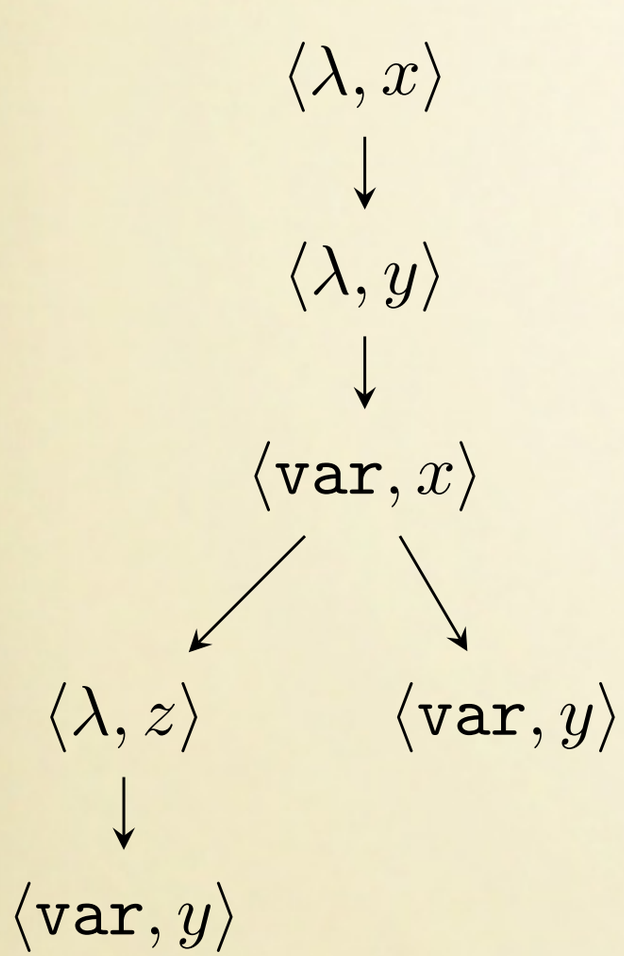
$$\mathtt{var}, q_5, \mathtt{r}_9 \rightsquigarrow q_6, q_2, \varnothing$$

$$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$$

$$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$$

$$(q_5, [\{x\}_9, \{y\}_4, \emptyset_0])$$

Inhabitation



$$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$$

$$\lambda, q_8, \varnothing \rightsquigarrow q_5, \mathtt{r}_4$$

$$\mathtt{var}, q_5, \mathtt{r}_9 \rightsquigarrow q_6, q_2, \varnothing$$

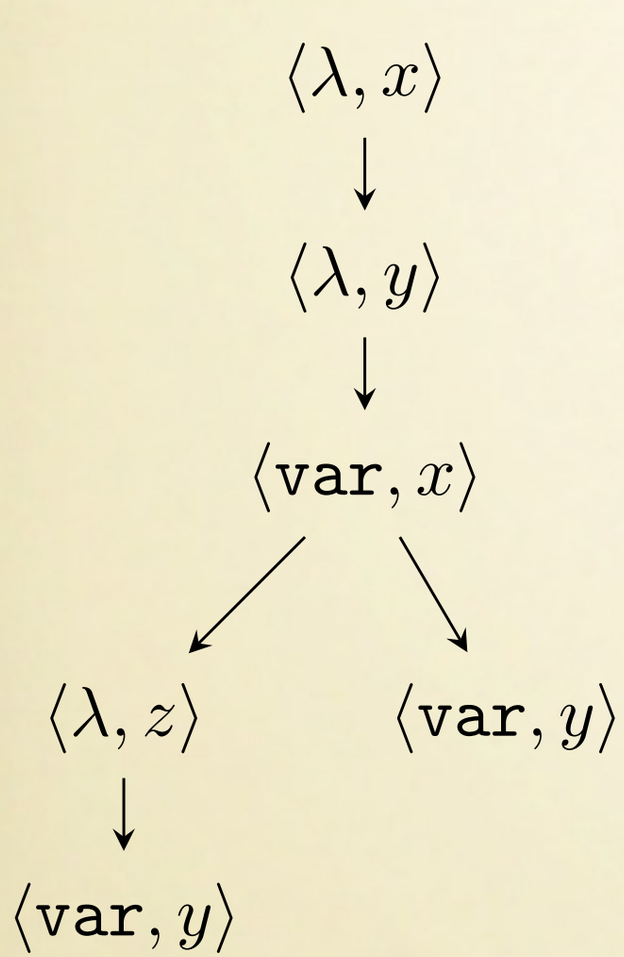
$$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$$

$$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$$

$$(q_5, [\{x\}_9, \{y\}_4, \emptyset_0])$$

$$(q_6, [\{x\}_9, \{y\}_4, \emptyset_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$$

Inhabitation



$$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$$

$$\lambda, q_8, \varnothing \rightsquigarrow q_5, \mathtt{r}_4$$

$$\mathtt{var}, q_5, \mathtt{r}_9 \rightsquigarrow q_6, q_2, \varnothing$$

$$\lambda, q_6, \varnothing \rightsquigarrow q_1, \mathtt{r}_0$$

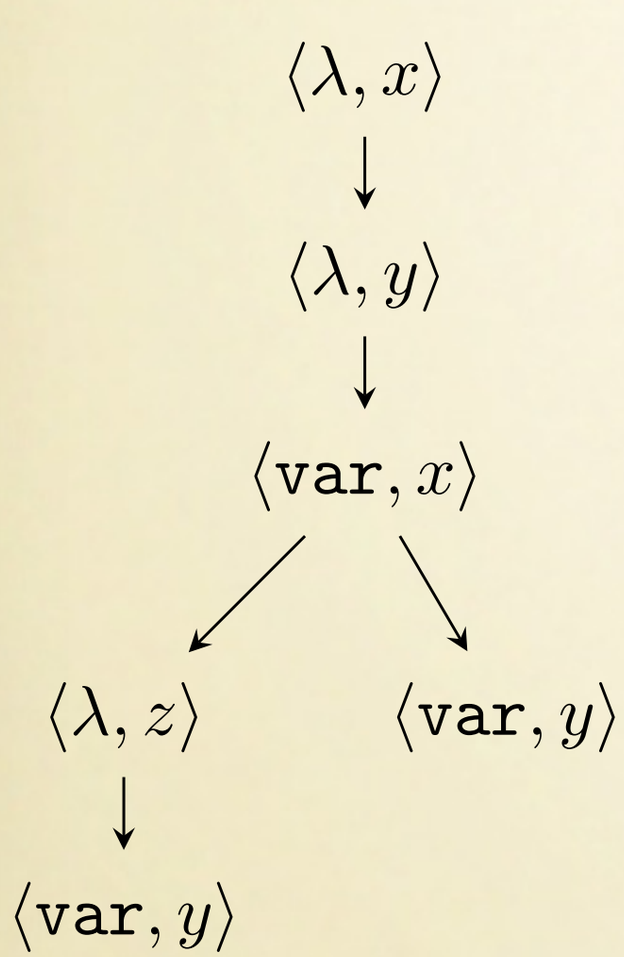
$$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$$

$$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$$

$$(q_5, [\{x\}_9, \{y\}_4, \emptyset_0])$$

$$(q_6, [\{x\}_9, \{y\}_4, \emptyset_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$$

Inhabitation



$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$

$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$

$\lambda, q_8, \varnothing \rightsquigarrow q_5, \mathtt{r}_4$

$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$

$\mathtt{var}, q_5, \mathtt{r}_9 \rightsquigarrow q_6, q_2, \varnothing$

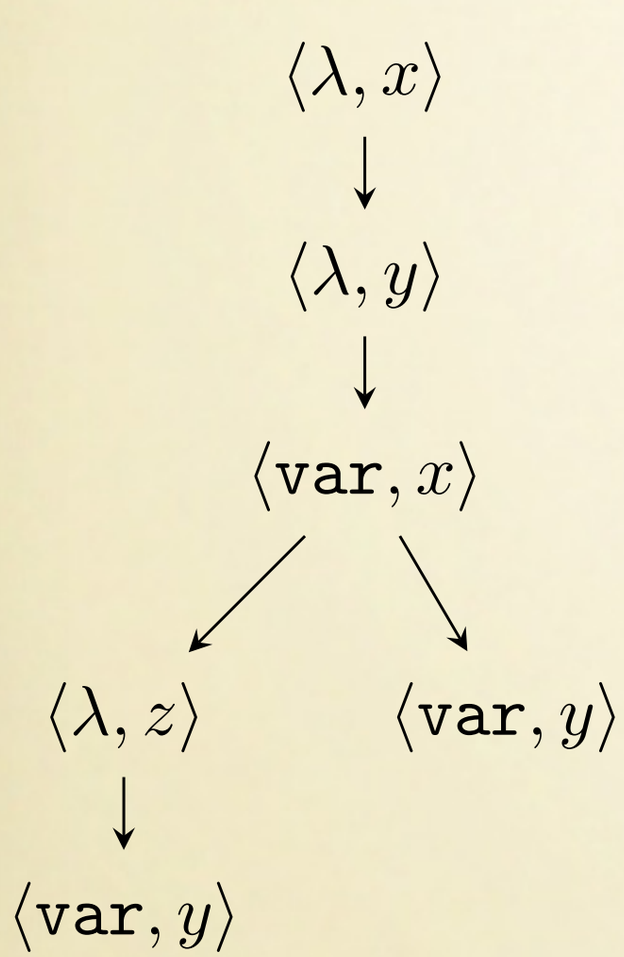
$(q_5, [\{x\}_9, \{y\}_4, \emptyset_0])$

$\lambda, q_6, \varnothing \rightsquigarrow q_1, \mathtt{r}_0$

$(q_6, [\{x\}_9, \{y\}_4, \emptyset_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$

$(q_1, [\{x\}_9, \{y\}_4, \{z\}_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$

Inhabitation



$$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$$

$$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$$

$$\lambda, q_8, \varnothing \rightsquigarrow q_5, \mathtt{r}_4$$

$$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$$

$$\mathtt{var}, q_5, \mathtt{r}_9 \rightsquigarrow q_6, q_2, \varnothing$$

$$(q_5, [\{x\}_9, \{y\}_4, \emptyset_0])$$

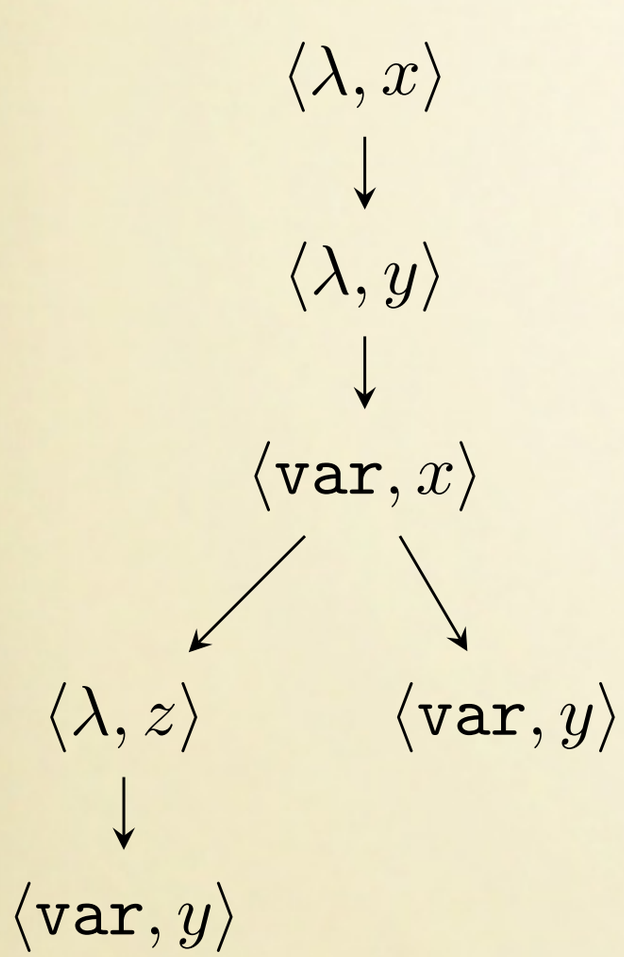
$$\lambda, q_6, \varnothing \rightsquigarrow q_1, \mathtt{r}_0$$

$$(q_6, [\{x\}_9, \{y\}_4, \emptyset_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$$

$$\mathtt{var}, q_1, \mathtt{r}_4 \rightsquigarrow \varnothing$$

$$(q_1, [\{x\}_9, \{y\}_4, \{z\}_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$$

Inhabitation



$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$

$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$

$\lambda, q_8, \varnothing \rightsquigarrow q_5, \mathtt{r}_4$

$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$

$\mathtt{var}, q_5, \mathtt{r}_9 \rightsquigarrow q_6, q_2, \varnothing$

$(q_5, [\{x\}_9, \{y\}_4, \emptyset_0])$

$\lambda, q_6, \varnothing \rightsquigarrow q_1, \mathtt{r}_0$

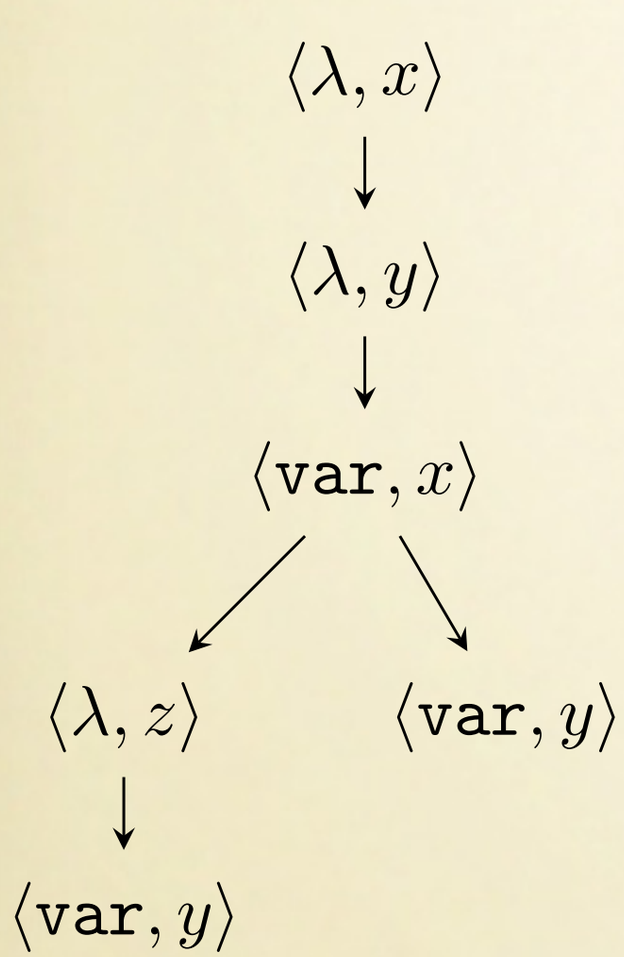
$(q_6, [\{x\}_9, \{y\}_4, \emptyset_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$

$\mathtt{var}, q_1, \mathtt{r}_4 \rightsquigarrow \varnothing$

$(q_1, [\{x\}_9, \{y\}_4, \{z\}_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$

$(q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$

Inhabitation



$$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$$

$$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$$

$$\lambda, q_8, \varnothing \rightsquigarrow q_5, \mathtt{r}_4$$

$$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$$

$$\mathtt{var}, q_5, \mathtt{r}_9 \rightsquigarrow q_6, q_2, \varnothing$$

$$(q_5, [\{x\}_9, \{y\}_4, \emptyset_0])$$

$$\lambda, q_6, \varnothing \rightsquigarrow q_1, \mathtt{r}_0$$

$$(q_6, [\{x\}_9, \{y\}_4, \emptyset_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$$

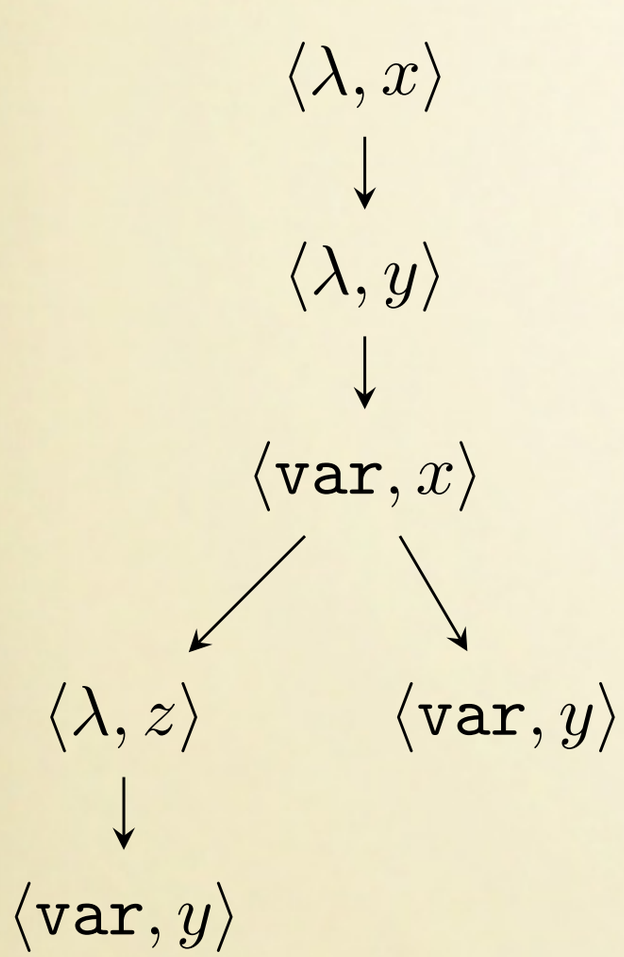
$$\mathtt{var}, q_1, \mathtt{r}_4 \rightsquigarrow \varnothing$$

$$(q_1, [\{x\}_9, \{y\}_4, \{z\}_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$$

$$\mathtt{var}, q_2, \mathtt{r}_4 \rightsquigarrow \varnothing$$

$$(q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$$

Inhabitation



$\lambda, q_{10}, \varnothing \rightsquigarrow q_8, \mathtt{r}_9$

$(q_{10}, [\emptyset_9, \emptyset_4, \emptyset_0])$

$\lambda, q_8, \varnothing \rightsquigarrow q_5, \mathtt{r}_4$

$(q_8, [\{x\}_9, \emptyset_4, \emptyset_0])$

$\mathtt{var}, q_5, \mathtt{r}_9 \rightsquigarrow q_6, q_2, \varnothing$

$(q_5, [\{x\}_9, \{y\}_4, \emptyset_0])$

$\lambda, q_6, \varnothing \rightsquigarrow q_1, \mathtt{r}_0$

$(q_6, [\{x\}_9, \{y\}_4, \emptyset_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$

$\mathtt{var}, q_1, \mathtt{r}_4 \rightsquigarrow \varnothing$

$(q_1, [\{x\}_9, \{y\}_4, \{z\}_0]), (q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$

$\mathtt{var}, q_2, \mathtt{r}_4 \rightsquigarrow \varnothing$

$(q_2, [\{x\}_9, \{y\}_4, \emptyset_0])$

ε

Inhabitation

Inhabitation

Proposition Given $\alpha \in \mathcal{T}$ and a normal term M,
the inhabitation machine $\mathcal{A}_\alpha$ operates in a deterministic way on $\mathbf{t}^M$.

Inhabitation

Proposition Given $\alpha \in \mathcal{T}$ and a normal term M,
the inhabitation machine $\mathcal{A}_\alpha$ operates in a deterministic way on $\mathbf{t}^M$.

Theorem $\mathcal{L}(\mathcal{A}_\alpha)$ is the set of closed normal inhabitants of α.

Principal Inhabitants

Principal Inhabitants

- $M \in \Lambda(\alpha)$ is called a **principal inhabitant** of α if every other type inhabited by M is an instance of α, i.e. α is the most general type that can be assigned to M

Principal Inhabitants

- $M \in \Lambda(\alpha)$ is called a **principal inhabitant** of α if every other type inhabited by M is an instance of α, i.e. α is the most general type that can be assigned to M

- $\lambda x.x$ inhabits $(a \to b) \to a \to b$, but is no principal inhabitant of this type

Principal Inhabitants

- $M \in \Lambda(\alpha)$ is called a **principal inhabitant** of α if every other type inhabited by M is an instance of α, i.e. α is the most general type that can be assigned to M

- $\lambda x.x$ inhabits $(a \to b) \to a \to b$, but is no principal inhabitant of this type

- $\lambda x.x$ is a principal inhabitant of $a \to a$

Principal Inhabitants

- $M \in \Lambda(\alpha)$ is called a **principal inhabitant** of α if every other type inhabited by M is an instance of α, i.e. α is the most general type that can be assigned to M

- $\lambda x.x$ inhabits $(a \to b) \to a \to b$, but is no principal inhabitant of this type

- $\lambda x.x$ is a principal inhabitant of $a \to a$

- $\lambda xy.xy$ inhabits $(a \to a) \to a \to a$, but is no principal inhabitant of this type

Principal Inhabitants

- $M \in \Lambda(\alpha)$ is called a **principal inhabitant** of α if every other type inhabited by M is an instance of α, i.e. α is the most general type that can be assigned to M

- $\lambda x.x$ inhabits $(a \to b) \to a \to b$, but is no principal inhabitant of this type

- $\lambda x.x$ is a principal inhabitant of $a \to a$

- $\lambda xy.xy$ inhabits $(a \to a) \to a \to a$, but is no principal inhabitant of this type

- $\lambda xy.xy$ is a principal inhabitant of $(a \to b) \to a \to b$

Principal Inhabitants

Principal Inhabitants

- when searching for principal inhabitants one usually focusses on long inhabitants

Principal Inhabitants

- when searching for principal inhabitants one usually focusses on long inhabitants
- a term is called **long** if every variable in function position is given as many arguments as allowed by its type

Principal Inhabitants

- when searching for principal inhabitants one usually focusses on long inhabitants
- a term is called **long** if every variable in function position is given as many arguments as allowed by its type
- pre-grammars can be easily changed to consider exactly the long normal inhabitants of a type

Principal Inhabitants

- when searching for principal inhabitants one usually focusses on long inhabitants
- a term is called **long** if every variable in function position is given as many arguments as allowed by its type
- pre-grammars can be easily changed to consider exactly the long normal inhabitants of a type
- for principal inhabitants the inhabitation machines are supplied with two additional global registers

Principal Inhabitants

- when searching for principal inhabitants one usually focusses on long inhabitants
- a term is called **long** if every variable in function position is given as many arguments as allowed by its type
- pre-grammars can be easily changed to consider exactly the long normal inhabitants of a type
- for principal inhabitants the inhabitation machines are supplied with two additional global registers
- register `id` keeps track of which occurrences of type variables have to be the same

Principal Inhabitants

- when searching for principal inhabitants one usually focusses on long inhabitants
- a term is called **long** if every variable in function position is given as many arguments as allowed by its type
- pre-grammars can be easily changed to consider exactly the long normal inhabitants of a type
- for principal inhabitants the inhabitation machines are supplied with two additional global registers
- register `id` keeps track of which occurrences of type variables have to be the same
- register `needs` keeps track of which occurrences of compound subtypes have to have the given structure

Principal Inhabitants

Principal Inhabitants

Proposition Given type α and a normal term M, then M is a long inhabitant of α if and only if $\mathcal{P}_\alpha$ accepts M with some output $(\texttt{id}', \texttt{needs}')$.

Principal Inhabitants

Proposition Given type α and a normal term M, then M is a long inhabitant of α if and only if $\mathcal{P}_\alpha$ accepts M with some output $(\mathtt{id}', \mathtt{needs}')$.

Definition
$\mathtt{raise}(\alpha, \mathtt{id}, \mathtt{needs})$ denotes the type obtained from α as follows.

- Every occurrence of a subtype β with $\mathtt{n}(\beta) \in \mathtt{needs}$ is substituted by a fresh type variable;
- all occurrences of type variables are given fresh names, in such a way that two occurrences receive the same name if and only if they belong to the same class in $\mathtt{id}$.

Principal Inhabitants

Principal Inhabitants

Proposition If $\mathcal{P}_\alpha$ accepts M with output $(\texttt{id}, \texttt{needs})$, then $\texttt{raise}(\alpha, \texttt{id}, \texttt{needs})$ is the principal type of M.

Principal Inhabitants

Proposition If $\mathcal{P}_\alpha$ accepts M with output $(\texttt{id}, \texttt{needs})$, then $\texttt{raise}(\alpha, \texttt{id}, \texttt{needs})$ is the principal type of M.

Corollary M is a principal long normal inhabitant of α if and only if $\mathcal{P}_\alpha$ accepts M with some output $(\texttt{id}, \emptyset)$ such that `id` contains one class for each type variable in α, each class containing exactly the identifiers of all occurrences of that type variable.

Principal Inhabitants

Principal Inhabitants

$$((o_0 \to o_1) \to o_2 \to o_3)_9 \to o_4 \to o_5$$

Principal Inhabitants

$$((o_0 \to o_1) \to o_2 \to o_3)_9 \to o_4 \to o_5$$

$\lambda xy.x(\lambda z.y)y$

Principal Inhabitants

$$((o_0 \to o_1) \to o_2 \to o_3)_9 \to o_4 \to o_5$$

$\lambda xy.x(\lambda z.y)y$ $\qquad$ $(\{5 = 3, 1 = 2 = 4\}, \emptyset)$

Principal Inhabitants

$$((o_0 \to o_1) \to o_2 \to o_3)_9 \to o_4 \to o_5$$

$\lambda xy.x(\lambda z.y)y$ $\quad$ $(\{5 = 3, 1 = 2 = 4\}, \emptyset)$

$$((o \to o') \to o' \to o'') \to o' \to o''$$

Principal Inhabitants

$$((o_0 \to o_1) \to o_2 \to o_3)_9 \to o_4 \to o_5$$

$\lambda xy.x(\lambda z.y)y$ $\quad$ $(\{5 = 3, 1 = 2 = 4\}, \emptyset)$

$$((o \to o') \to o' \to o'') \to o' \to o''$$

$\lambda xy.y$

Principal Inhabitants

$$((o_0 \to o_1) \to o_2 \to o_3)_9 \to o_4 \to o_5$$

$\lambda xy.x(\lambda z.y)y$ $\quad$ $(\{5 = 3, 1 = 2 = 4\}, \emptyset)$

$$((o \to o') \to o' \to o'') \to o' \to o''$$

$\lambda xy.y$ $\quad$ $(\{5 = 4\}; \{9\})$

Principal Inhabitants

$$((o_0 \to o_1) \to o_2 \to o_3)_9 \to o_4 \to o_5$$

$\lambda xy.x(\lambda z.y)y$ $\quad$ $(\{5 = 3, 1 = 2 = 4\}, \emptyset)$ $\quad$ $((o \to o') \to o' \to o'') \to o' \to o''$

$\lambda xy.y$ $\quad$ $(\{5 = 4\}; \{9\})$ $\quad$ $o' \to o \to o$

Principal Inhabitants

$$((o_0 \to o_1) \to o_2 \to o_3)_9 \to o_4 \to o_5$$

$\lambda xy.x(\lambda z.y)y$ $\quad$ $(\{5 = 3, 1 = 2 = 4\}, \emptyset)$ $\quad$ $((o \to o') \to o' \to o'') \to o' \to o''$

$\lambda xy.y$ $\quad$ $(\{5 = 4\}; \{9\})$ $\quad$ $o' \to o \to o$

$\lambda xy.x(\lambda z.z)(x(\lambda u.y)y)$

Principal Inhabitants

$$((o_0 \to o_1) \to o_2 \to o_3)_9 \to o_4 \to o_5$$

$\lambda xy.x(\lambda z.y)y$ $\quad$ $(\{5 = 3, 1 = 2 = 4\}, \emptyset)$ $\quad$ $((o \to o') \to o' \to o'') \to o' \to o''$

$\lambda xy.y$ $\quad$ $(\{5 = 4\}; \{9\})$ $\quad$ $o' \to o \to o$

$\lambda xy.x(\lambda z.z)(x(\lambda u.y)y)$ $\quad$ $(\{0 = 1 = 2 = 3 = 4 = 5\}; \emptyset)$

Principal Inhabitants

$$((o_0 \to o_1) \to o_2 \to o_3)_9 \to o_4 \to o_5$$

$\lambda xy.x(\lambda z.y)y$ $\quad$ $(\{5 = 3, 1 = 2 = 4\}, \emptyset)$

$$((o \to o') \to o' \to o'') \to o' \to o''$$

$\lambda xy.y$ $\quad$ $(\{5 = 4\}; \{9\})$ $\quad$ $o' \to o \to o$

$\lambda xy.x(\lambda z.z)(x(\lambda u.y)y)$ $\quad$ $(\{0 = 1 = 2 = 3 = 4 = 5\}; \emptyset)$

$$((o \to o) \to o \to o) \to o \to o$$

Conclusions

Conclusions

- Definition of inhabitation machines based on pre-grammars;

Conclusions

- Definition of inhabitation machines based on pre-grammars;
- work deterministically recognizing all normal inhabitants of a type;

Conclusions

- Definition of inhabitation machines based on pre-grammars;
- work deterministically recognizing all normal inhabitants of a type;
- extend the definition of the machines in order to deal with principal inhabitation;

Conclusions

- Definition of inhabitation machines based on pre-grammars;
- work deterministically recognizing all normal inhabitants of a type;
- extend the definition of the machines in order to deal with principal inhabitation;
- closure properties;

Conclusions

- Definition of inhabitation machines based on pre-grammars;
- work deterministically recognizing all normal inhabitants of a type;
- extend the definition of the machines in order to deal with principal inhabitation;
- closure properties;
- complexity of the principal inhabitation problem.

Thank You!