A Verified Cost Model for Call-by-Push-Value

ITP2025

Zhuo Zoey Chen, Johannes Åman Pohjola, Christine Rizkallah

A Verified Cost Model for Call-by-Push-Value

ITP2025

Zhuo Zoey Chen, Johannes Åman Pohjola, Christine Rizkallah

A Verified Cost Model for Call-by-Push-Value

ITP2025

Zhuo Zoey Chen, Johannes Åman Pohjola, Christine Rizkallah

A Verified Cost Model for Call-by-Push-Value

ITP2025

Zhuo Zoey Chen, Johannes Åman Pohjola, Christine Rizkallah

A Verified Cost Model for Call-by-Push-Value

ITP2025

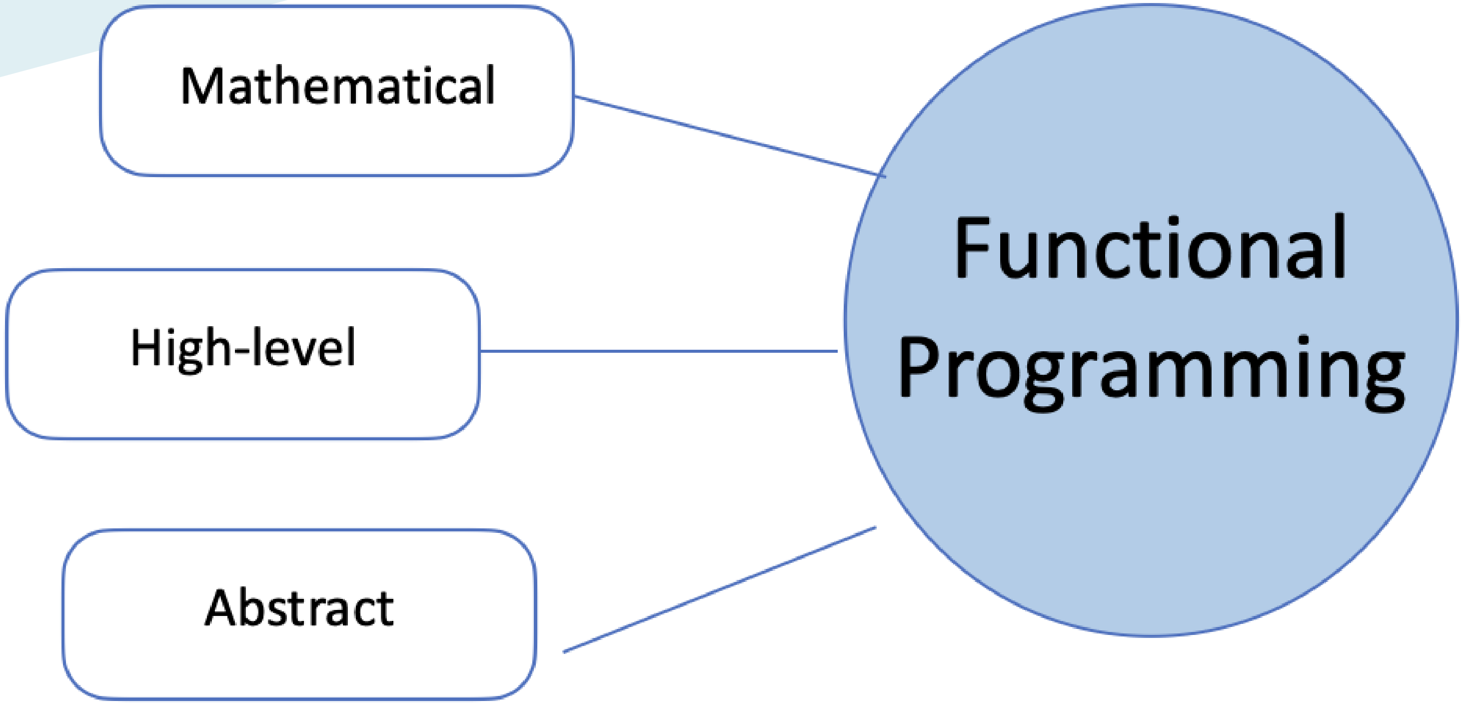
Zhuo Zoey Chen, Johannes Åman Pohjola, Christine Rizkallah

A Verified Cost Model for Call-by-Push-Value

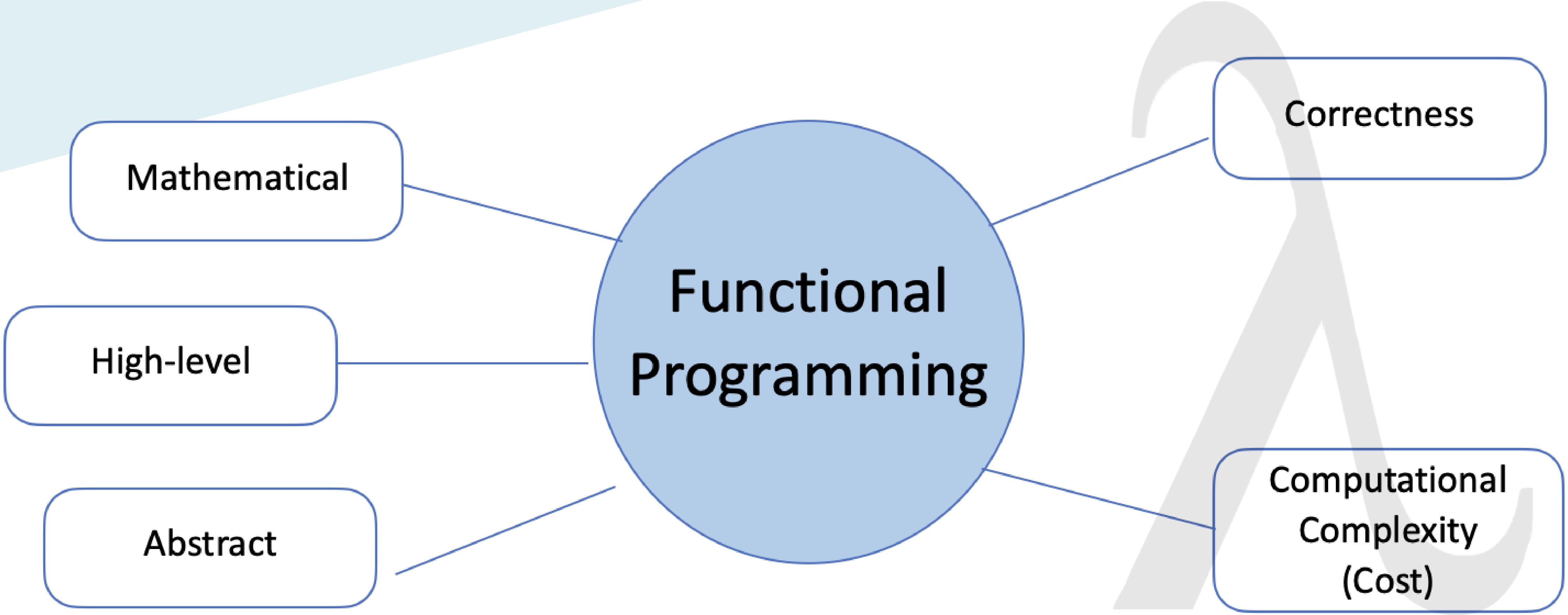
ITP2025

Zhuo Zoey Chen, Johannes Åman Pohjola, Christine Rizkallah

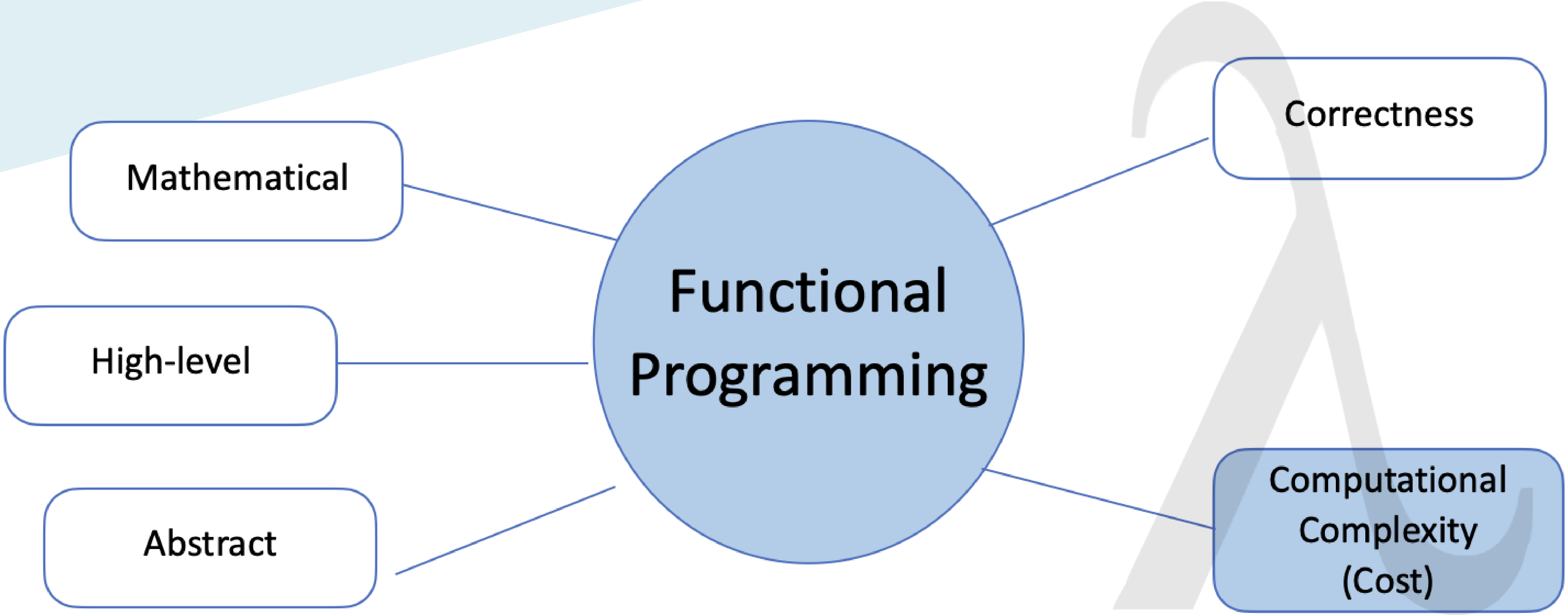
Problem and Motivation



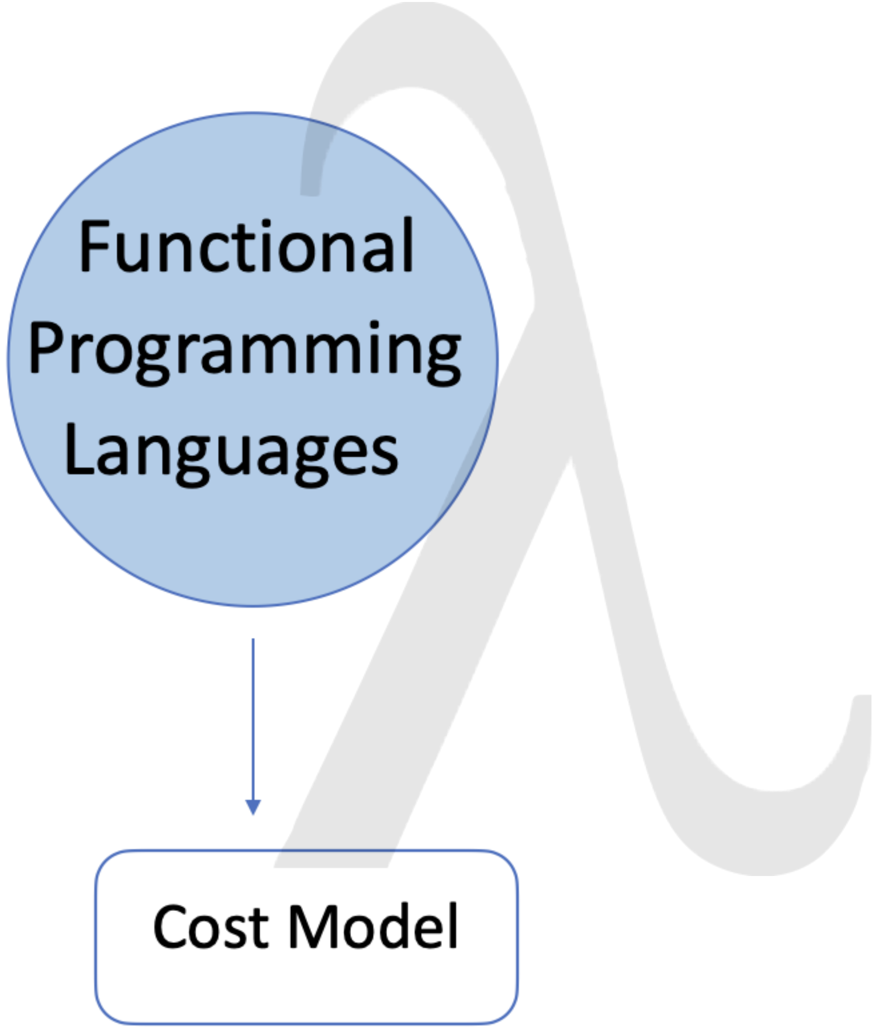
Problem and Motivation



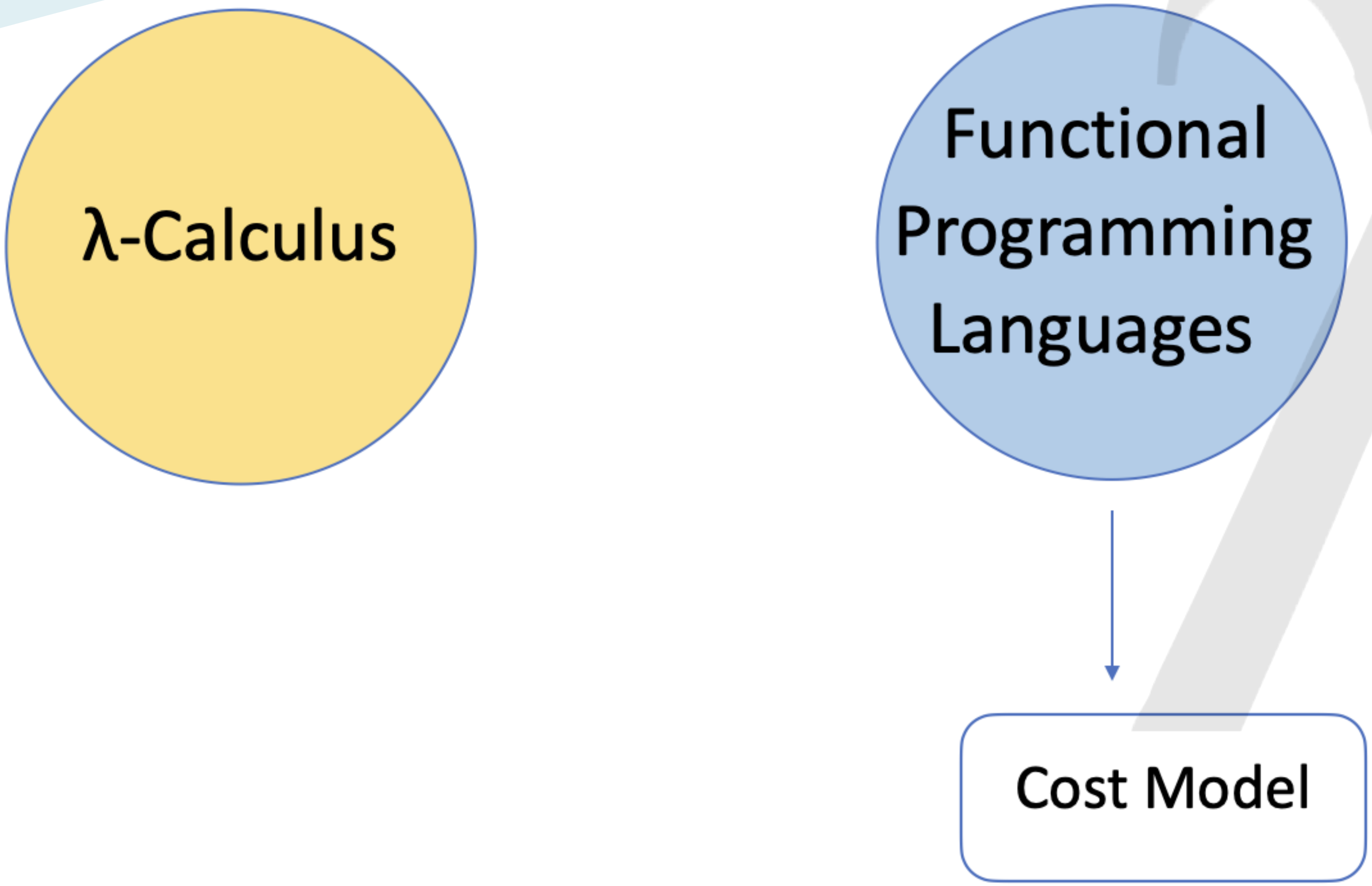
Problem and Motivation



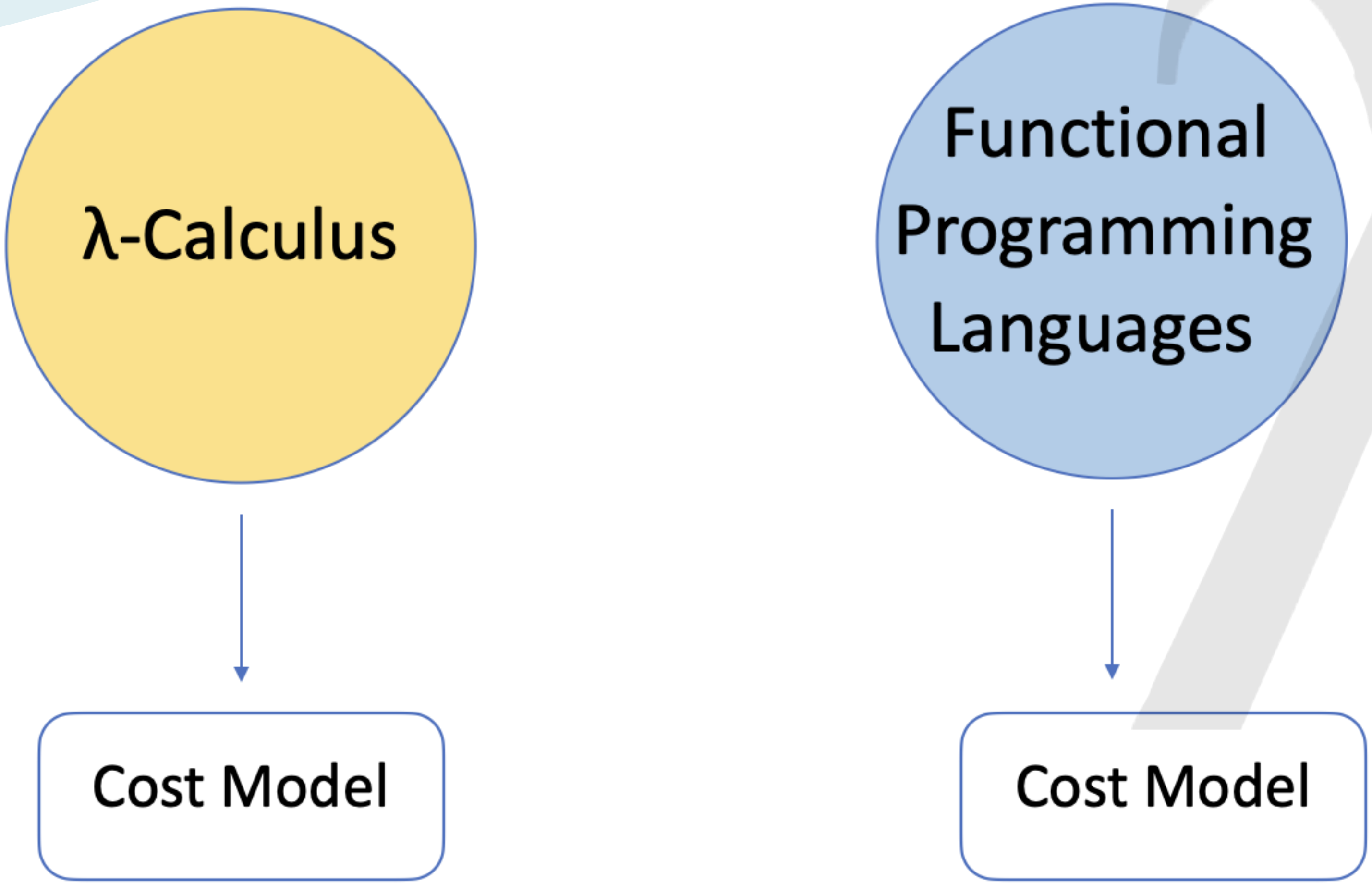
Problem and Motivation



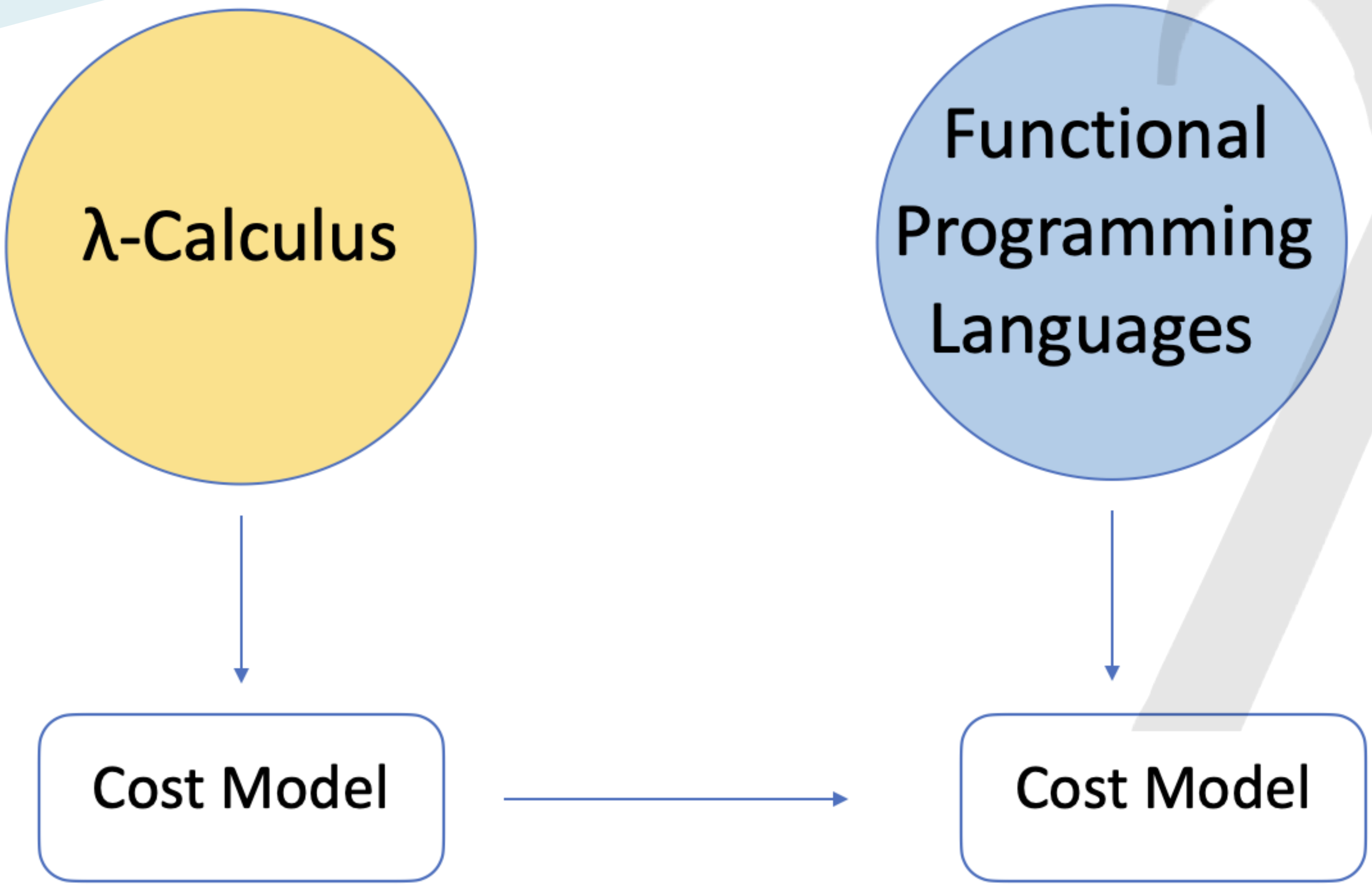
Problem and Motivation



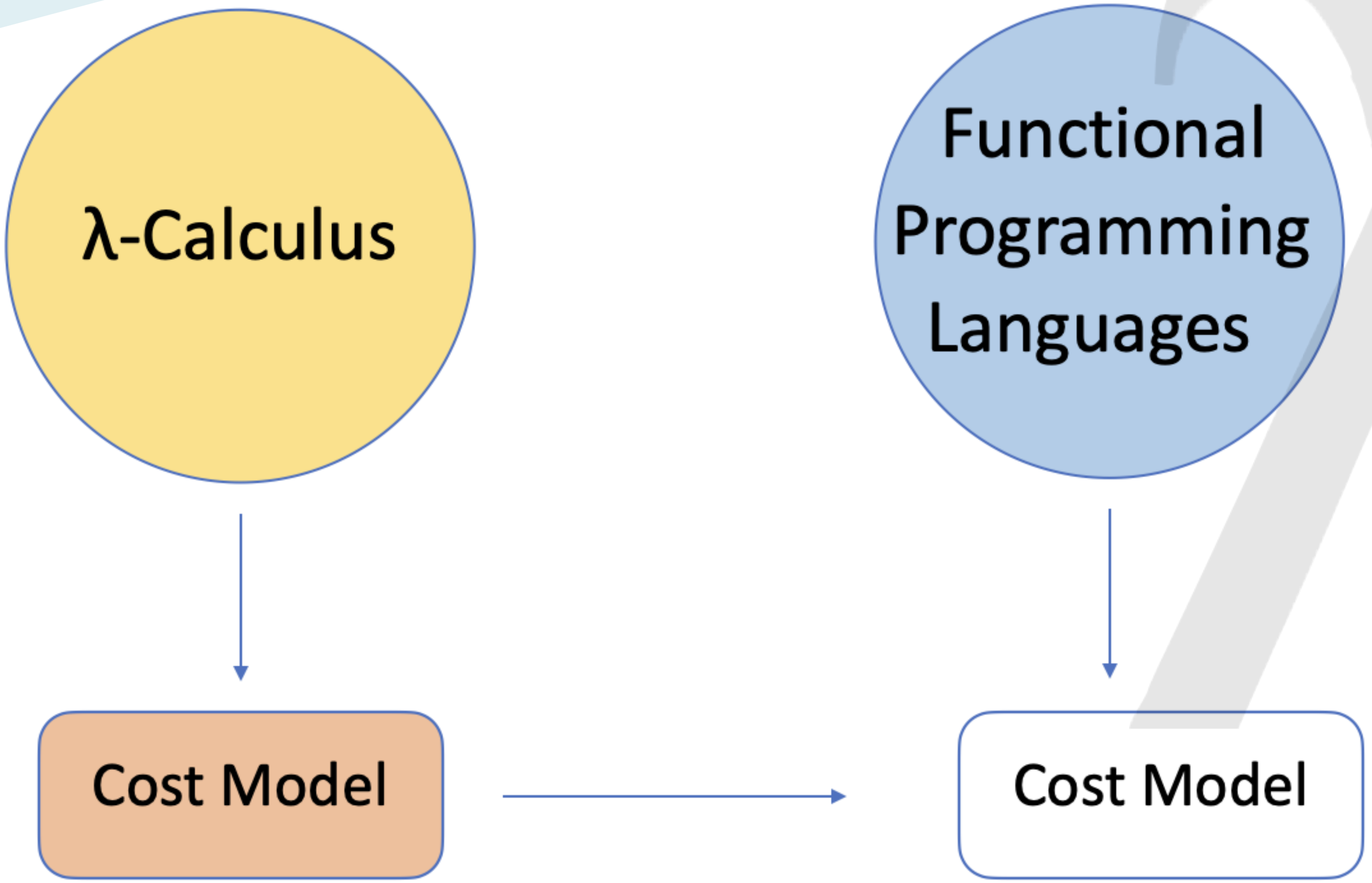
Problem and Motivation



Problem and Motivation



Problem and Motivation



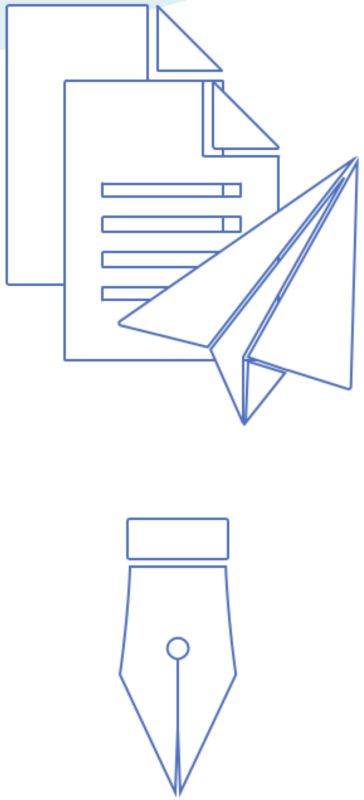
THE UNIVERSITY OF MELBOURNE

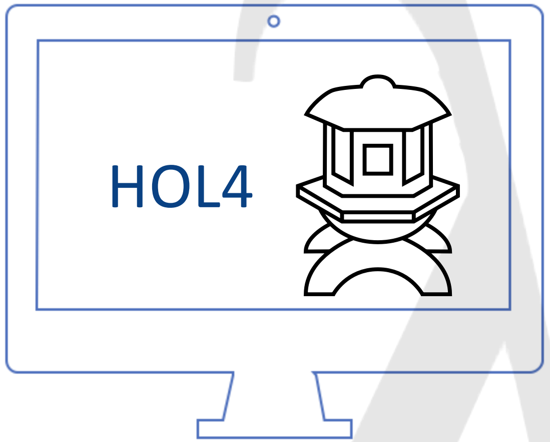
A Verified Cost Model for Call-by-Push-Value

ITP2025

Zhuo Zoey Chen, Johannes Åman Pohjola, Christine Rizkallah

Formal Verification





Interactive Theorem Prover

A Verified Cost Model for Call-by-Push-Value

ITP2025

Zhuo Zoey Chen, Johannes Åman Pohjola, Christine Rizkallah

Background – Lambda Calculus

λ-expression $s, t := x \mid (\lambda x.\, s) \mid (s\, t)$

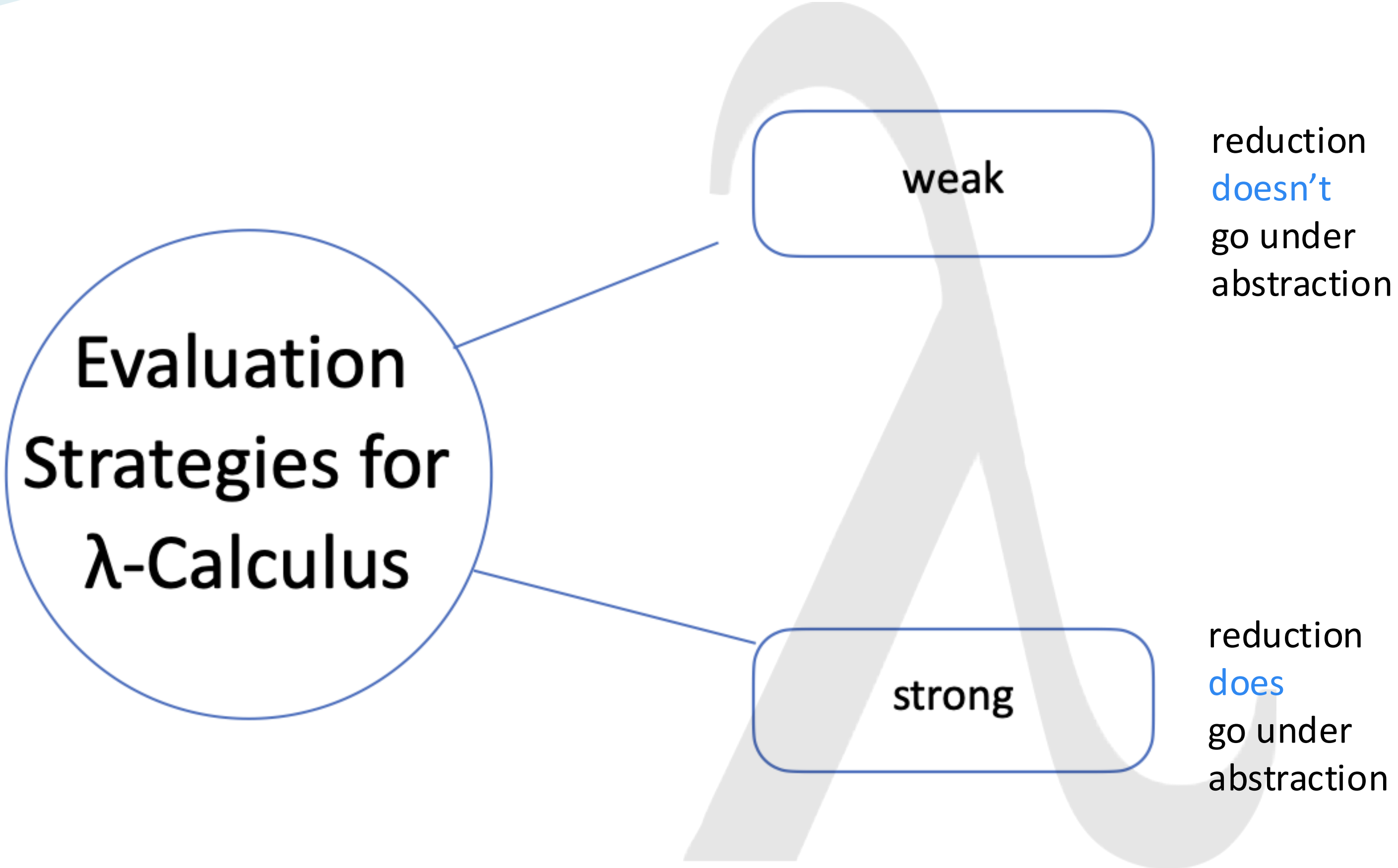
(β-reduction). $(\lambda x.s)\, t \to_\beta s[x := t]$

(Substitution). $s[x := t]$

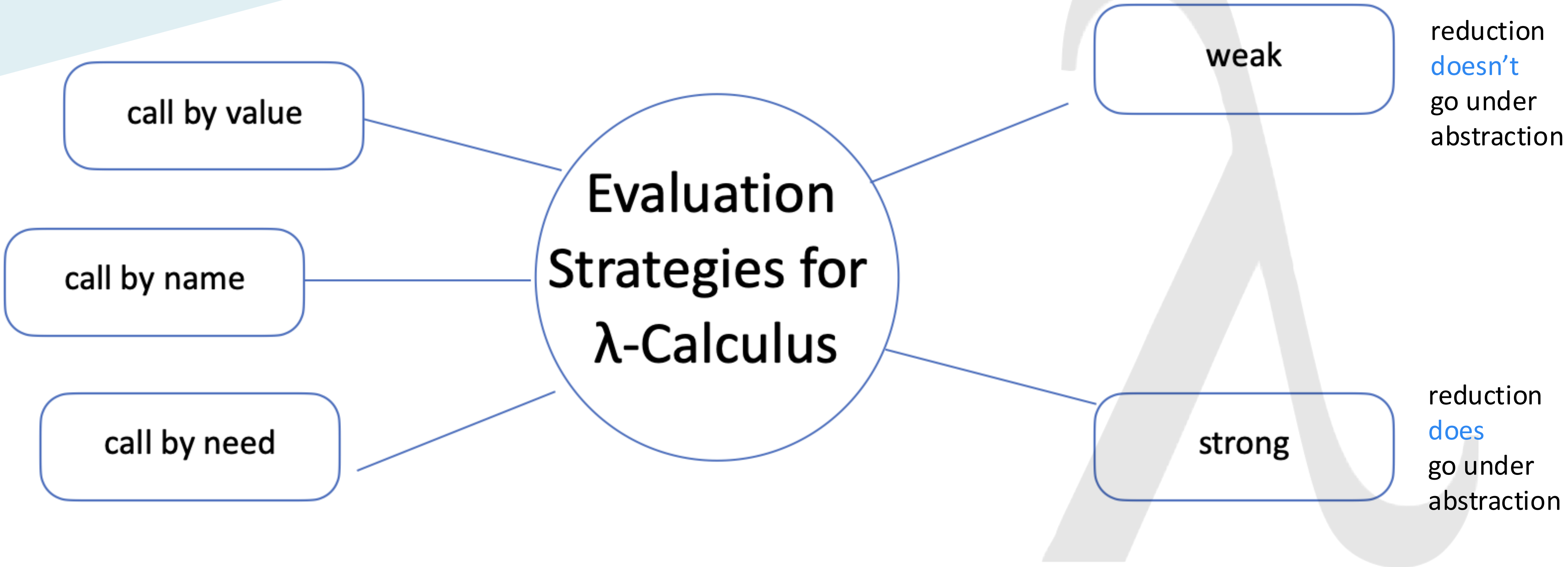
Background – Lambda Calculus

Evaluation Strategies for λ-Calculus

Background – Lambda Calculus



Background – Lambda Calculus



Background – Lambda Calculus

Evaluates the function argument before passing them into the function call

call by value

call by name

call by need

Evaluation Strategies for λ-Calculus

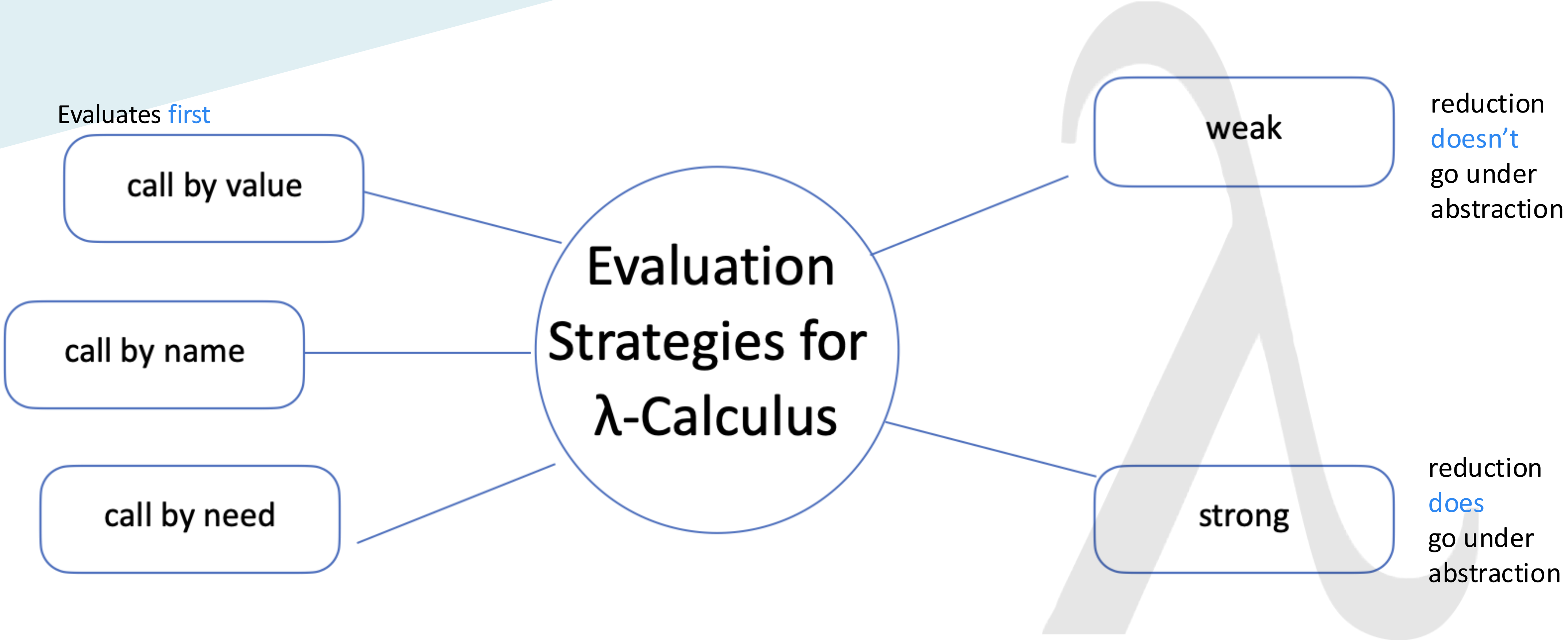
weak

reduction doesn't go under abstraction

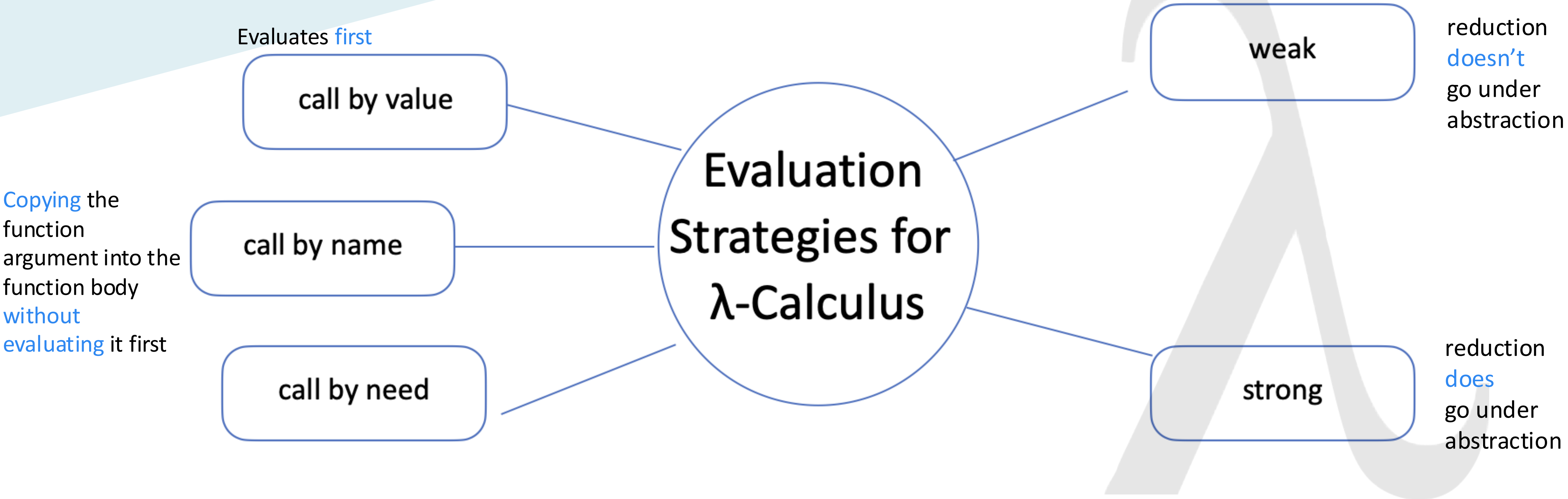
strong

reduction does go under abstraction

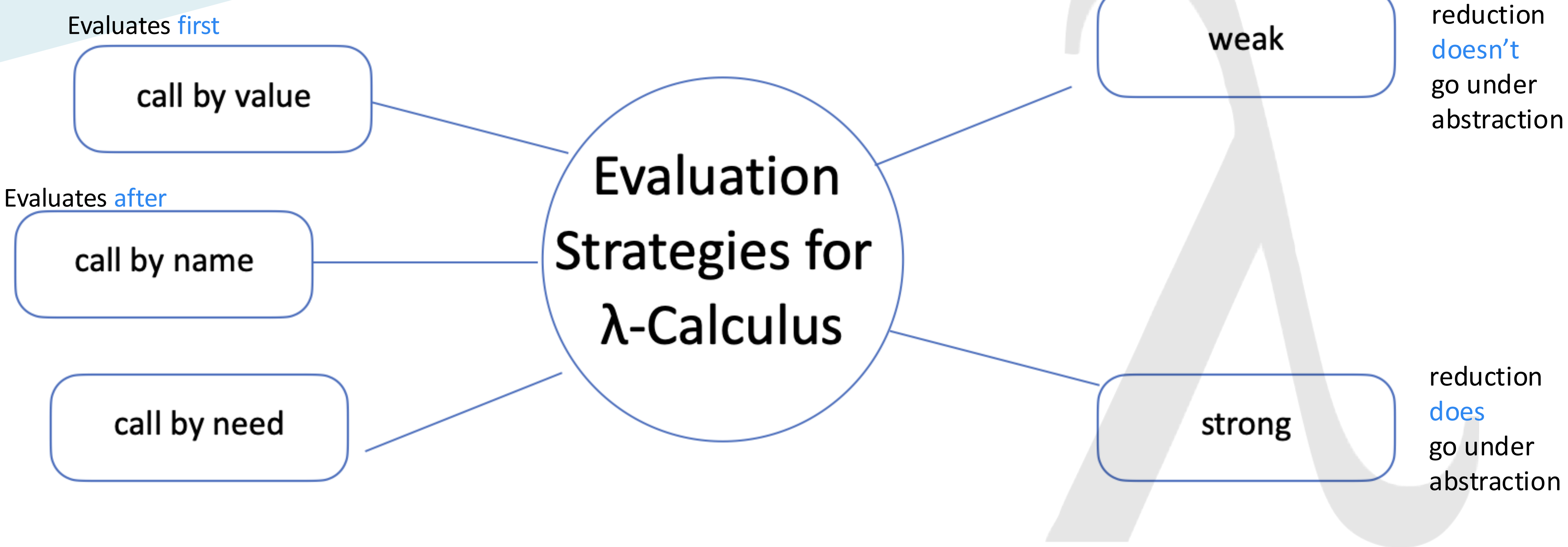
Background – Lambda Calculus



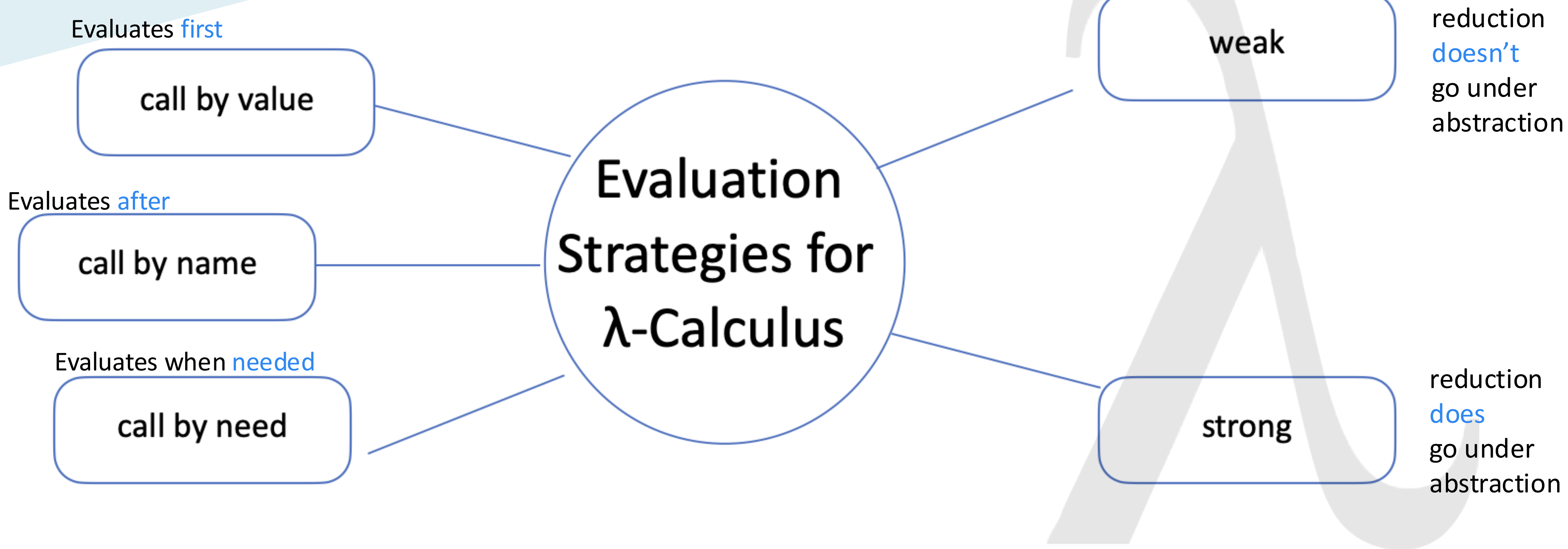
Background – Lambda Calculus



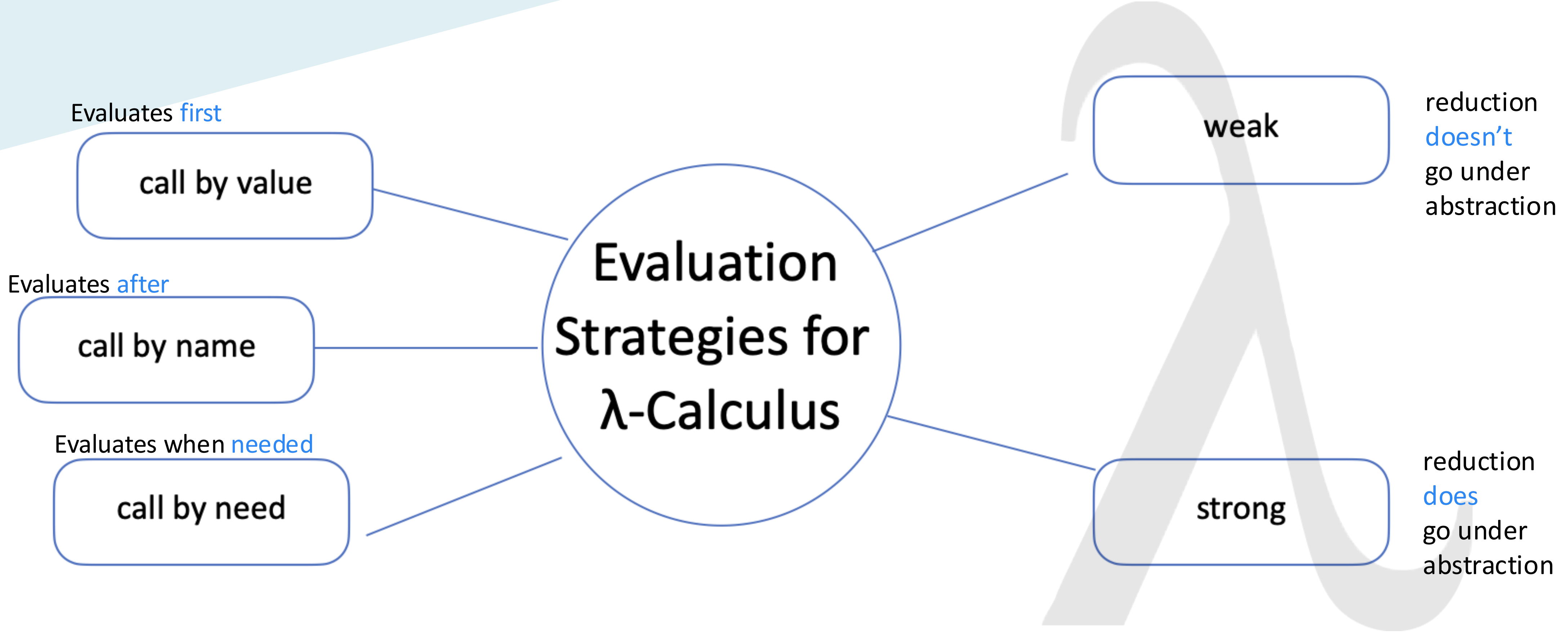
Background – Lambda Calculus



Background – Lambda Calculus



Background – Lambda Calculus



Need different cost models!

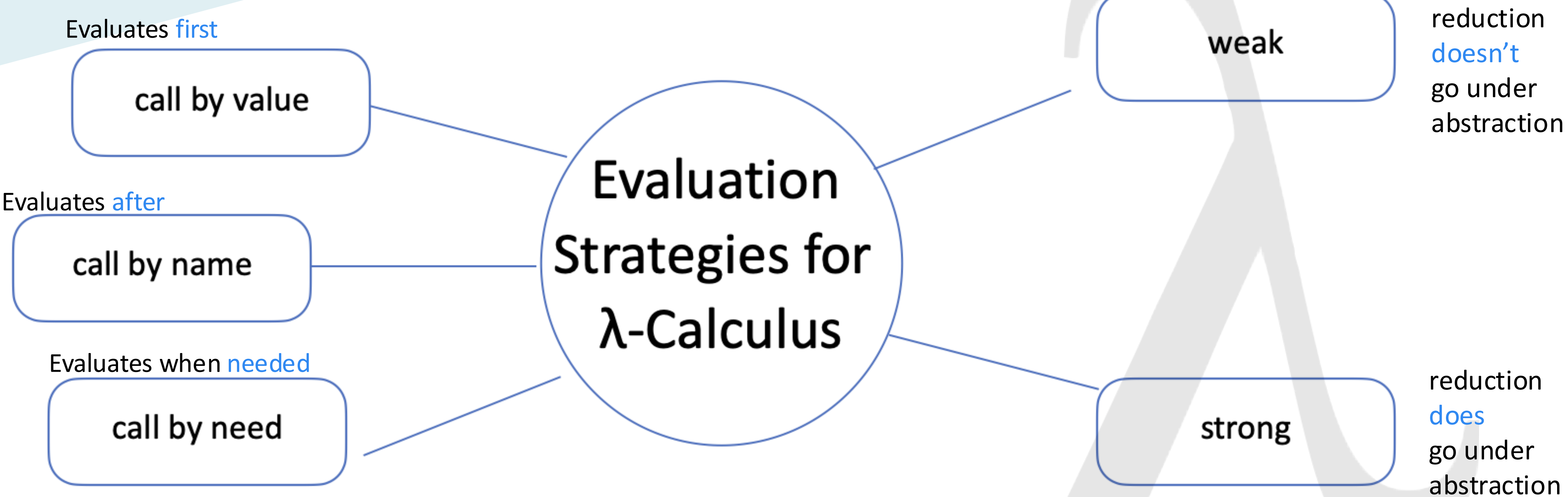
THE UNIVERSITY OF MELBOURNE

A Verified Cost Model for Call-by-Push-Value

ITP2025

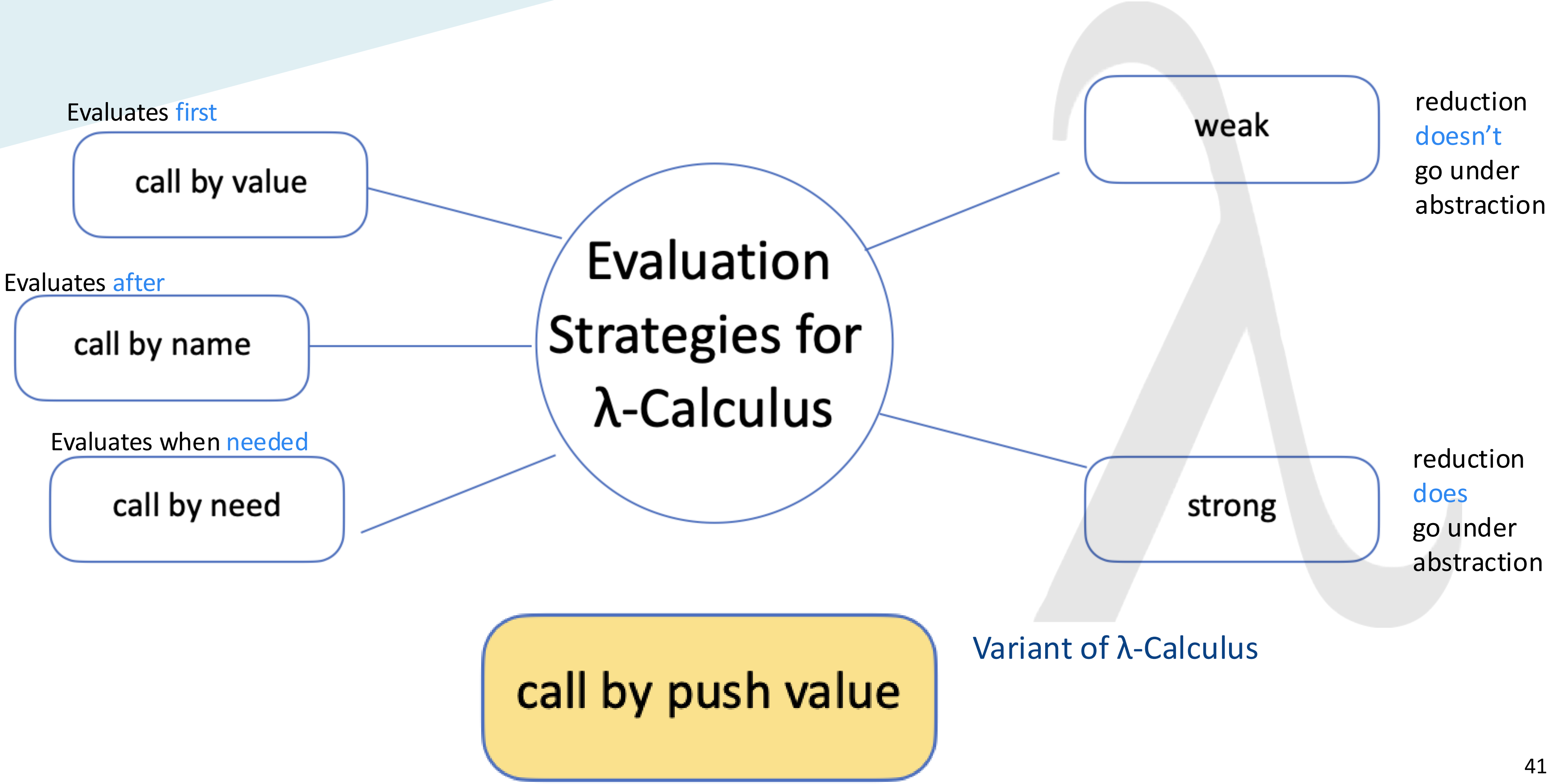
Zhuo Zoey Chen, Johannes Åman Pohjola, Christine Rizkallah

Background – CBPV [Levy, 1999]

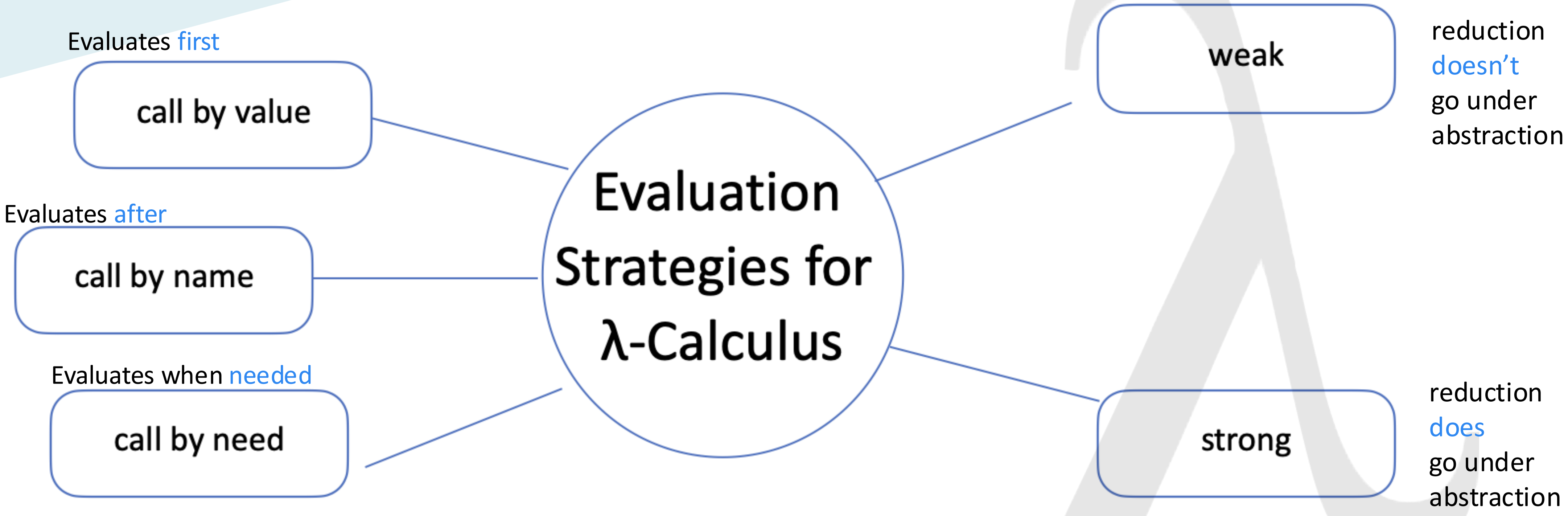


call by push value

Background – CBPV



Background – CBPV

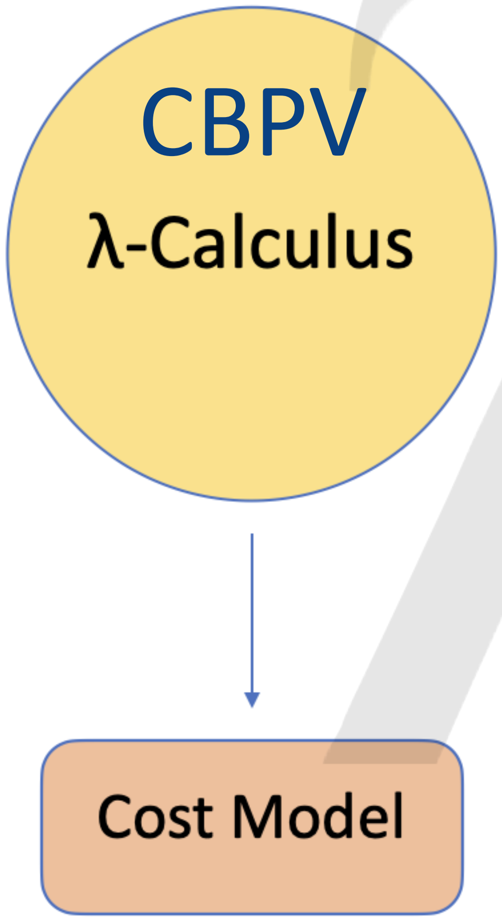


call by push value

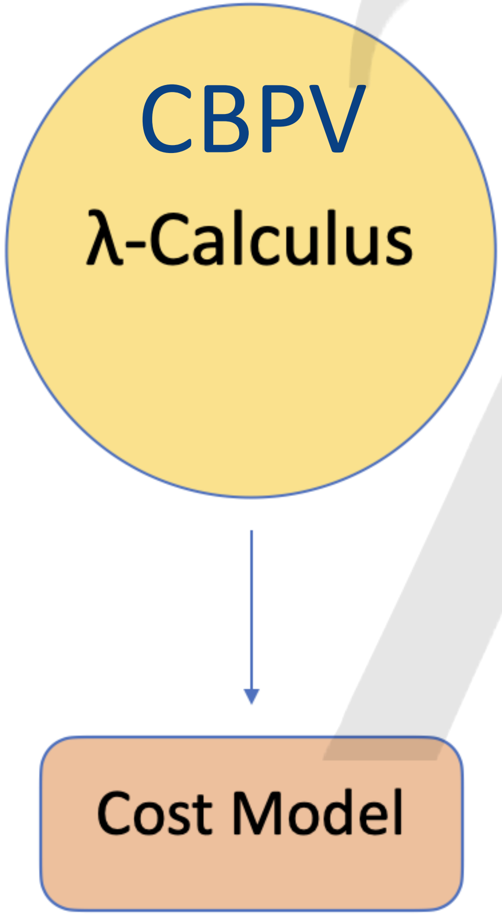
Variant of λ-Calculus

uses the pair "thunk" and "force" to let the programmer freely decide when they wish to evaluate which evaluation

A Verified Cost Model for CBPV

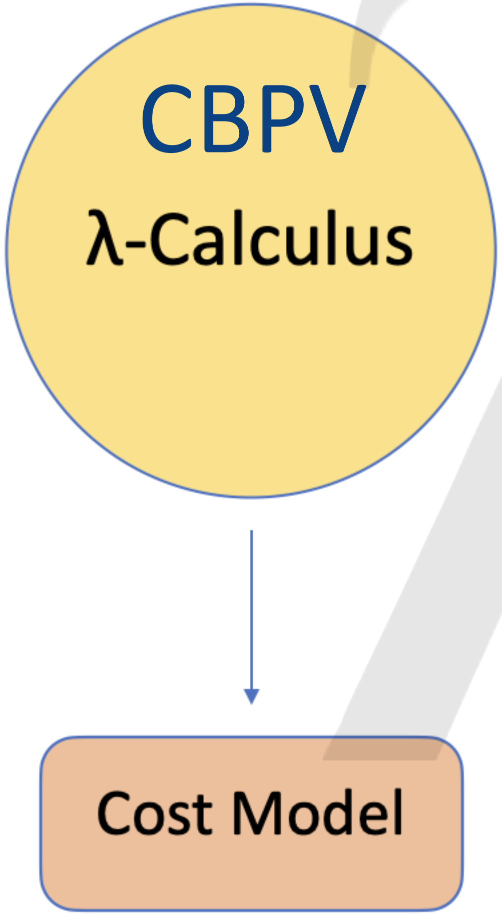


A Verified Cost Model for CBPV



<u>Time</u>: Number of β-reductions
<u>Space</u>: Size of the biggest intermediate term

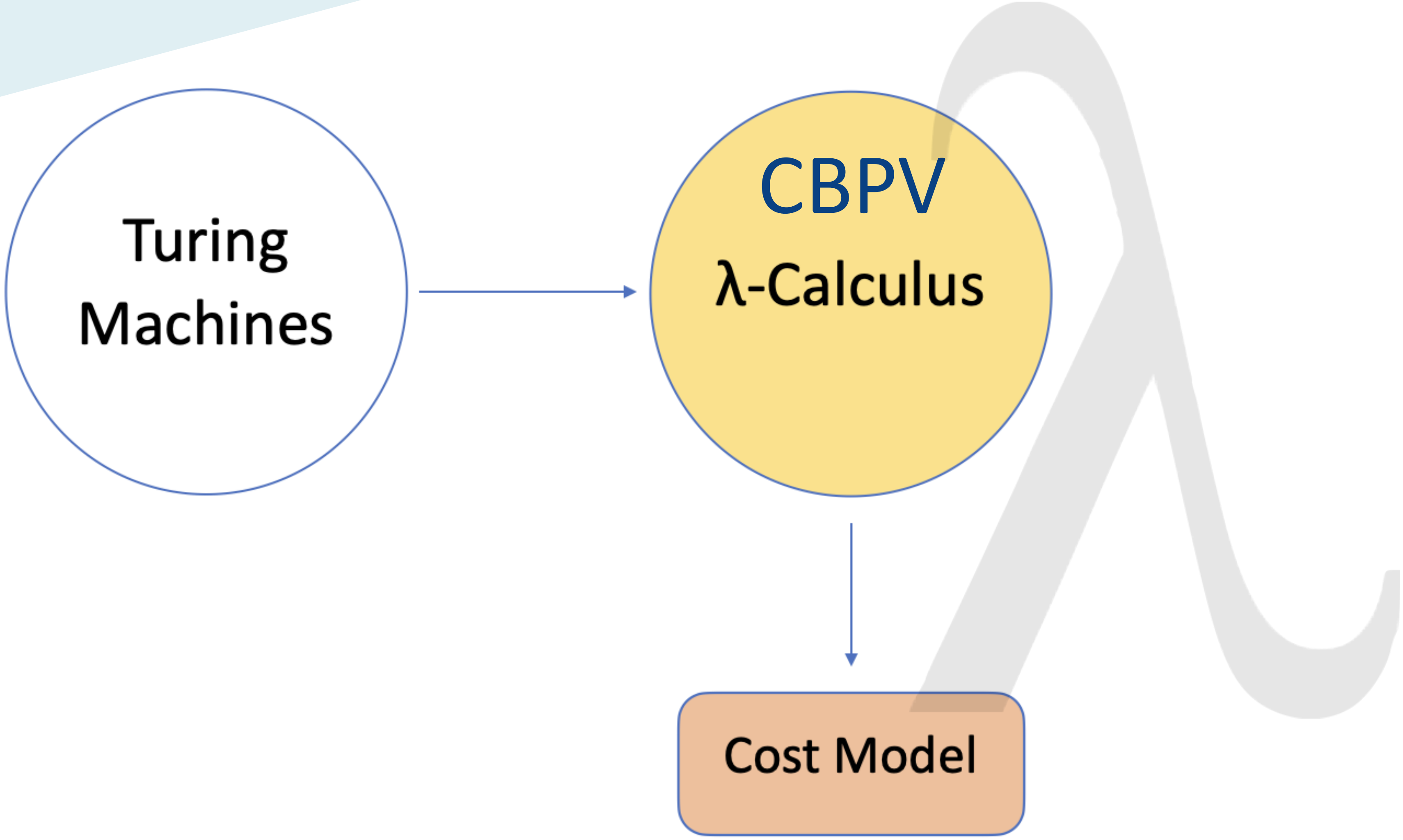
A Verified Cost Model for CBPV



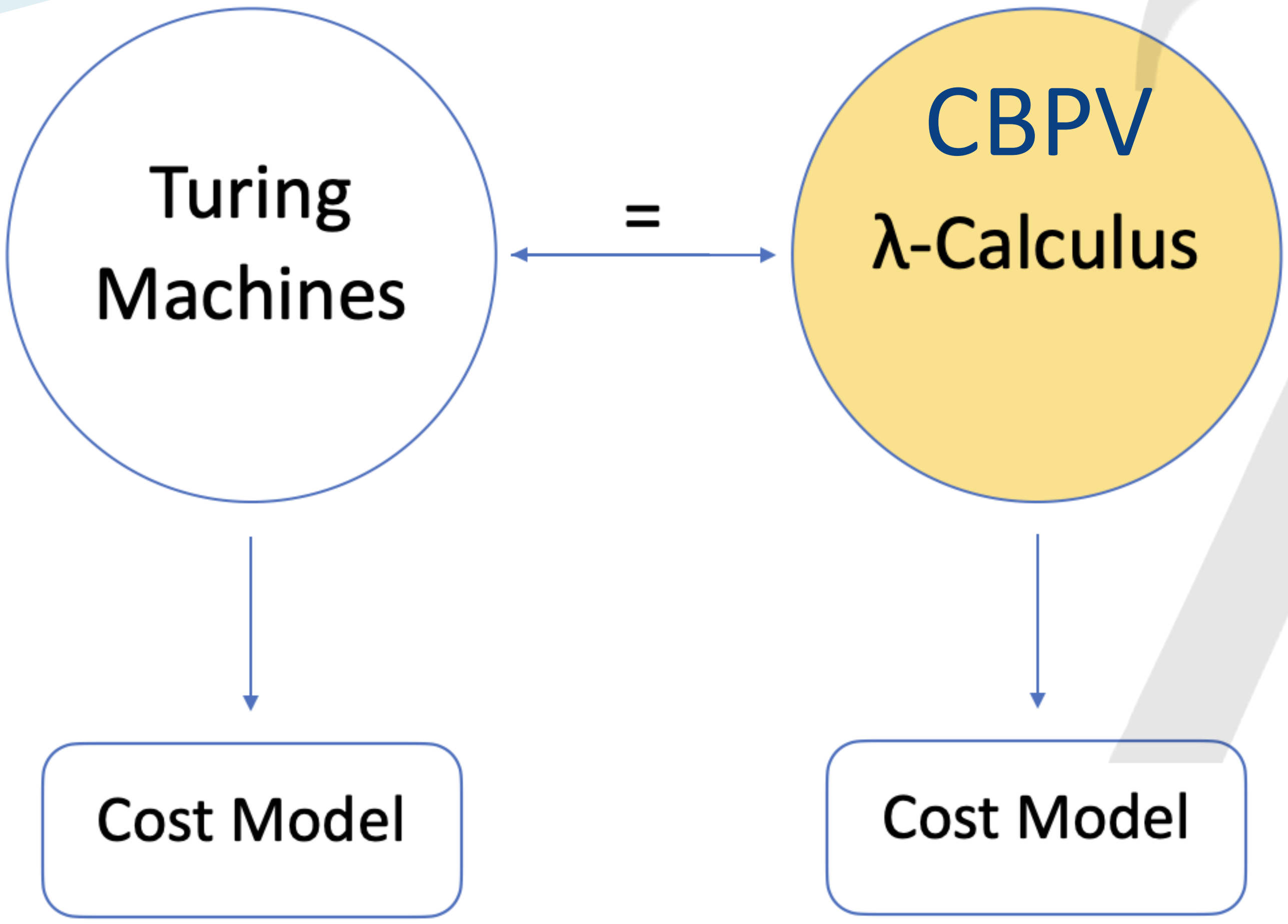
?

Time: Number of β-reductions
Space: Size of the biggest intermediate term

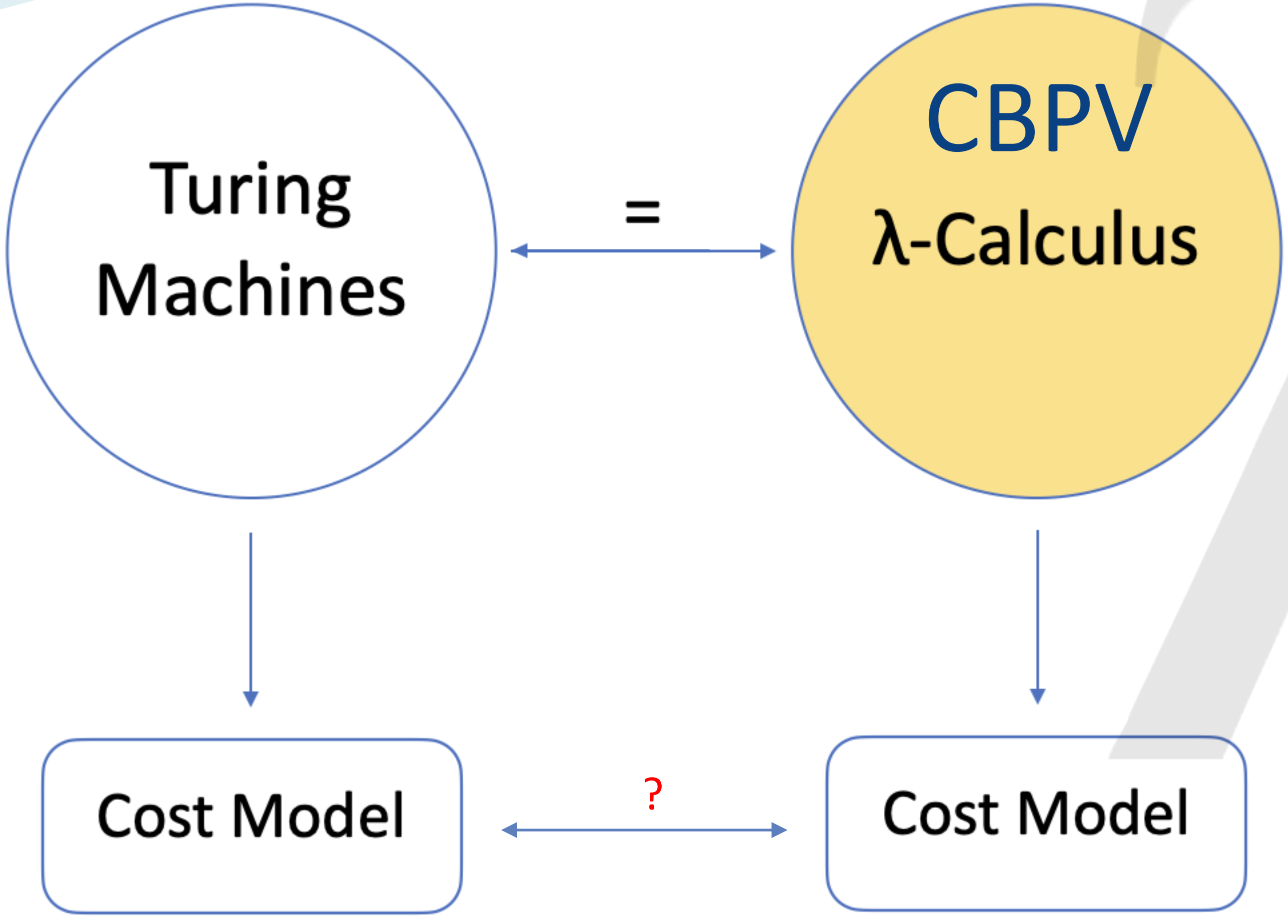
A Verified Cost Model for CBPV



A Verified Cost Model for CBPV



A Verified Cost Model for CBPV



The Invariance Thesis

Slot and van Emde Boas (1984):

“*Reasonable* machines simulate each other with *polynomially* bounded overhead in *time* and *constant* factor overhead in *space*”

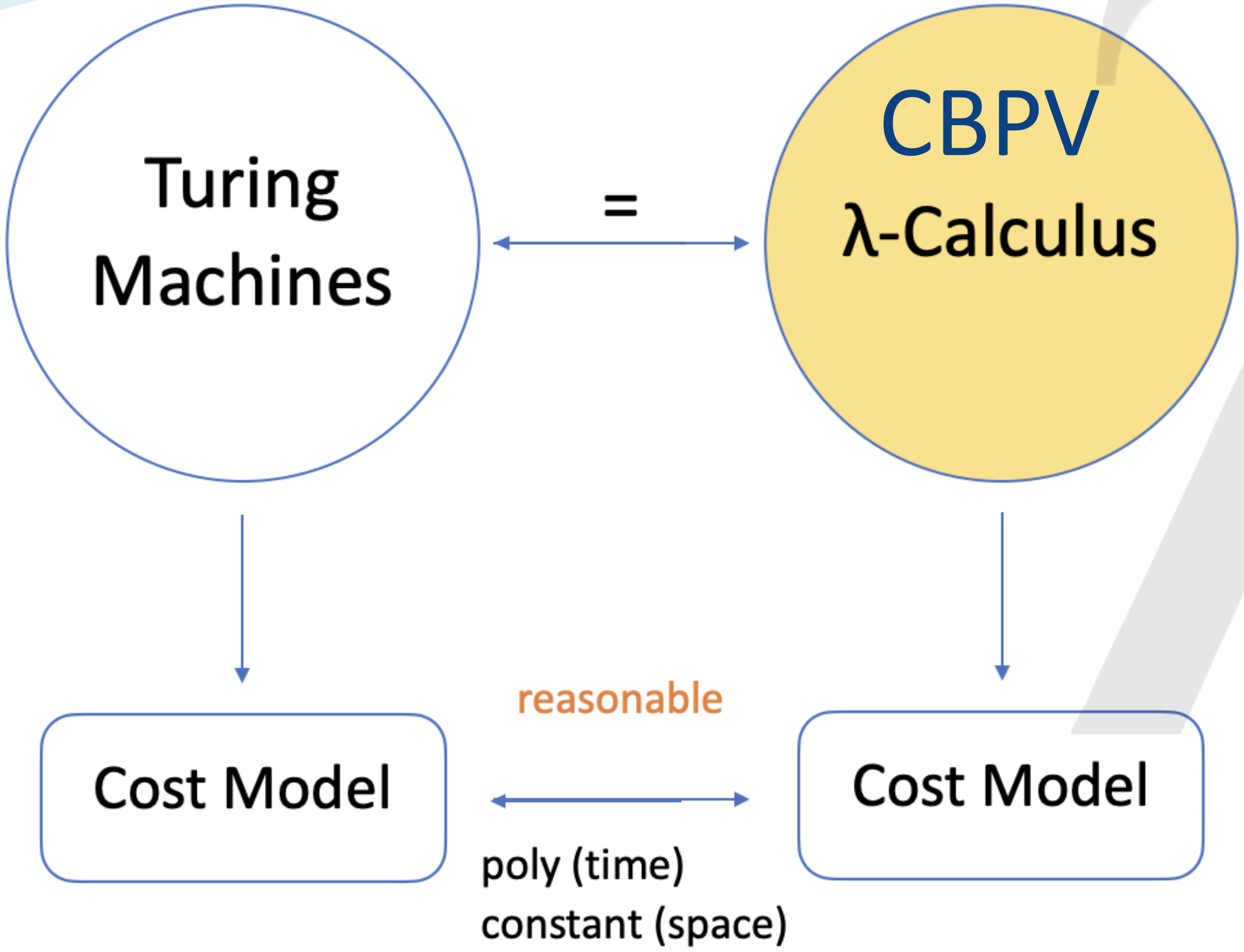
The Invariance Thesis

Slot and van Emde Boas (1984):

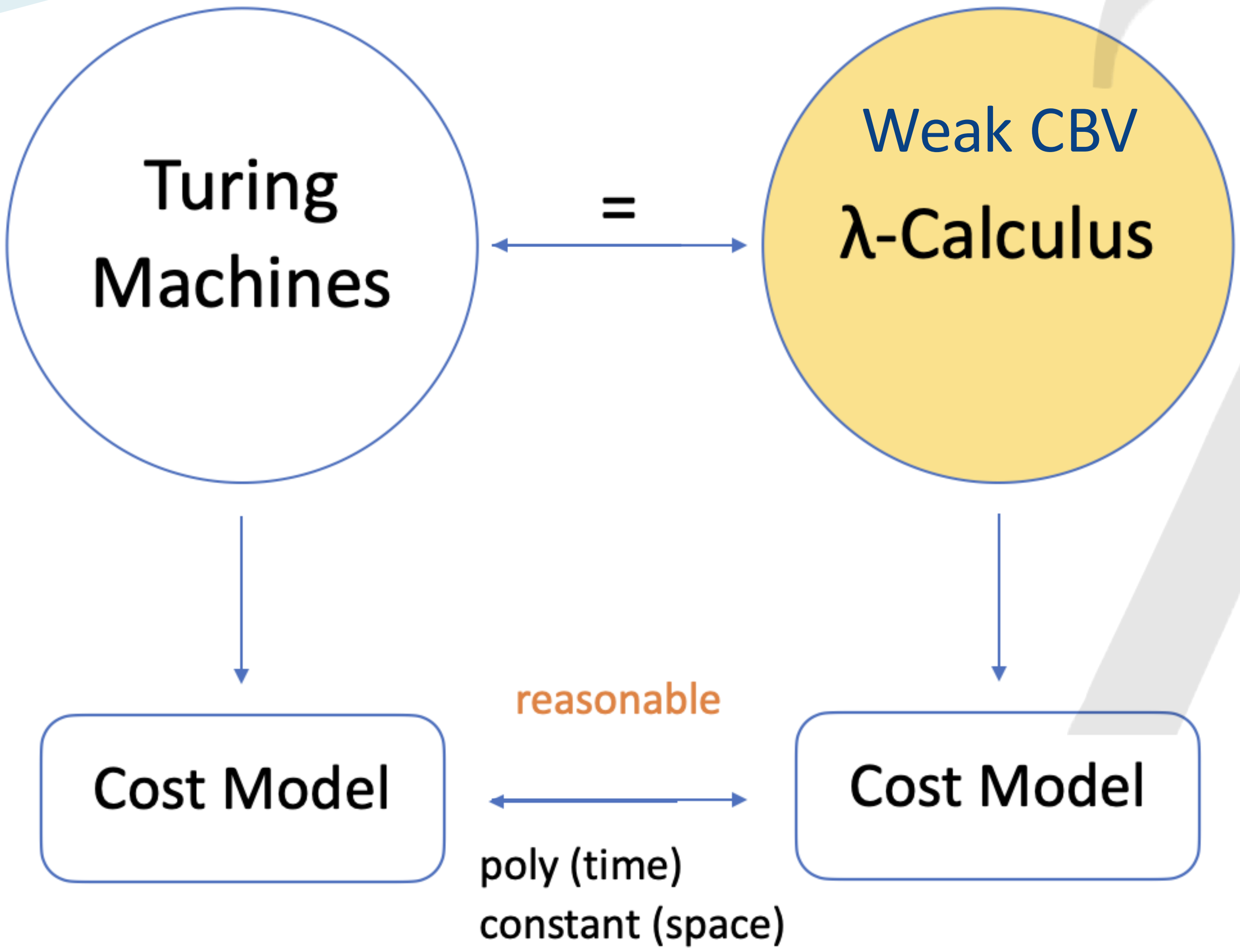
"*Reasonable* machines simulate each other with *polynomially* bounded overhead in *time* and *constant* factor overhead in *space*"

Turing machines are reasonable

A Verified Cost Model for CBPV



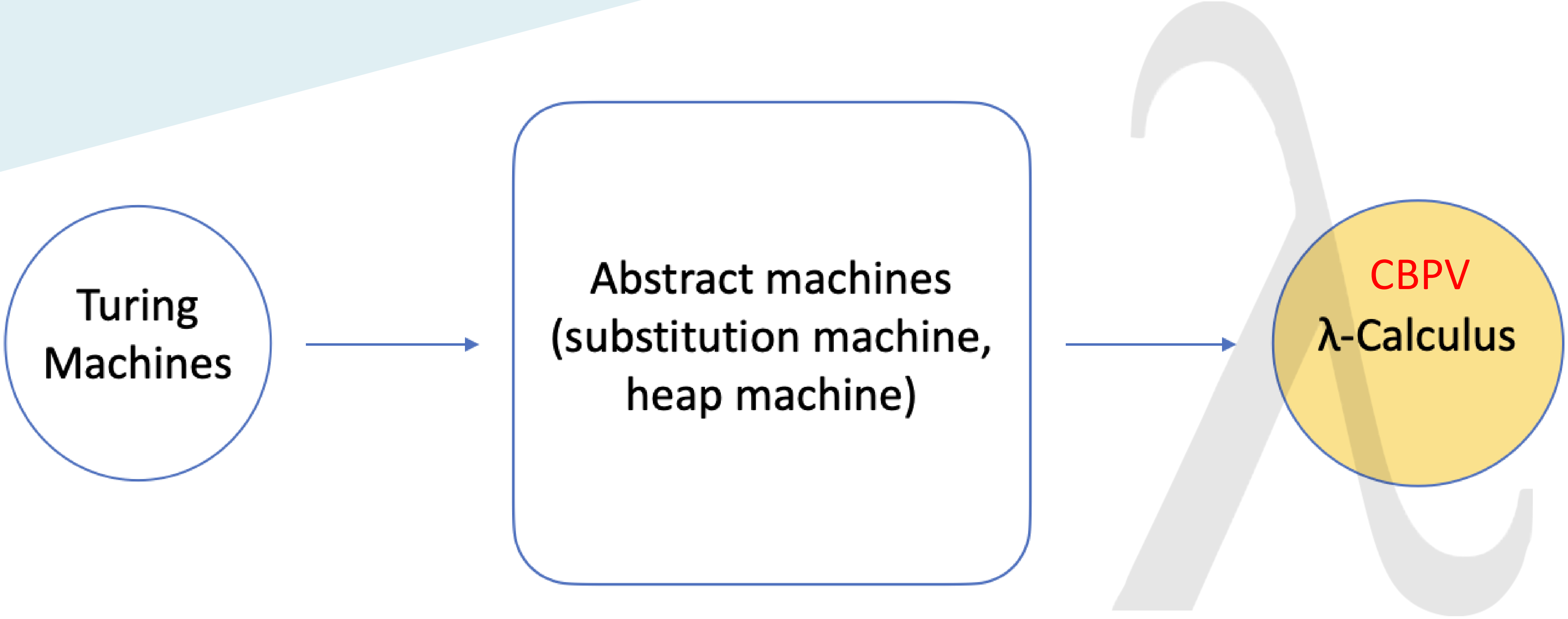
Related Work [Forster, Kunze, and Roth 2020]



Related Work [Forster, Kunze, and Roth 2020]

Turing Machines → Abstract machines (substitution machine, heap machine) → Weak CBV λ-Calculus

A Verified Cost Model for CBPV



A Verified Cost Model for CBPV

Turing Machines → Abstract machines (substitution machine, heap machine) → CBPV λ-Calculus

CBPV

de Bruijn syntax

Definition 33 (Syntax for CBPV).

$$
\begin{array}{llllll}
\textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid & \mathsf{thunk}\ \mathtt{M} & \\
\textit{Computations} & \mathtt{M} & := & \lambda.\,\mathtt{M} \mid & \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid & \mathsf{force}\ \mathtt{V} \mid \quad \mathsf{ret}\ \mathtt{V} \mid \\
 & & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M}
\end{array}
$$

CBPV

Definition 33 (Syntax for CBPV).

$$
\begin{array}{llll}
\textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid \quad \mathsf{thunk}\ \mathtt{M} \\
\textit{Computations} & \mathtt{M} & := & \lambda.\,\mathtt{M} \mid \quad \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid \quad \mathsf{force}\ \mathtt{V} \mid \quad \mathsf{ret}\ \mathtt{V} \mid \\
 & & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid \quad \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid \quad \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M}
\end{array}
$$

CBPV

Definition 33 (Syntax for CBPV).

$$
\begin{array}{llll}
\textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid \mathsf{thunk}\ \mathtt{M} \\
\textit{Computations} & \mathtt{M} & := & \lambda.\,\mathtt{M} \mid \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid \mathsf{force}\ \mathtt{V} \mid \mathsf{ret}\ \mathtt{V} \mid \\
 & & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M}
\end{array}
$$

CBPV

Definition 33 (Syntax for CBPV).

$$\begin{array}{llll} \textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid \mathsf{thunk}\ \mathtt{M} \\ \textit{Computations} & \mathtt{M} & := & \lambda.\,\mathtt{M} \mid \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid \mathsf{force}\ \mathtt{V} \mid \mathsf{ret}\ \mathtt{V} \mid \\ & & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M} \end{array}$$

CBPV

Definition 33 (Syntax for CBPV).

$$
\begin{array}{llllll}
\textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid & \mathsf{thunk}\ \mathtt{M} & \\
\textit{Computations} & \mathtt{M} & := & \lambda.\mathtt{M} \mid & \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid & \mathsf{force}\ \mathtt{V} \mid \quad \mathsf{ret}\ \mathtt{V} \mid \\
& & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M}
\end{array}
$$

CBPV

Definition 33 (Syntax for CBPV).

$$
\begin{array}{llllll}
\textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid & \mathsf{thunk}\ \mathtt{M} & \\
\textit{Computations} & \mathtt{M} & := & \lambda.\,\mathtt{M} \mid & \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid & \mathsf{force}\ \mathtt{V} \mid \quad \mathsf{ret}\ \mathtt{V} \mid \\
 & & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M}
\end{array}
$$

pseq m2 m1 n = seq m2 (seq m1 n)

CBPV

Definition 34. *(Substitution function)*

$$
\begin{aligned}
(\lambda.\, m)^i_u &= \lambda.\, (m^{i+1}_u) \\
(\mathsf{app}\; m\; v)^i_u &= \mathsf{app}\; (m^i_u)\; (v^i_u) \\
(\mathsf{ret}\; v)^i_u &= \mathsf{ret}\; (v^i_u) \\
(\mathsf{seq}\; m\; n)^i_u &= \mathsf{seq}\; (m^i_u)\; (n^{i+1}_u) \\
(\mathsf{pseq}\; m_2\; m_1\; n)^i_u &= \mathsf{pseq}\; ({m_2}^i_u)\; ({m_1}^i_u)\; (n^{i+2}_u)
\end{aligned}
$$

$$
\begin{aligned}
(\mathsf{var}\; x)^i_u &= u \;(\mathit{if}\; x = i) \\
(\mathsf{var}\; x)^i_u &= \mathsf{var}\; x \;(\mathit{if}\; x \neq i) \\
(\mathsf{thunk}\; m)^i_u &= \mathsf{thunk}\; (m^i_u) \\
(\mathsf{force}\; v)^i_u &= \mathsf{force}\; (v^i_u) \\
(\mathsf{let}\; v.\, \mathsf{in}\; m)^i_u &= \mathsf{let}\; (v^i_u).\, \mathsf{in}\; (m^{i+1}_u)
\end{aligned}
$$

CBPV

$$
\frac{}{\lambda.\, m \Downarrow \lambda.\, m}
\qquad
\frac{}{\mathsf{ret}\; v \Downarrow \mathsf{ret}\; v}
\qquad
\frac{m \Downarrow m'}{\mathsf{force}\,(\mathsf{thunk}\; m) \Downarrow m'}
\qquad
\frac{m_v^0 \Downarrow m'}{(\mathsf{let}\; v.\, \mathsf{in}\; m) \Downarrow m'}
$$

$$
\frac{m \Downarrow \lambda.\, n \qquad n_v^0 \Downarrow n'}{\mathsf{app}\; m\; v \Downarrow n'}
\qquad
\frac{m \Downarrow \mathsf{ret}\; v \qquad n_v^0 \Downarrow n'}{\mathsf{seq}\; m\; n \Downarrow n'}
$$

$$
\frac{m_1 \Downarrow \mathsf{ret}\; v_1 \qquad m_2 \Downarrow \mathsf{ret}\; v_2 \qquad (n_{v_1}^0)_{v_2}^1 \Downarrow n'}{\mathsf{pseq}\; m_2\; m_1\; n \Downarrow n'}
$$

FIGURE 4.3: Big-Step Semantics for CBPV

CBPV

$$\frac{}{\lambda.\, m \Downarrow \lambda.\, m} \qquad \frac{}{\mathsf{ret}\, v \Downarrow \mathsf{ret}\, v} \qquad \frac{m \Downarrow m'}{\mathsf{force}\,(\mathsf{thunk}\, m) \Downarrow m'} \qquad \frac{m^0_v \Downarrow m'}{(\mathsf{let}\, v.\, \mathsf{in}\, m) \Downarrow m'}$$

$$\frac{m \Downarrow \lambda.\, n \qquad n^0_v \Downarrow n'}{\mathsf{app}\, m\, v \Downarrow n'} \qquad \frac{m \Downarrow \mathsf{ret}\, v \qquad n^0_v \Downarrow n'}{\mathsf{seq}\, m\, n \Downarrow n'}$$

$$\frac{m_1 \Downarrow \mathsf{ret}\, v_1 \qquad m_2 \Downarrow \mathsf{ret}\, v_2 \qquad (n^0_{v_1})^1_{v_2} \Downarrow n'}{\mathsf{pseq}\, m_2\, m_1\, n \Downarrow n'}$$

FIGURE 4.3: Big-Step Semantics for CBPV

CBPV

$$\frac{}{\lambda.\, m \Downarrow \lambda.\, m} \qquad \frac{}{\mathsf{ret}\, v \Downarrow \mathsf{ret}\, v} \qquad \frac{m \Downarrow m'}{\mathsf{force}\,(\mathsf{thunk}\, m) \Downarrow m'} \qquad \frac{m_v^0 \Downarrow m'}{(\mathsf{let}\, v.\, \mathsf{in}\, m) \Downarrow m'}$$

$$\frac{m \Downarrow \lambda.\, n \qquad n_v^0 \Downarrow n'}{\mathsf{app}\, m\, v \Downarrow n'} \qquad \frac{m \Downarrow \mathsf{ret}\, v \qquad n_v^0 \Downarrow n'}{\mathsf{seq}\, m\, n \Downarrow n'}$$

$$\frac{m_1 \Downarrow \mathsf{ret}\, v_1 \qquad m_2 \Downarrow \mathsf{ret}\, v_2 \qquad (n_{v_1}^0)_{v_2}^1 \Downarrow n'}{\mathsf{pseq}\, m_2\, m_1\, n \Downarrow n'}$$

FIGURE 4.3: Big-Step Semantics for CBPV

CBPV

$$\frac{}{\lambda.\, m \Downarrow_0 \lambda.\, m} \qquad \frac{}{\mathsf{ret}\; v \Downarrow_0 \mathsf{ret}\; v} \qquad \frac{m \Downarrow_k m'}{\mathsf{force}\;(\mathsf{thunk}\; m) \Downarrow_{k+2} m'}$$

$$\frac{m_v^0 \Downarrow_k m'}{\mathsf{let}\; v.\, \mathsf{in}\; m \Downarrow_{k+1} m'} \qquad \frac{m \Downarrow_{k_1} \lambda.\, n \qquad n_v^0 \Downarrow_{k_2} n'}{\mathsf{app}\; m\; v \Downarrow_{k_1+k_2+1} n'} \qquad \frac{m \Downarrow_{k_1} \mathsf{ret}\; v \qquad n_v^0 \Downarrow_{k_2} n'}{\mathsf{seq}\; m\; n \Downarrow_{k_1+k_2+1} n'}$$

$$\frac{m_1 \Downarrow_{k_1} \mathsf{ret}\; v_1 \qquad m_2 \Downarrow_{k_2} \mathsf{ret}\; v_2 \qquad (n_{v_1}^0)_{v_2}^1 \Downarrow_{k_3} n'}{\mathsf{pseq}\; m_2\; m_1\; n \Downarrow_{k_1+k_2+k_3+1} n'}$$

FIGURE 4.4: Big-Step Semantics for CBPV with Time Cost

CBPV

$$\frac{}{\lambda. m \Downarrow_0 \lambda. m} \qquad \frac{}{\mathsf{ret}\ v \Downarrow_0 \mathsf{ret}\ v} \qquad \frac{m \Downarrow_k m'}{\mathsf{force}\,(\mathsf{thunk}\ m) \Downarrow_{k+2} m'}$$

$$\frac{m^0_v \Downarrow_k m'}{\mathsf{let}\ v.\ \mathsf{in}\ m \Downarrow_{k+1} m'} \qquad \frac{m \Downarrow_{k_1} \lambda. n \qquad n^0_v \Downarrow_{k_2} n'}{\mathsf{app}\ m\ v \Downarrow_{k_1+k_2+1} n'} \qquad \frac{m \Downarrow_{k_1} \mathsf{ret}\ v \qquad n^0_v \Downarrow_{k_2} n'}{\mathsf{seq}\ m\ n \Downarrow_{k_1+k_2+1} n'}$$

$$\frac{m_1 \Downarrow_{k_1} \mathsf{ret}\ v_1 \qquad m_2 \Downarrow_{k_2} \mathsf{ret}\ v_2 \qquad (n^0_{v_1})^1_{v_2} \Downarrow_{k_3} n'}{\mathsf{pseq}\ m_2\ m_1\ n \Downarrow_{k_1+k_2+k_3+1} n'}$$

FIGURE 4.4: Big-Step Semantics for CBPV with Time Cost

CBPV

$$\frac{}{\lambda.\, m \Downarrow_0 \lambda.\, m} \qquad \frac{}{\mathsf{ret}\; v \Downarrow_0 \mathsf{ret}\; v} \qquad \frac{m \Downarrow_k m'}{\mathsf{force}\,(\mathsf{thunk}\; m) \Downarrow_{k+2} m'}$$

$$\frac{m_v^0 \Downarrow_k m'}{\mathsf{let}\; v.\, \mathsf{in}\; m \Downarrow_{k+1} m'} \qquad \frac{m \Downarrow_{k_1} \lambda.\, n \qquad n_v^0 \Downarrow_{k_2} n'}{\mathsf{app}\; m\; v \Downarrow_{k_1+k_2+1} n'} \qquad \frac{m \Downarrow_{k_1} \mathsf{ret}\; v \qquad n_v^0 \Downarrow_{k_2} n'}{\mathsf{seq}\; m\; n \Downarrow_{k_1+k_2+1} n'}$$

$$\frac{m_1 \Downarrow_{k_1} \mathsf{ret}\; v_1 \qquad m_2 \Downarrow_{k_2} \mathsf{ret}\; v_2 \qquad (n_{v_1}^0)_{v_2}^1 \Downarrow_{k_3} n'}{\mathsf{pseq}\; m_2\; m_1\; n \Downarrow_{k_1+k_2+k_3+1} n'}$$

FIGURE 4.4: Big-Step Semantics for CBPV with Time Cost

CBPV

$$\frac{}{\lambda.\, m \Downarrow^{\|\lambda.\, m\|} \lambda.\, m} \qquad \frac{}{\mathsf{ret}\, v \Downarrow^{\|\mathsf{ret}\, v\|} \mathsf{ret}\, v}$$

$$\frac{m \Downarrow^{s} m'}{\mathsf{force}\,(\mathsf{thunk}\, m) \Downarrow^{\max(s,\, \|m\|+2)} m'} \qquad \frac{m^0_v \Downarrow^{s} m'}{\mathsf{let}\, v.\, \mathsf{in}\, m \Downarrow^{\max(s,\, \|v\|+\|m\|+1)} m'}$$

$$\frac{m \Downarrow^{s_1} \lambda.\, n \qquad n^0_v \Downarrow^{s_2} n'}{\mathsf{app}\, m\, v \Downarrow^{\max(s_1+\|n\|+1,\, s_2)} n'} \qquad \frac{m \Downarrow^{s_1} \mathsf{ret}\, v \qquad n^0_v \Downarrow^{s_2} n'}{\mathsf{seq}\, m\, n \Downarrow^{\max(s_1+\|n\|+1,\, s_2)} n'}$$

$$\frac{m_1 \Downarrow^{s_1} \mathsf{ret}\, v_1 \qquad m_2 \Downarrow^{s_2} \mathsf{ret}\, v_2 \qquad (n^0_{v_1})^1_{v2} \Downarrow^{s_3} n'}{\mathsf{pseq}\, m_2\, m_1\, n \Downarrow^{\max(s_1+\|m_2\|+\|n\|+1,\, \max(\|v_1\|+s_2+\|n\|+1, s_3))} m}$$

FIGURE 4.5: Big-Step Semantics for CBPV with Space Cost

CBPV

$$\frac{}{\lambda.\, m \Downarrow^{\|\lambda.\, m\|} \lambda.\, m} \qquad \frac{}{\mathsf{ret}\ v \Downarrow^{\|\mathsf{ret}\ v\|} \mathsf{ret}\ v}$$

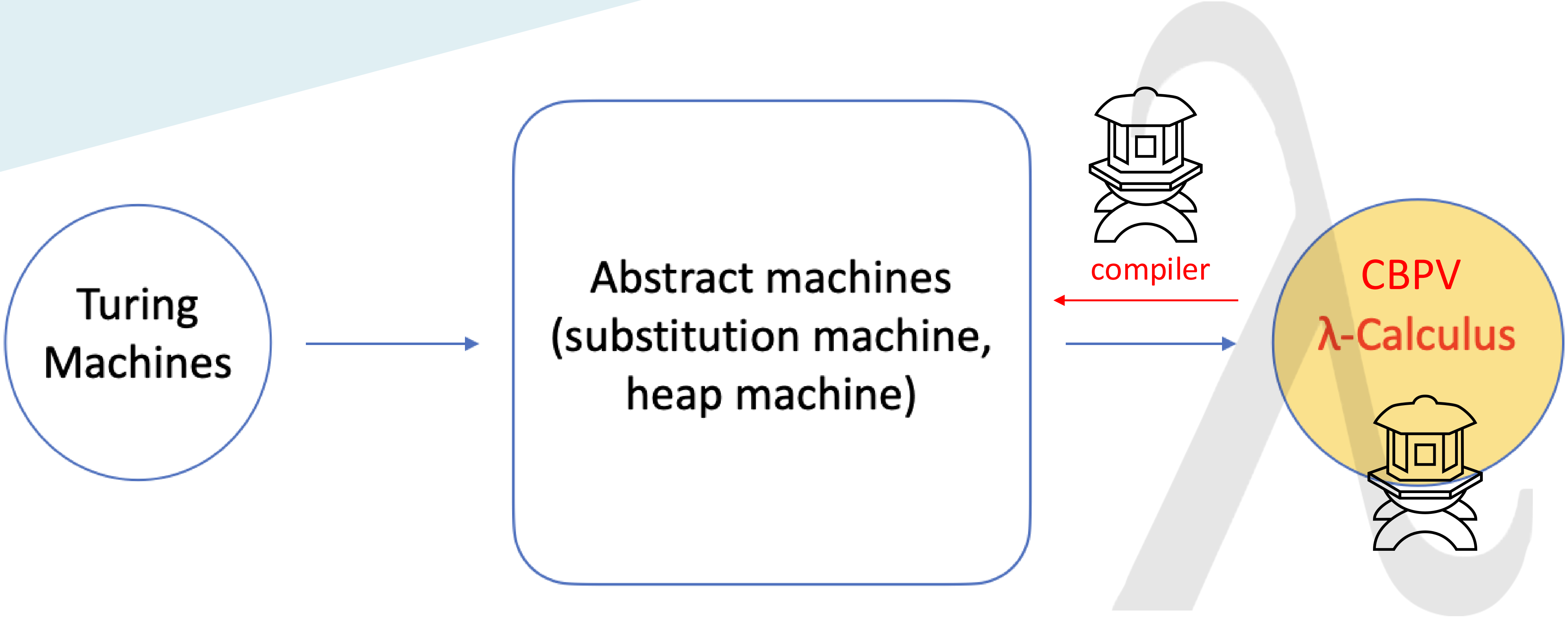
$$\frac{m \Downarrow^{s} m'}{\mathsf{force}\ (\mathsf{thunk}\ m) \Downarrow^{\max(s, \|m\|+2)} m'} \qquad \frac{m_v^0 \Downarrow^{s} m'}{\mathsf{let}\ v.\ \mathsf{in}\ m \Downarrow^{\max(s, \|v\|+\|m\|+1)} m'}$$

$$\frac{m \Downarrow^{s_1} \lambda.\, n \qquad n_v^0 \Downarrow^{s_2} n'}{\mathsf{app}\ m\ v \Downarrow^{\max(s_1+\|n\|+1, s_2)} n'} \qquad \frac{m \Downarrow^{s_1} \mathsf{ret}\ v \qquad n_v^0 \Downarrow^{s_2} n'}{\mathsf{seq}\ m\ n \Downarrow^{\max(s_1+\|n\|+1, s_2)} n'}$$

$$\frac{m_1 \Downarrow^{s_1} \mathsf{ret}\ v_1 \qquad m_2 \Downarrow^{s_2} \mathsf{ret}\ v_2 \qquad (n_{v_1}^0)_{v2}^1 \Downarrow^{s_3} n'}{\mathsf{pseq}\ m_2\ m_1\ n \Downarrow^{\max(s_1+\|m_2\|+\|n\|+1, \max(\|v_1\|+s_2+\|n\|+1, s_3))} m}$$

FIGURE 4.5: Big-Step Semantics for CBPV with Space Cost

A Verified Cost Model for CBPV



Compiler

Definition 36 (Program Tokens).

$$\begin{aligned} \mathsf{t} \in \mathsf{Tok} \quad := \quad & \mathsf{varT}\, x \mid \mathsf{thunkT} \mid \mathsf{endThunkT} \mid \mathsf{forceT} \mid \mathsf{applyT} \mid \\ & \mathsf{lamT} \mid \mathsf{endLamT} \mid \mathsf{retT} \mid \mathsf{endRetT} \mid \mathsf{seqT} \mid \mathsf{endSeqT} \mid \\ & \mathsf{pseqT} \mid \mathsf{endPseqT} \mid \mathsf{letinT} \mid \mathsf{endLetinT} \end{aligned}$$

Compiler

Definition 39 (Compilation Function for Substitution Machine).

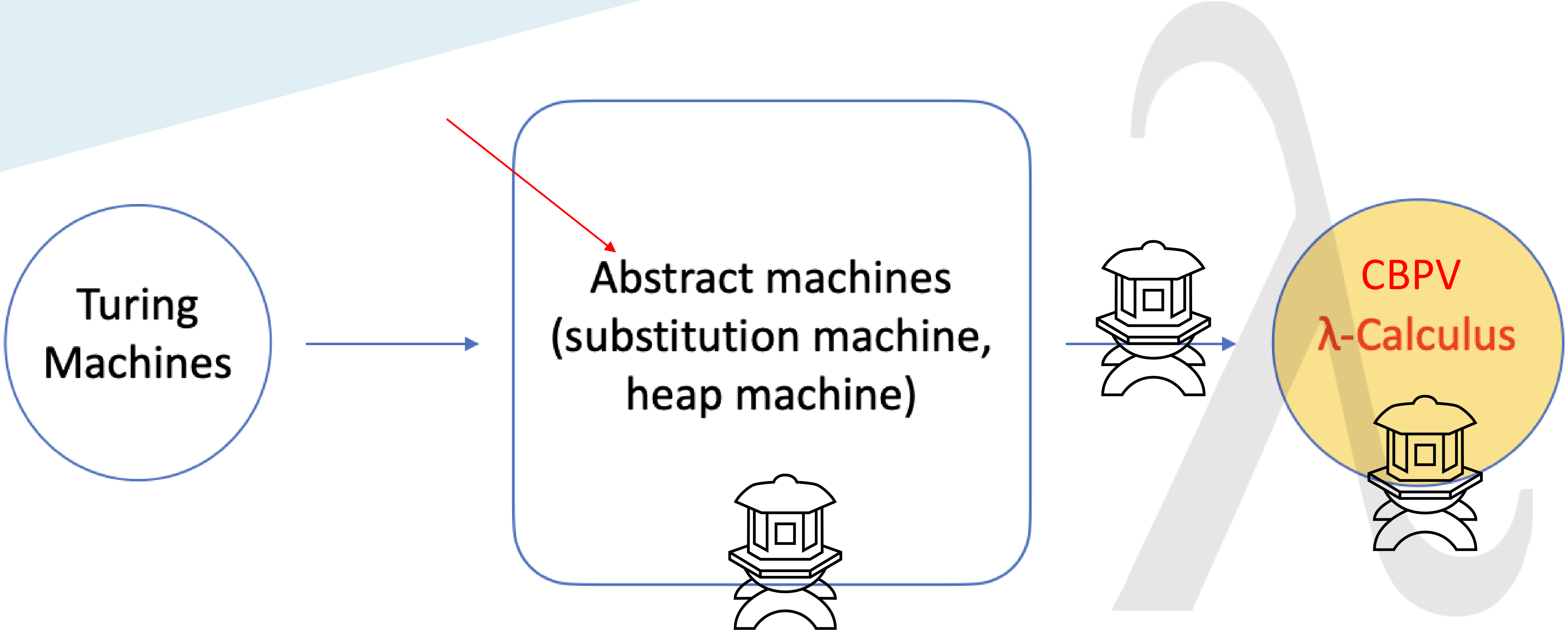
$$
\begin{array}{lcl}
\gamma(\mathsf{var}\ x) & = & \mathsf{varT}\ x \\
\gamma(\mathsf{thunk}\ m) & = & \mathsf{thunkT} :: \gamma(m) +\!\!+ [\mathsf{endThunkT}] \\
\gamma(\mathsf{force}\ v) & = & \gamma(v) +\!\!+ [\mathsf{forceT}] \\
\gamma(\mathsf{ret}\ v) & = & \mathsf{retT} :: \gamma(v) +\!\!+ [\mathsf{endRetT}] \\
\gamma(\lambda.\ m) & = & \mathsf{lamT} :: \gamma(m) +\!\!+ [\mathsf{endLamT}] \\
\gamma(\mathsf{app}\ m\ v) & = & \gamma(m) +\!\!+ \gamma(v) +\!\!+ [\mathsf{applyT}] \\
\gamma(\mathsf{seq}\ m\ n) & = & \gamma(m) +\!\!+ [\mathsf{seqT}] +\!\!+ \gamma(n) +\!\!+ [\mathsf{endSeqT}] \\
\gamma(\mathsf{pseq}\ m_1\ m_2\ n) & = & \gamma(m_1) +\!\!+ \gamma(m_2) +\!\!+ [\mathsf{pseqT}] +\!\!+ \gamma(n) +\!\!+ [\mathsf{endPseqT}] \\
\gamma(\mathsf{let}\ v.\ \mathsf{in}\ m) & = & \gamma(v) +\!\!+ [\mathsf{letinT}] +\!\!+ \gamma(m) +\!\!+ [\mathsf{endLetinT}]
\end{array}
$$

Compiler

Definition 40 (Compilation Function for Heap Machine).

$$
\begin{array}{lcl}
\gamma(\mathsf{var}\ x) & = & \mathsf{varT}\ x \\
\gamma(\mathsf{thunk}\ m) & = & \mathsf{thunkT}\ ::\ \gamma(m) +\!\!+ [\mathsf{endThunkT}] \\
\gamma(\mathsf{force}\ v) & = & \gamma(v) +\!\!+ [\mathsf{forceT}] \\
\gamma(\mathsf{ret}\ v) & = & \gamma(v) +\!\!+ [\mathsf{retT}] \\
\gamma(\lambda.\ m) & = & \mathsf{lamT}\ ::\ \gamma(m) +\!\!+ [\mathsf{endLamT}] \\
\gamma(\mathsf{app}\ m\ v) & = & \gamma(m) +\!\!+ \gamma(v) +\!\!+ [\mathsf{applyT}] \\
\gamma(\mathsf{seq}\ m\ n) & = & \gamma(m) +\!\!+ [\mathsf{seqT}] +\!\!+ \gamma(n) +\!\!+ [\mathsf{endSeqT}] \\
\gamma(\mathsf{pseq}\ m_1\ m_2\ n) & = & \gamma(m_1) +\!\!+ \gamma(m_2) +\!\!+ [\mathsf{pseqT}] +\!\!+ \gamma(n) +\!\!+ [\mathsf{endPseqT}] \\
\gamma(\mathsf{let}\ v.\ \mathsf{in}\ m) & = & \gamma(v) +\!\!+ [\mathsf{letinT}] +\!\!+ \gamma(m) +\!\!+ [\mathsf{endLetinT}]
\end{array}
$$

Abstract Machines



Substitution Machine

Task Stack | Value Stack

Substitution Machine

Definition 33 (Syntax for CBPV).

$$
\begin{array}{llllll}
\textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid & \mathsf{thunk}\ \mathtt{M} & & \\
\textit{Computations} & \mathtt{M} & := & \lambda.\,\mathtt{M} \mid & \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid & \mathsf{force}\ \mathtt{V} \mid & \mathsf{ret}\ \mathtt{V} \mid \\
& & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M} &
\end{array}
$$

Substitution Machine

Definition 33 (Syntax for CBPV).

$$
\begin{array}{llllll}
\textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid & \mathsf{thunk}\ \mathtt{M} & \\
\textit{Computations} & \mathtt{M} & := & \lambda.\mathtt{M} \mid & \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid & \mathsf{force}\ \mathtt{V} \mid \quad \mathsf{ret}\ \mathtt{V} \mid \\
 & & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M}
\end{array}
$$

Example Transition Rule for Substitution Machine - Variable

$$
\begin{array}{ll|r|}
 & (\mathsf{varT}\ n :: P) :: T & V \\
\triangleright & P ::_{tc} T & \mathsf{varT}\ n :: V
\end{array}
$$

Substitution Machine

Definition 33 (Syntax for CBPV).

$$
\begin{array}{llllll}
\textit{Values} & \mathtt{V} ::= & \mathsf{var}\ x \mid & \mathsf{thunk}\ \mathtt{M} & & \\
\textit{Computations} & \mathtt{M} ::= & \lambda.\mathtt{M} \mid & \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid & \mathsf{force}\ \mathtt{V} \mid & \mathsf{ret}\ \mathtt{V} \mid \\
 & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M} &
\end{array}
$$

Example Transition Rule for Substitution Machine - Variable

$$
\begin{array}{ll|r|}
 & (\mathsf{varT}\ n :: P) :: T & V \\
\triangleright & P ::_{tc} T & \mathsf{varT}\ n :: V
\end{array}
$$

Lemma 45 (Substitution Machine Time Simulation). *If $m \Downarrow_k n$, then there exists k' such that $(P_m, []) \triangleright_{k'} ([], P_n)$ where $k' \leq 3 * k + 1$ and $P_m \gg m$ and $P_n \gg n$.*

Lemma 46 (Substitution Machine Space Simulation). *If $m \Downarrow^s n$ then there exists σ such that $(P_m, []) \triangleright_*^\sigma ([], P_n)$, where $s \leq \sigma \leq 9 * s$ and $P_m \gg m$ and $P_n \gg n$.*

Substitution Machine

Definition 33 (Syntax for CBPV).

$$\begin{array}{llllll} \textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid & \mathsf{thunk}\ \mathtt{M} & \\ \textit{Computations} & \mathtt{M} & := & \lambda.\mathtt{M} \mid & \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid & \mathsf{force}\ \mathtt{V} \mid \quad \mathsf{ret}\ \mathtt{V} \mid \\ & & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M} \end{array}$$

Example Transition Rule for Substitution Machine - Variable

$$\begin{array}{ll|r|} & (\mathsf{varT}\ n :: P) :: T & V \\ \triangleright & P ::_{tc} T & \mathsf{varT}\ n :: V \end{array}$$

Lemma 45 (Substitution Machine Time Simulation). *If $m \Downarrow_k n$, then there exists k' such that $(P_m, []) \triangleright_{k'} ([], P_n)$ where $k' \leq 3 * k + 1$ and $P_m \gg m$ and $P_n \gg n$.*

Lemma 46 (Substitution Machine Space Simulation). *If $m \Downarrow^s n$ then there exists σ such that $(P_m, []) \triangleright^{\sigma}_{*} ([], P_n)$, where $s \leq \sigma \leq 9 * s$ and $P_m \gg m$ and $P_n \gg n$.*

Heap Machine

Task Stack | Value Stack | Heap

Heap Machine

Definition 33 (Syntax for CBPV).

$$
\begin{array}{llllll}
\textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid & \mathsf{thunk}\ \mathtt{M} & & \\
\textit{Computations} & \mathtt{M} & := & \lambda.\mathtt{M} \mid & \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid & \mathsf{force}\ \mathtt{V} \mid & \mathsf{ret}\ \mathtt{V} \mid \\
 & & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M} &
\end{array}
$$

Example Transition Rule for Substitution Machine - Application

$$
\begin{array}{rl|r|c|l}
 & \langle \mathsf{applyT} :: P, a\rangle :: T & Q :: \langle \mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}], b\rangle :: V & H & \mathsf{put}\ H\{Q, b\} = (H', c) \\
\triangleright & \langle M, c\rangle :: \langle P, a\rangle :: T & V & H' &
\end{array}
$$

Heap Machine

Definition 33 (Syntax for CBPV).

Values	$\mathtt{V}$	$:=$	$\mathsf{var}\ x \mid$	$\mathsf{thunk}\ \mathtt{M}$		
Computations	$\mathtt{M}$	$:=$	$\lambda.\mathtt{M} \mid$	$\mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid$	$\mathsf{force}\ \mathtt{V} \mid$	$\mathsf{ret}\ \mathtt{V} \mid$
			$\mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid$	$\mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid$	$\mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M}$	

Example Transition Rule for Substitution Machine - Variable

$\langle \mathsf{varT}\ x :: P,\ a \rangle :: T$	V	H	$\mathsf{lookup}\ H\ a\ x = g$
$\triangleright\ \langle P,\ a \rangle :: T$	$g :: V$	H	

Heap Machine

Definition 33 (Syntax for CBPV).

$$\begin{array}{llllll} \textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid & \mathsf{thunk}\ \mathtt{M} & \\ \textit{Computations} & \mathtt{M} & := & \lambda.\mathtt{M} \mid & \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid & \mathsf{force}\ \mathtt{V} \mid \quad \mathsf{ret}\ \mathtt{V} \mid \\ & & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M} \end{array}$$

Example Transition Rule for Substitution Machine - Variable

$\langle \mathsf{varT}\ x :: P, a\rangle :: T$	V	H	$\mathsf{lookup}\ H\ a\ x = g$
$\rhd\ \langle P, a\rangle :: T$	$g :: V$	H	

Lemma 55 (Heap Machine Time Simulation). *If $m \Downarrow_k n$ then there exists k' such that $(\langle P_m, 0\rangle, [\,], [\,]) \rhd_{k'} ([\,], \langle P_n, a\rangle, H)$, where $k' \leq 7 * k + 5$ and $\langle P_m, 0\rangle \gg m$ and $\langle P_n, a\rangle \gg_H n$.*

Lemma 56 (Heap Machine Space Simulation). *If $(P_m, [\,], [\,]) \rhd_k (T, N, H)$ and $P_m \gg m$ then $\|T \mathbin{+\!\!+} N \mathbin{+\!\!+} H\| \leq (k+1) * (3 * k + 4 * \|m\|)$.*

Heap Machine

Definition 33 (Syntax for CBPV).

$$
\begin{array}{llllll}
\textit{Values} & \mathtt{V} & := & \mathsf{var}\ x \mid & \mathsf{thunk}\ \mathtt{M} & \\
\textit{Computations} & \mathtt{M} & := & \lambda.\mathtt{M} \mid & \mathsf{app}\ \mathtt{M}\ \mathtt{V} \mid & \mathsf{force}\ \mathtt{V} \mid \quad \mathsf{ret}\ \mathtt{V} \mid \\
 & & & \mathsf{seq}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{pseq}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid & \mathsf{let}\ \mathtt{V}.\ \mathsf{in}\ \mathtt{M}
\end{array}
$$

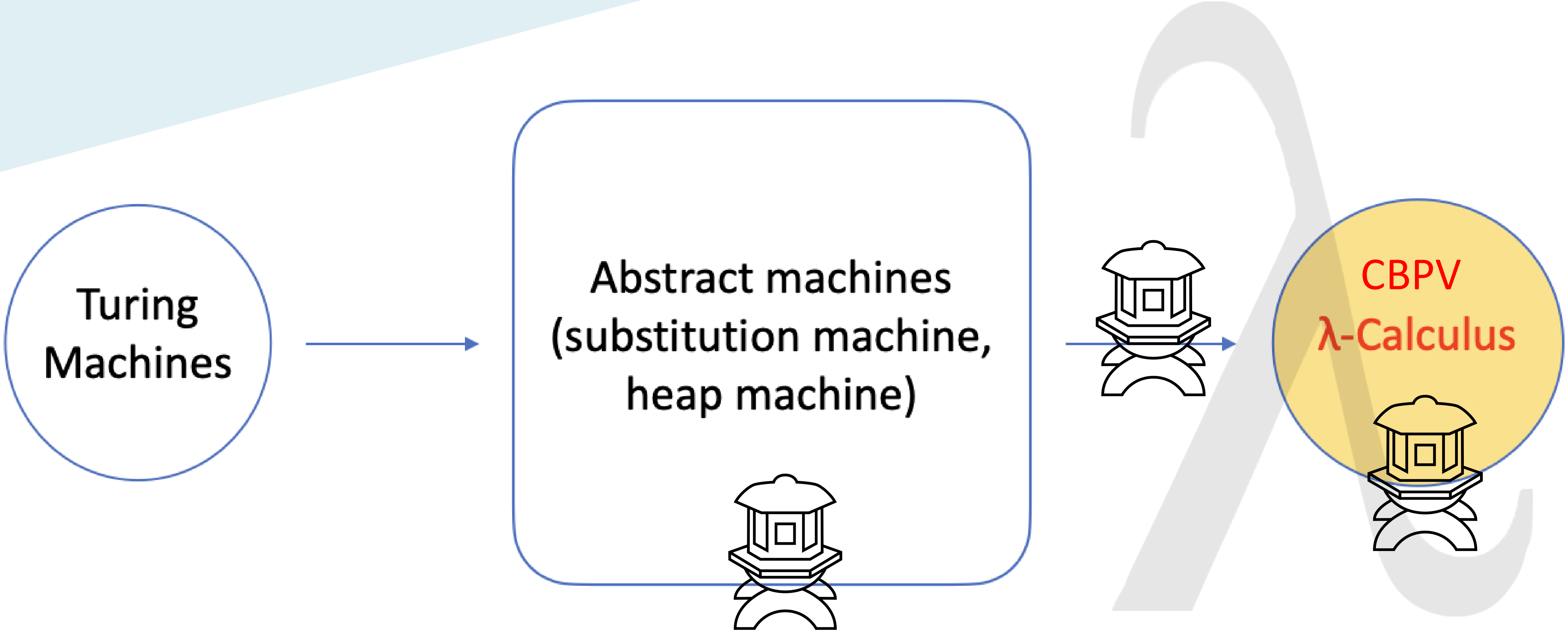
Example Transition Rule for Substitution Machine - Variable

$\langle \mathsf{varT}\ x :: P, a\rangle :: T$ $\triangleright\ \langle P, a\rangle :: T$	V $g :: V$	H H	$\mathsf{lookup}\ H\ a\ x = g$

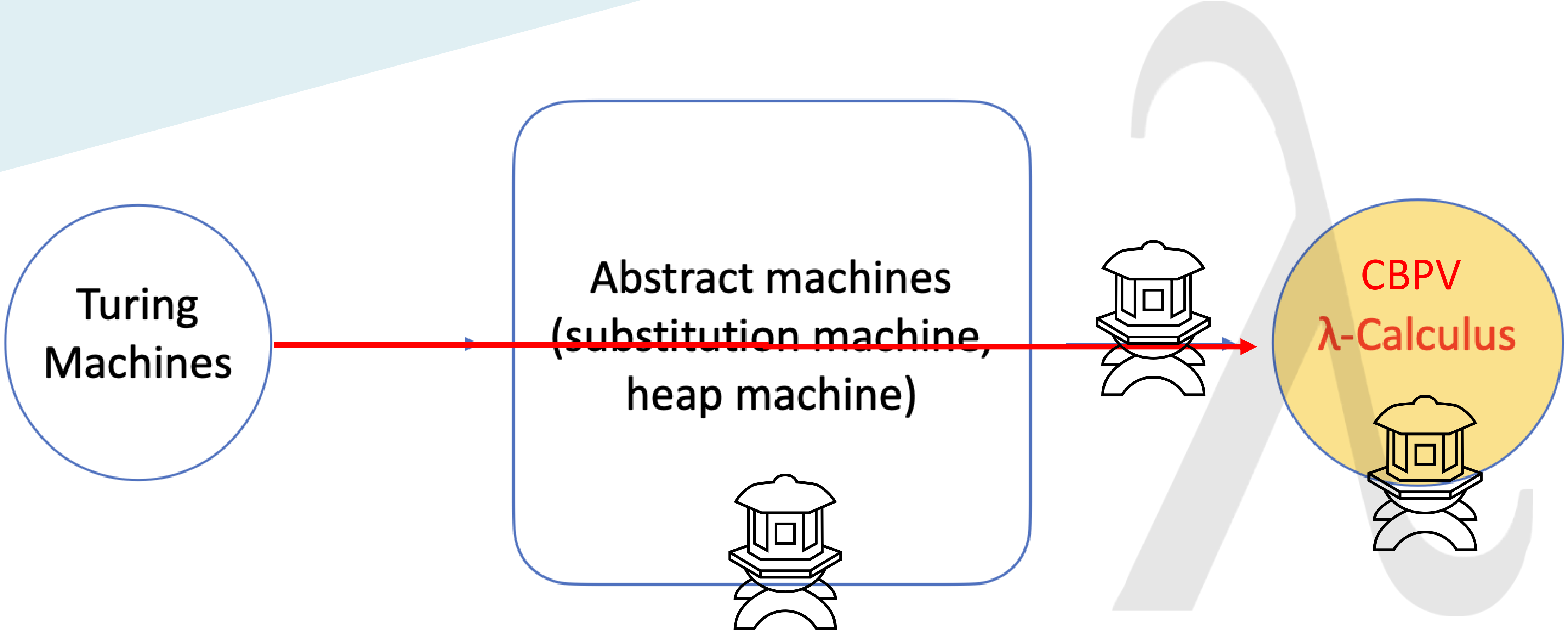
Lemma 55 (Heap Machine Time Simulation). *If $m \Downarrow_k n$ then there exists k' such that $(\langle P_m, 0\rangle, [\,], [\,]) \triangleright_{k'} ([\,], \langle P_n, a\rangle, H)$, where $k' \leq 7 * k + 5$ and $\langle P_m, 0\rangle \gg m$ and $\langle P_n, a\rangle \gg_H n$.*

Lemma 56 (Heap Machine Space Simulation). *If $(P_m, [\,], [\,]) \triangleright_k (T, N, H)$ and $P_m \gg m$ then $\|T +\!\!+ N +\!\!+ H\| \leq (k+1) * (3 * k + 4 * \|m\|)$.*

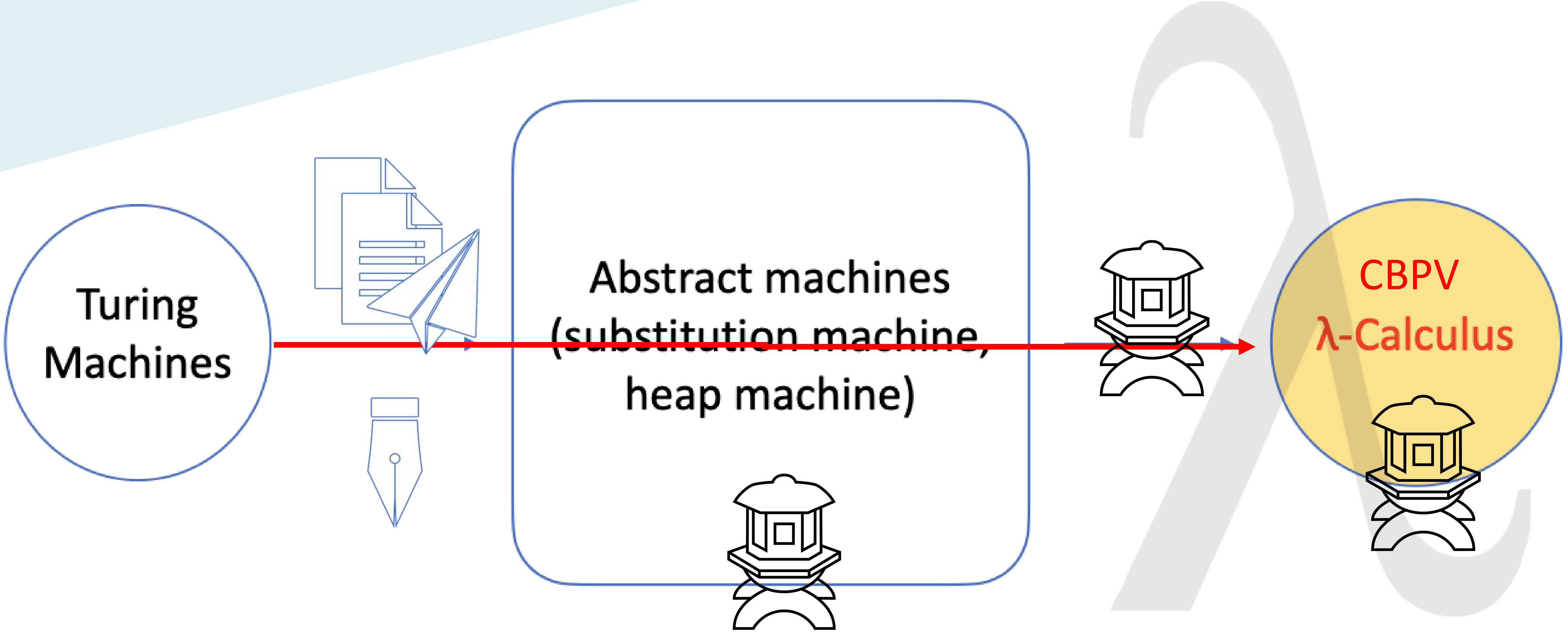
A Verified Cost Model for CBPV



A Verified Cost Model for CBPV



A Verified Cost Model for CBPV



Turing Machine Simulating CBPV

Task Stack	Value Stack	Assumption
$(\mathsf{varT}\ n :: P) :: T$	V	
$\triangleright\ P ::_{tc} T$	$\mathsf{varT}\ n :: V$	
$(\mathsf{thunkT} :: P) :: T$	V	$\varphi P = (M, Q)$
$\triangleright\ Q ::_{tc} T$	$\mathsf{thunkT} :: M \mathbin{+\!\!+} [\mathsf{endThunkT}] :: V$	
$(\mathsf{forceT} :: P) :: T$	$(\mathsf{thunkT} :: K) :: V$	$\varphi K = (M, [\,])$
$\triangleright\ (M \mathbin{+\!\!+} P) ::_{tc} T$	V	
$(\mathsf{lamT} :: P) :: T$	V	$\varphi P = (M, Q)$
$\triangleright\ Q ::_{tc} T$	$(\mathsf{lamT} :: M \mathbin{+\!\!+} [\mathsf{endLamT}]) :: V$	
$(\mathsf{applyT} :: P) :: T$	$Q :: (\mathsf{lamT} :: M \mathbin{+\!\!+} [\mathsf{endLamT}]) :: V$	
$\triangleright\ M^0_Q :: (P ::_{tc} T)$	V	
$(\mathsf{retT} :: P) :: T$	V	$\varphi P = (U, Q)$
$\triangleright\ Q ::_{tc} T$	$(\mathsf{retT} :: U \mathbin{+\!\!+} [\mathsf{endRetT}]) :: V$	
$(\mathsf{seqT} :: P) :: T$	$(\mathsf{retT} :: U \mathbin{+\!\!+} [\mathsf{endRetT}]) :: V$	$\varphi P = (N, Q)$
$\triangleright\ N^0_U :: (Q ::_{tc} T)$	V	
$(\mathsf{pseqT} :: P) :: T$	$(\mathsf{retT} :: U_2 \mathbin{+\!\!+} [\mathsf{endRetT}]) :: (\mathsf{retT} :: U_1 \mathbin{+\!\!+} [\mathsf{endRetT}]) :: V$	$\varphi P = (N, Q)$
$\triangleright\ (N^0_{U_1})^1_{U_2} :: (Q ::_{tc} T)$	V	
$(\mathsf{letinT} :: P) :: T$	$K :: V$	$\varphi P = (M, Q)$
$\triangleright\ M^0_K :: (Q ::_{tc} T)$	V	

FIGURE 4.6: Transition Rules for the Substitution Machine

Turing Machine Simulating CBPV

Task Stack	Value Stack	Assumption
$(\mathsf{varT}\ n :: P) :: T$ $\triangleright\ P ::_{tc} T$	V $\mathsf{varT}\ n :: V$	
$(\mathsf{thunkT} :: P) :: T$ $\triangleright\ Q ::_{tc} T$	V $\mathsf{thunkT} :: M +\!\!+ [\mathsf{endThunkT}] :: V$	$\varphi P = (M, Q)$
$(\mathsf{forceT} :: P) :: T$ $\triangleright\ (M +\!\!+ P) ::_{tc} T$	$(\mathsf{thunkT} :: K) :: V$ V	$\varphi K = (M, [\,])$
$(\mathsf{lamT} :: P) :: T$ $\triangleright\ Q ::_{tc} T$	V $(\mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}]) :: V$	$\varphi P = (M, Q)$
$(\mathsf{applyT} :: P) :: T$ $\triangleright\ M^0_Q :: (P ::_{tc} T)$	$Q :: (\mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}]) :: V$ V	
$(\mathsf{retT} :: P) :: T$ $\triangleright\ Q ::_{tc} T$	V $(\mathsf{retT} :: U +\!\!+ [\mathsf{endRetT}]) :: V$	$\varphi P = (U, Q)$
$(\mathsf{seqT} :: P) :: T$ $\triangleright\ N^0_U :: (Q ::_{tc} T)$	$(\mathsf{retT} :: U +\!\!+ [\mathsf{endRetT}]) :: V$ V	$\varphi P = (N, Q)$
$(\mathsf{pseqT} :: P) :: T$ $\triangleright\ (N^0_{U_1})^1_{U_2} :: (Q ::_{tc} T)$	$(\mathsf{retT} :: U_2 +\!\!+ [\mathsf{endRetT}]) :: (\mathsf{retT} :: U_1 +\!\!+ [\mathsf{endRetT}]) :: V$ V	$\varphi P = (N, Q)$
$(\mathsf{letinT} :: P) :: T$ $\triangleright\ M^0_K :: (Q ::_{tc} T)$	$K :: V$ V	$\varphi P = (M, Q)$

M_1

FIGURE 4.6: Transition Rules for the Substitution Machine

Turing Machine Simulating CBPV

Task Stack	Value Stack	Assumption
$(\mathsf{varT}\ n :: P) :: T$ $\triangleright\ P ::_{tc} T$	V $\mathsf{varT}\ n :: V$	
$(\mathsf{thunkT} :: P) :: T$ $\triangleright\ Q ::_{tc} T$	V $\mathsf{thunkT} :: M +\!\!+ [\mathsf{endThunkT}] :: V$	$\varphi P = (M, Q)$
$(\mathsf{forceT} :: P) :: T$ $\triangleright\ (M +\!\!+ P) ::_{tc} T$	$(\mathsf{thunkT} :: K) :: V$ V	$\varphi K = (M, [\,])$
$(\mathsf{lamT} :: P) :: T$ $\triangleright\ Q ::_{tc} T$	V $(\mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}]) :: V$	$\varphi P = (M, Q)$
$(\mathsf{applyT} :: P) :: T$ $\triangleright\ M^0_Q :: (P ::_{tc} T)$	$Q :: (\mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}]) :: V$ V	
$(\mathsf{retT} :: P) :: T$ $\triangleright\ Q ::_{tc} T$	V $(\mathsf{retT} :: U +\!\!+ [\mathsf{endRetT}]) :: V$	$\varphi P = (U, Q)$
$(\mathsf{seqT} :: P) :: T$ $\triangleright\ N^0_U :: (Q ::_{tc} T)$	$(\mathsf{retT} :: U +\!\!+ [\mathsf{endRetT}]) :: V$ V	$\varphi P = (N, Q)$
$(\mathsf{pseqT} :: P) :: T$ $\triangleright\ (N^0_{U_1})^1_{U_2} :: (Q ::_{tc} T)$	$(\mathsf{retT} :: U_2 +\!\!+ [\mathsf{endRetT}]) :: (\mathsf{retT} :: U_1 +\!\!+ [\mathsf{endRetT}]) :: V$ V	$\varphi P = (N, Q)$
$(\mathsf{letinT} :: P) :: T$ $\triangleright\ M^0_K :: (Q ::_{tc} T)$	$K :: V$ V	$\varphi P = (M, Q)$

M_1

M_2

FIGURE 4.6: Transition Rules for the Substitution Machine

Turing Machine Simulating CBPV

Task Stack	Value Stack	Assumption
$(\mathsf{varT}\ n :: P) :: T$	V	
▷ $P ::_{tc} T$	$\mathsf{varT}\ n :: V$	
$(\mathsf{thunkT} :: P) :: T$	V	$\varphi P = (M, Q)$
▷ $Q ::_{tc} T$	$\mathsf{thunkT} :: M +\!\!+ [\mathsf{endThunkT}] :: V$	
$(\mathsf{forceT} :: P) :: T$	$(\mathsf{thunkT} :: K) :: V$	$\varphi K = (M, [\,])$
▷ $(M +\!\!+ P) ::_{tc} T$	V	
$(\mathsf{lamT} :: P) :: T$	V	$\varphi P = (M, Q)$
▷ $Q ::_{tc} T$	$(\mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}]) :: V$	
$(\mathsf{applyT} :: P) :: T$	$Q :: (\mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}]) :: V$	
▷ $M^0_Q :: (P ::_{tc} T)$	V	
$(\mathsf{retT} :: P) :: T$	V	$\varphi P = (U, Q)$
▷ $Q ::_{tc} T$	$(\mathsf{retT} :: U +\!\!+ [\mathsf{endRetT}]) :: V$	
$(\mathsf{seqT} :: P) :: T$	$(\mathsf{retT} :: U +\!\!+ [\mathsf{endRetT}]) :: V$	$\varphi P = (N, Q)$
▷ $N^0_U :: (Q ::_{tc} T)$	V	
$(\mathsf{pseqT} :: P) :: T$	$(\mathsf{retT} :: U_2 +\!\!+ [\mathsf{endRetT}]) :: (\mathsf{retT} :: U_1 +\!\!+ [\mathsf{endRetT}]) :: V$	$\varphi P = (N, Q)$
▷ $(N^0_{U_1})^1_{U_2} :: (Q ::_{tc} T)$	V	
$(\mathsf{letinT} :: P) :: T$	$K :: V$	$\varphi P = (M, Q)$
▷ $M^0_K :: (Q ::_{tc} T)$	V	

FIGURE 4.6: Transition Rules for the Substitution Machine

M_1

M_2

M_...

Turing Machine Simulating CBPV

Task Stack	Value Stack	Assumption
$(\mathsf{varT}\ n :: P) :: T$ $\triangleright\ P ::_{tc} T$	V $\mathsf{varT}\ n :: V$	
$(\mathsf{thunkT} :: P) :: T$ $\triangleright\ Q ::_{tc} T$	V $\mathsf{thunkT} :: M +\!\!+ [\mathsf{endThunkT}] :: V$	$\varphi P = (M, Q)$
$(\mathsf{forceT} :: P) :: T$ $\triangleright\ (M +\!\!+ P) ::_{tc} T$	$(\mathsf{thunkT} :: K) :: V$ V	$\varphi K = (M, [\,])$
$(\mathsf{lamT} :: P) :: T$ $\triangleright\ Q ::_{tc} T$	V $(\mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}]) :: V$	$\varphi P = (M, Q)$
$(\mathsf{applyT} :: P) :: T$ $\triangleright\ M^0_Q :: (P ::_{tc} T)$	$Q :: (\mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}]) :: V$ V	
$(\mathsf{retT} :: P) :: T$ $\triangleright\ Q ::_{tc} T$	V $(\mathsf{retT} :: U +\!\!+ [\mathsf{endRetT}]) :: V$	$\varphi P = (U, Q)$
$(\mathsf{seqT} :: P) :: T$ $\triangleright\ N^0_U :: (Q ::_{tc} T)$	$(\mathsf{retT} :: U +\!\!+ [\mathsf{endRetT}]) :: V$ V	$\varphi P = (N, Q)$
$(\mathsf{pseqT} :: P) :: T$ $\triangleright\ (N^0_{U_1})^1_{U_2} :: (Q ::_{tc} T)$	$(\mathsf{retT} :: U_2 +\!\!+ [\mathsf{endRetT}]) :: (\mathsf{retT} :: U_1 +\!\!+ [\mathsf{endRetT}]) :: V$ V	$\varphi P = (N, Q)$
$(\mathsf{letinT} :: P) :: T$ $\triangleright\ M^0_K :: (Q ::_{tc} T)$	$K :: V$ V	$\varphi P = (M, Q)$

FIGURE 4.6: Transition Rules for the Substitution Machine

M_{subst}

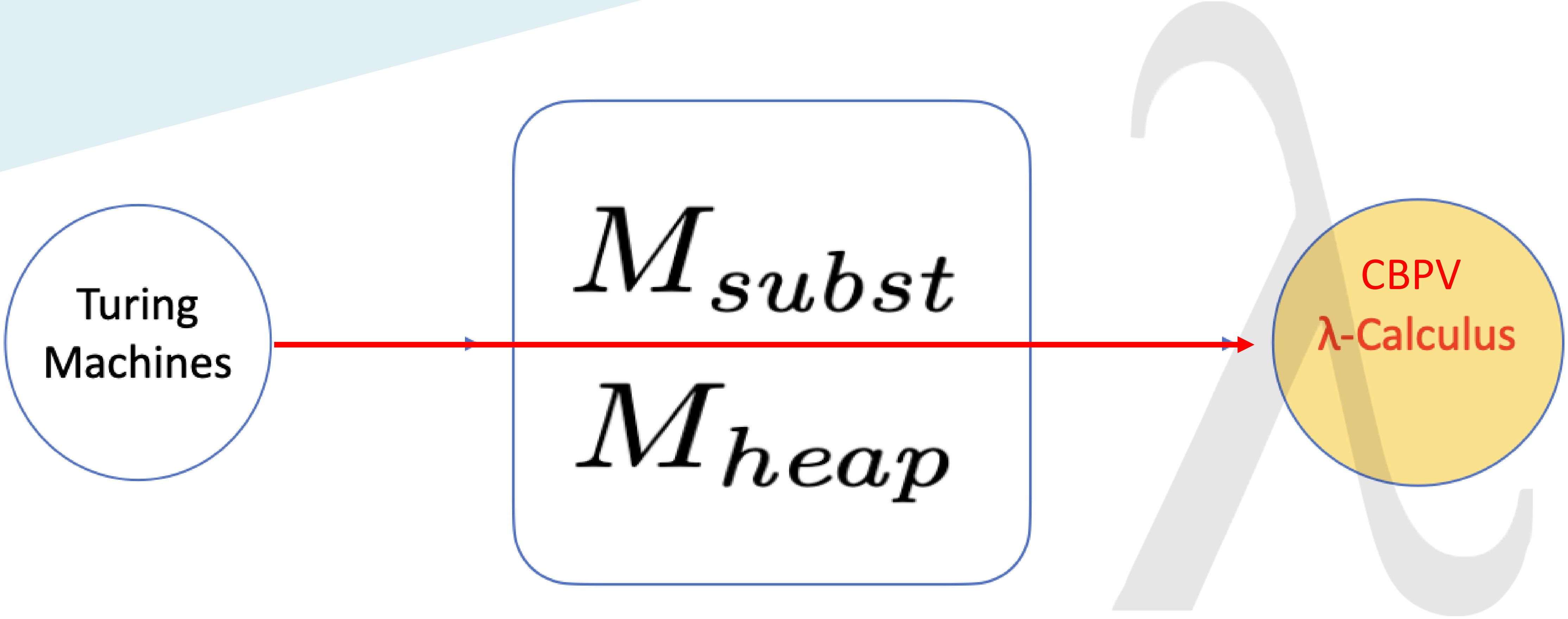
Turing Machine Simulating CBPV

	Task Stack	Value Stack	Heap	Assumption
	$\langle \mathsf{varT}\ x :: P,\ a\rangle :: T$	V	H	lookup $H\ a\ x = g$
$\triangleright$	$\langle P,\ a\rangle :: T$	$g :: V$	H	
	$\langle \mathsf{thunkT} :: P,\ a\rangle :: T$	V	H	$\varphi P = (M, Q)$
$\triangleright$	$\langle Q,\ a\rangle :: T$	$\langle \mathsf{thunkT} :: M +\!\!+ [\mathsf{endThunkT}],\ a\rangle :: V$	H	
	$\langle \mathsf{forceT} :: P,\ a\rangle :: T$	$\langle \mathsf{thunkT} :: M +\!\!+ [\mathsf{endThunkT}],\ b\rangle :: V$	H	
$\triangleright$	$\langle M,\ b\rangle :: \langle P,\ a\rangle :: T$	V	H	
	$\langle \mathsf{lamT} :: P,\ a\rangle :: T$	V	H	$\varphi P = (M, Q)$
$\triangleright$	$\langle Q,\ a\rangle :: T$	$\langle \mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}],\ a\rangle :: V$	H	
	$\langle \mathsf{applyT} :: P,\ a\rangle :: T$	$Q :: \langle \mathsf{lamT} :: M +\!\!+ [\mathsf{endLamT}],\ b\rangle :: V$	H	put $H\{Q,\ b\} = (H', c)$
$\triangleright$	$\langle M,\ c\rangle :: \langle P,\ a\rangle :: T$	V	H'	
	$\langle \mathsf{retT} :: P,\ a\rangle :: T$	$\langle M,\ b\rangle :: V$	H	
$\triangleright$	$\langle P,\ a\rangle :: T$	$\langle M,\ b\rangle :: V$	H	
				$\varphi P = (N, Q)$
	$\langle \mathsf{seqT} :: P,\ a\rangle :: T$	$\langle M,\ b\rangle :: V$	H	put $H\{\langle M,\ b\rangle,\ a\} = (H', c)$
$\triangleright$	$\langle N,\ c\rangle :: \langle Q,\ a\rangle :: T$	V	H'	
				$\varphi P = (N, Q)$
				put $H\{\langle M_2,\ b_2\rangle,\ a\} = (H_1, c_1)$
	$\langle \mathsf{pseqT} :: P,\ a\rangle :: T$	$\langle M_2,\ b_2\rangle :: \langle M_1,\ b_1\rangle :: V$	H	put $H_1\{\langle M_1,\ b_1\rangle,\ a\} = (H_2, c_2)$
$\triangleright$	$\langle N,\ c_2\rangle :: \langle Q,\ a\rangle :: T$	V	H_2	
				$\varphi P = (M, Q)$
	$\langle \mathsf{letinT} :: P,\ a\rangle :: T$	$K :: V$	H	put $H\{K,\ a\} = (H', b)$
$\triangleright$	$\langle M,\ b\rangle :: \langle Q,\ a\rangle :: T$	V	H'	
	$\langle [\,],\ a\rangle :: T$	V	H	
$\triangleright$	T	V	H	

FIGURE 4.8: Transition Rules for the Heap Machine

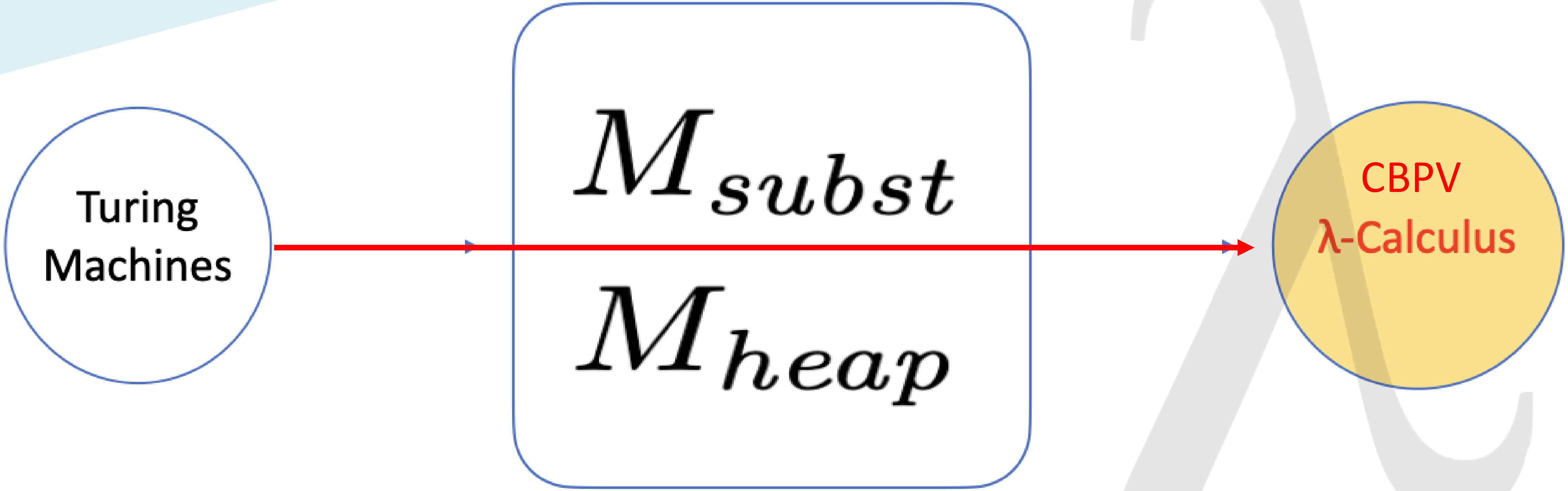
$\rightarrow$ M_{heap}

Turing Machine Simulating CBPV

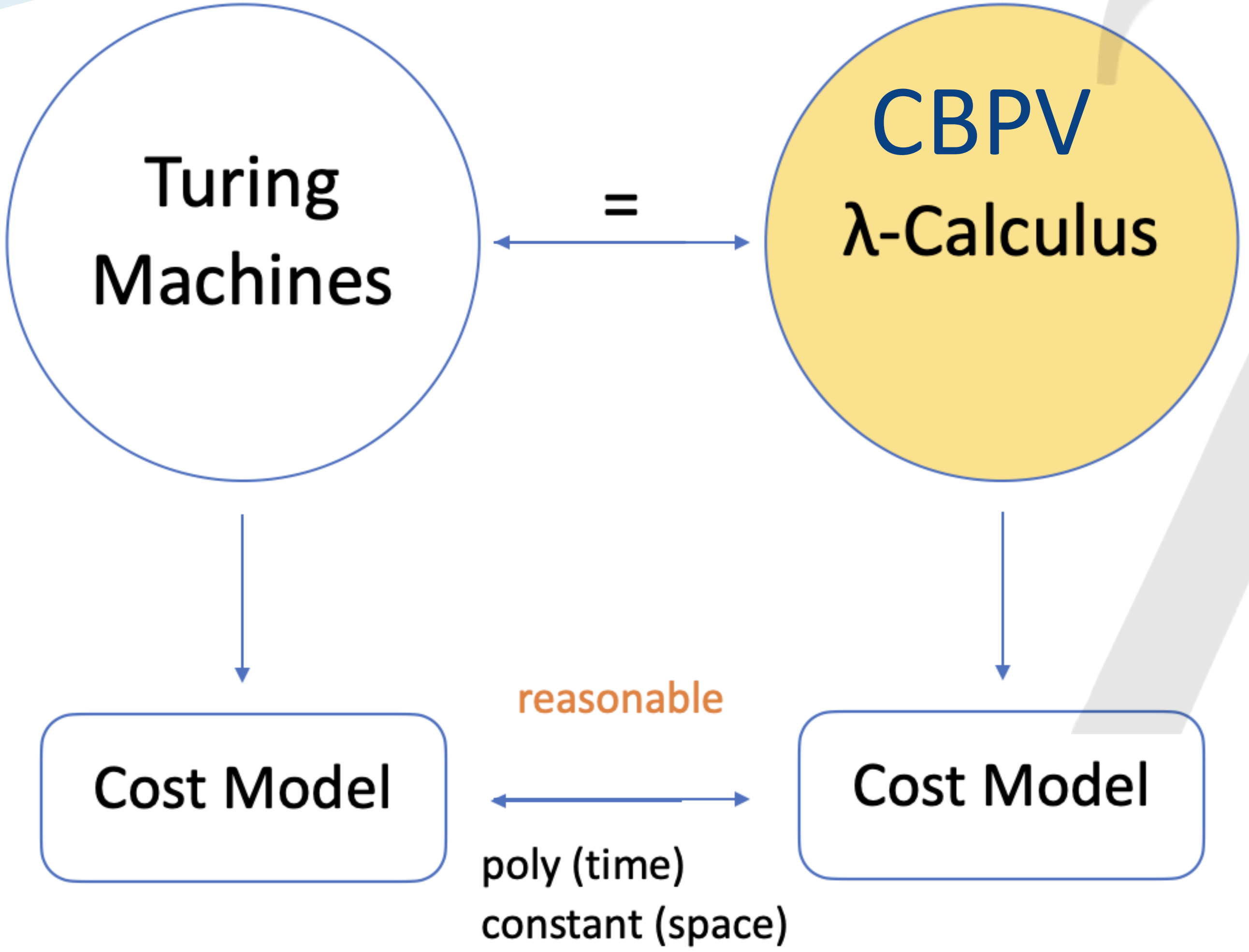


Turing Machine Simulating CBPV

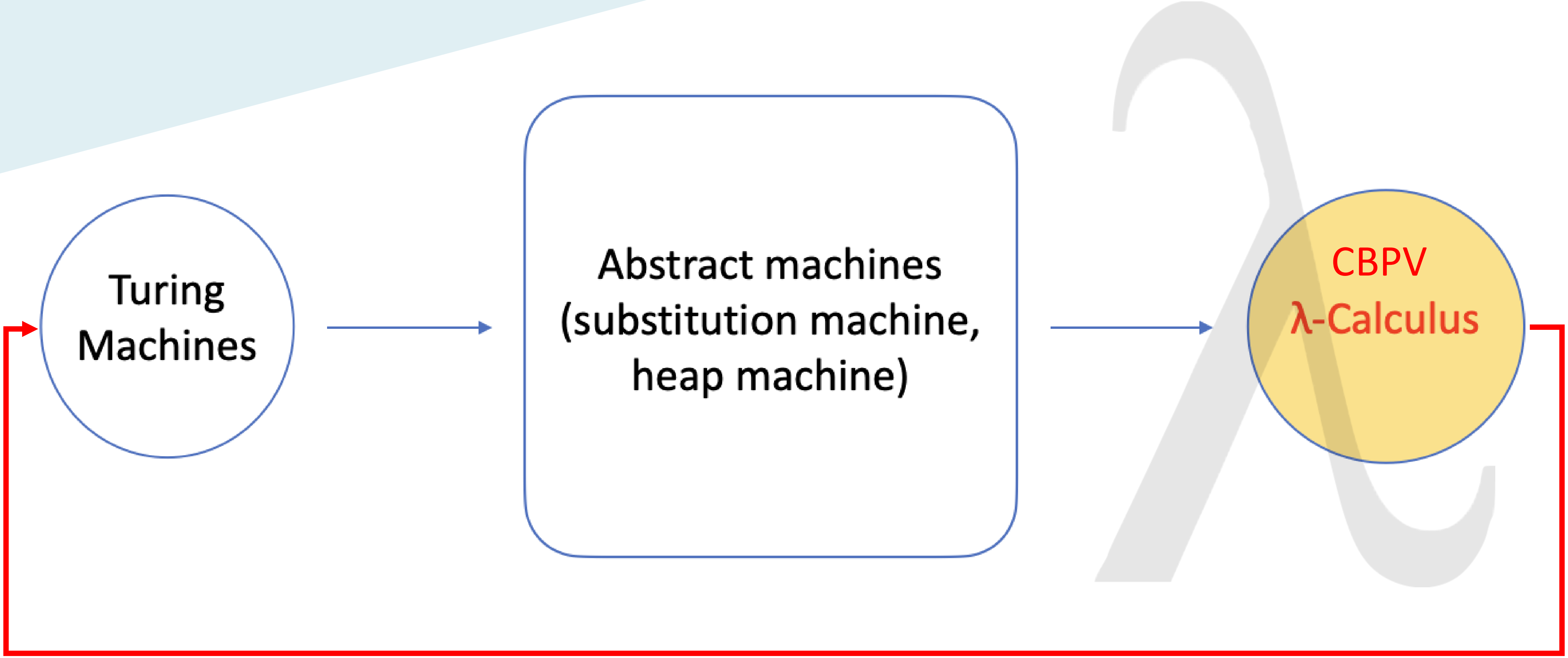
Alternating simulation strategy: [Forster, Kunze, and Roth 2020]



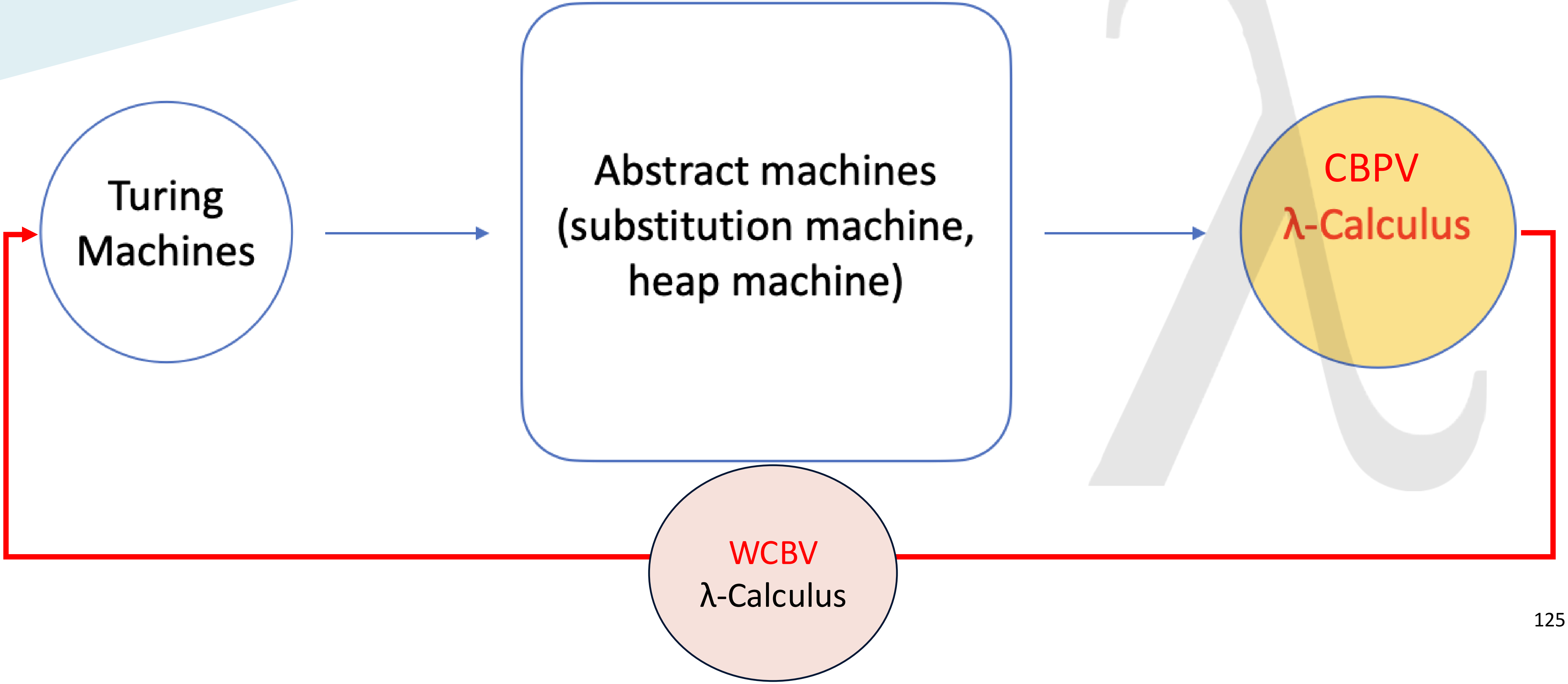
A Verified Cost Model for CBPV



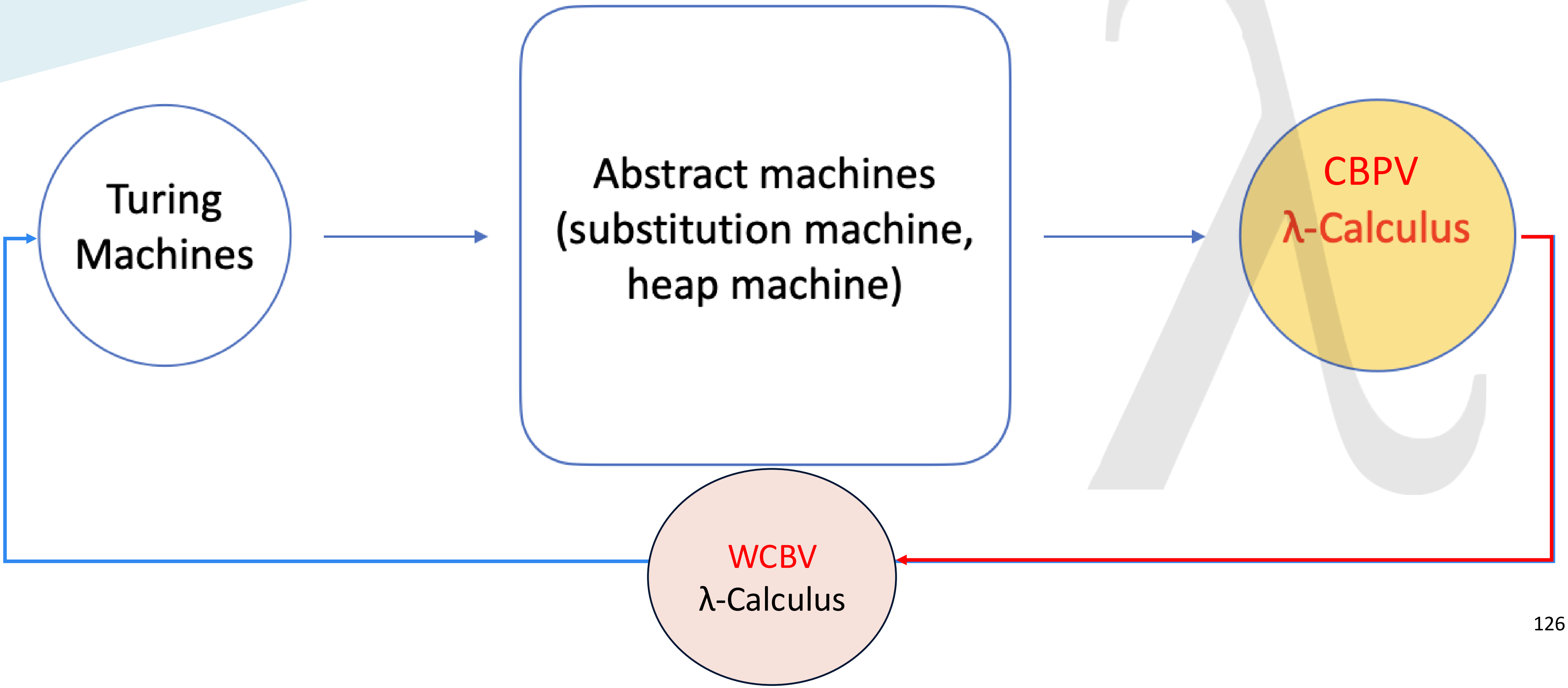
CBPV Simulating Turing Machines



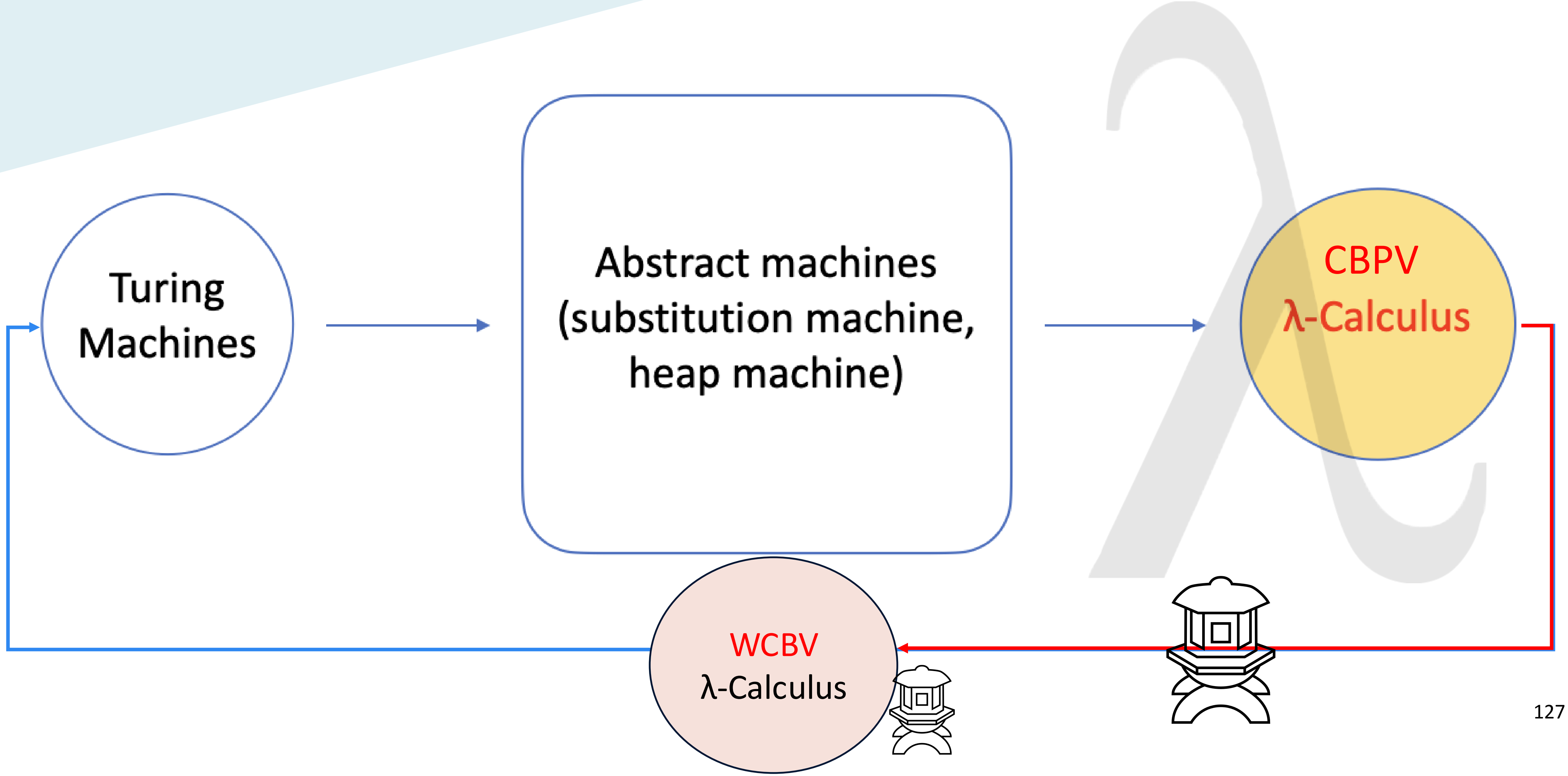
CBPV Simulating Turing Machines



CBPV Simulating Turing Machines



CBPV Simulating Turing Machines



CBPV Simulating WCBV

Definition 18 (Syntax for WCBV). *n is a constant.*

$$s, t, u, v = \mathsf{var}\ n \mid \mathsf{app}\ s\ t \mid \lambda.\ s$$

CBPV Simulating WCBV

Definition 18 (Syntax for WCBV). *n is a constant.*

$$s, t, u, v = \mathsf{var}\, n \,|\, \mathsf{app}\, s\, t \,|\, \lambda.\, s$$

Compilation Function:

$$c(\mathsf{var}\, x) \;=\; \mathsf{var}\, x \qquad c(\lambda.\, t) \;=\; \mathsf{ret\, thunk}\, \lambda.\, c(t)$$

$$c(\mathsf{app}\, t\, u) \;=\; \mathsf{pseq}\, (c(u))\, (c(t))\, (\mathsf{app}\, (\mathsf{force\, var}\, 0)\, (\mathsf{var}\, 1))$$

CBPV Simulating WCBV

Definition 18 (Syntax for WCBV). *n is a constant.*

$$s, t, u, v = \mathsf{var}\ n \mid \mathsf{app}\ s\ t \mid \lambda. s$$

Compilation Function:

$$\begin{array}{lcllcl} c(\mathsf{var}\ x) & = & \mathsf{var}\ x & c(\lambda. t) & = & \mathsf{ret\ thunk}\ \lambda. c(t) \\ c(\mathsf{app}\ t\ u) & = & \multicolumn{4}{l}{\mathsf{pseq}\ (c(u))\ (c(t))\ (\mathsf{app}\ (\mathsf{force\ var}\ 0)\ (\mathsf{var}\ 1))} \end{array}$$

Theorem 62. *For each closed WCBV term t, we have:*

1. *If $t \Downarrow_k u$, then there exists k' such that $c(t) \Downarrow_{k'} c(u)$ and $k' \leq 5 * k$; and*
2. *If $t \Downarrow^s u$, then there exists s' such that $c(t) \Downarrow^{s'} c(u)$ and $s' \leq 6 * s$.*

CBPV Simulating WCBV

Definition 18 (Syntax for WCBV). *n is a constant.*

$$s, t, u, v = \mathsf{var}\, n | \mathsf{app}\, s\, t | \lambda. s$$

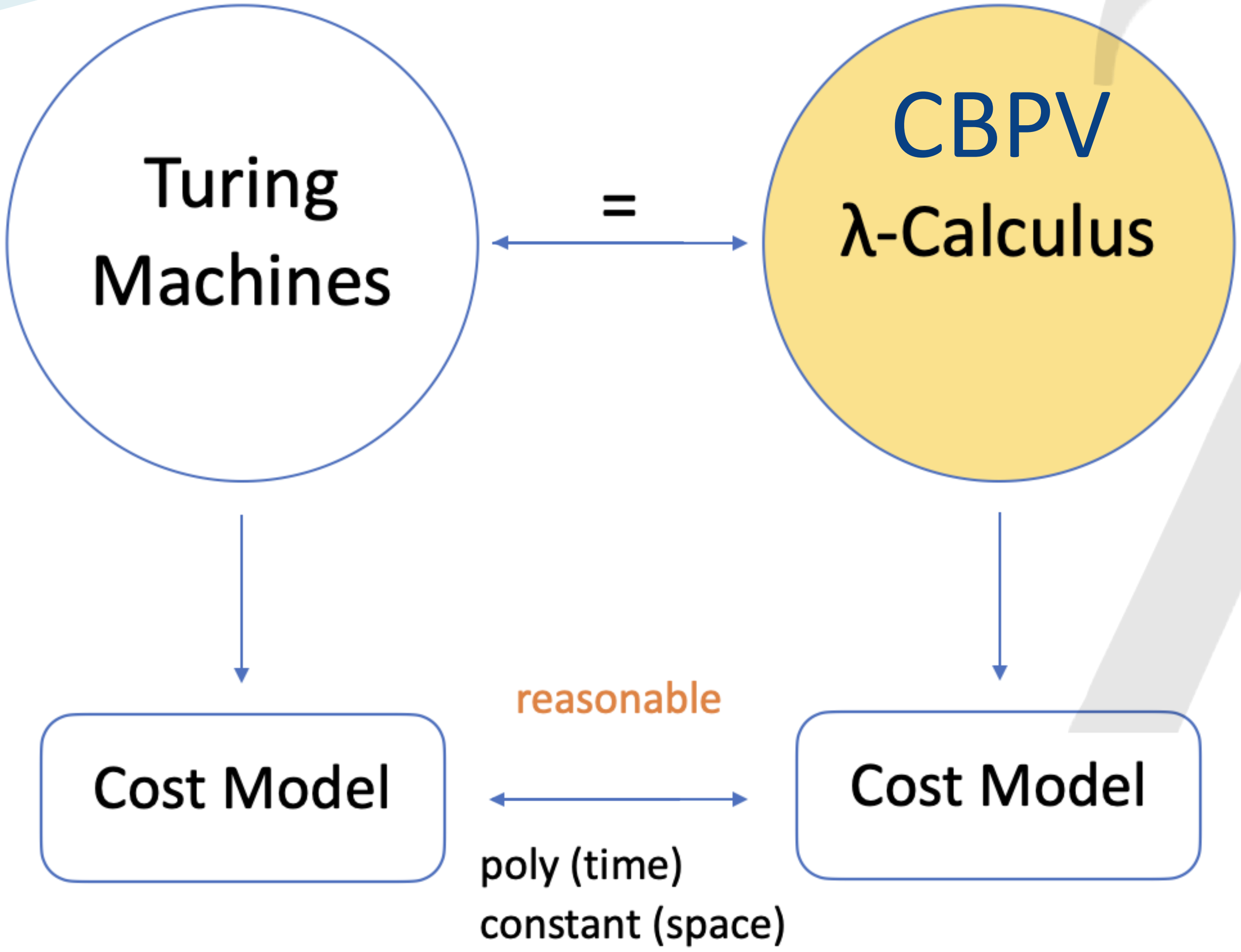
Compilation Function:

$$\begin{aligned} c(\mathsf{var}\, x) &= \mathsf{var}\, x & c(\lambda. t) &= \mathsf{ret}\,\mathsf{thunk}\, \lambda. c(t) \\ c(\mathsf{app}\, t\, u) &= \mathsf{pseq}\, (c(u))\, (c(t))\, (\mathsf{app}\, (\mathsf{force}\,\mathsf{var}\, 0)\, (\mathsf{var}\, 1)) \end{aligned}$$

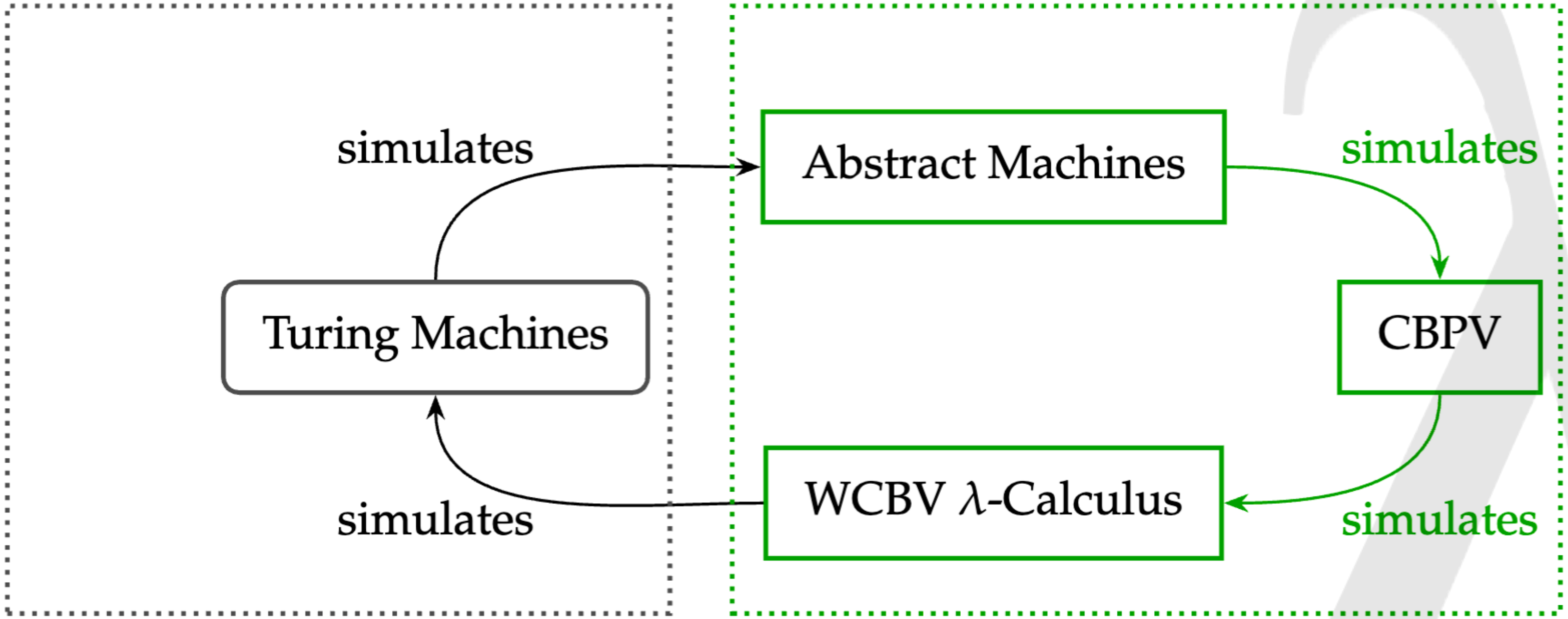
Theorem 62. *For each closed WCBV term t, we have:*

1. *If $t \Downarrow_k u$, then there exists k' such that $c(t) \Downarrow_{k'} c(u)$ and $k' \leq 5 * k$; and*
2. *If $t \Downarrow^s u$, then there exists s' such that $c(t) \Downarrow^{s'} c(u)$ and $s' \leq 6 * s$.*

A Verified Cost Model for CBPV



A Verified Cost Model for CBPV



- Adapted from: Forster, Kunze, and Roth (2020).
- Mechanised in this thesis

FIGURE 4.10: Simulation between Turing Machines and CBPV with intermediate models

Thank you

Questions?

Formalisation available at:

https://github.com/ZhuoZoeyChen/cbpv-reasonable-HOL/

References

Accattoli, B., 2017. (in)efficiency and reasonable cost models. In 12th Workshop on Logical and Semantic Frameworks, with Applications, LSFA 2017, Bras ́ılia, Brazil, September 23-24, 2017, vol. 338 of Electronic Notes in Theoretical Computer Science, 23–43. Elsevier. doi: 10.1016/j.entcs. 2018.10.003. URL https://doi.org/10.1016/j.entcs.2018.10.003.

Accattoli, B. and dal Lago, U., 2016. (leftmost-outermost) beta reduc- tion is invariant, indeed. Log. Methods Comput. Sci., 12, 1 (2016). doi: 10. 2168/LMCS-12(1:4)2016. URL https://doi.org/10.2168/LMCS-12(1: 4)2016.

Accattoli, B. and Lago, U. D., 2012. On the invariance of the uni- tary cost model for head reduction. In 23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan, vol. 15 of LIPIcs, 22–37. Schloss Dagstuhl - Leibniz-Zentrum fu ̈r Informatik. doi: 10.4230/LIPIcs.RTA.2012.22. URL https://doi.org/10.4230/LIPIcs.RTA.2012.22.

Blelloch, G. E. and Greiner, J., 1995. Parallelism in sequential functional languages. In Proceedings of the seventh international con- ference on Functional programming languages and computer architecture, FPCA 1995, La Jolla, California, USA, June 25-28, 1995, 226–237. ACM. doi: 10.1145/224164.224210. URL https://doi.org/10.1145/224164. 224210.

Bucciarelli, A.; Kesner, D.; and Ventura, D., 2017. Non- idempotent intersection types for the lambda-calculus. Log. J. IGPL, 25, 4 (2017), 431–464. doi: 10.1093/jigpal/jzx018. URL https://doi.org/ 10.1093/jigpal/jzx018.

Church, A., 1932. A set of postulates for the foundation of logic. Annals 18 of Mathematics, 33, 2 (1932), 346–366. URL http://www.jstor.org/ stable/1968337.

dal Lago, U. and Martini, S., 2008. The weak lambda calculus as a reasonable machine. Theor. Comput. Sci., 398, 1-3 (2008), 32–50. doi: 10.1016/j.tcs.2008.01.044. URL https://doi.org/10.1016/j.tcs.2008. 01.044.

de Carvalho, D., 2018. Execution time of λ-terms via denotational se- mantics and intersection types. Math. Struct. Comput. Sci., 28, 7 (2018), 1169–1203. doi: 10.1017/S0960129516000396. URL https://doi.org/ 10.1017/S0960129516000396.

Ehrhard, T. and Regnier, L., 2006. Bo ̈hm trees, krivine's machine and the taylor expansion of lambda-terms. In Logical Approaches to Computa- tional Barriers, Second Conference on Computability in Europe, CiE 2006, Swansea, UK, June 30-July 5, 2006, Proceedings, vol. 3988 of Lecture Notes in Computer Science, 186–197. Springer. doi: 10.1007/11780342\ 20. URL https://doi.org/10.1007/11780342_20.

Forster, Y.; Kunze, F.; and Roth, M., 2020a. The weak call-by-value λ-calculus is reasonable for both time and space. Proc. ACM Program. Lang., 4, POPL (2020), 27:1–27:23. doi: 10.1145/3371095. URL https: //doi.org/10.1145/3371095.

Forster, Y.; Kunze, F.; Smolka, G.; and Wuttke, M., 2021. A mechanised proof of the time invariance thesis for the weak call-by- value λ-calculus. In 12th International Conference on Interactive Theo- rem Proving, ITP 2021, June 29 to July 1, 2021, Rome, Italy (Virtual Conference), vol. 193 of LIPIcs, 19:1–19:20. Schloss Dagstuhl - Leibniz- Zentrum fu ̈r Informatik. doi: 10.4230/LIPIcs.ITP.2021.19. URL https: //doi.org/10.4230/LIPIcs.ITP.2021.19.

Forster, Y.; Kunze, F.; and Wuttke, M., 2020b. Verified program- ming of turing machines in coq. In Proceedings of the 9th ACM SIG- PLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020, 114–128. ACM. doi: 10.1145/3372885.3373816. URL https://doi.org/10.1145/3372885. 3373816.

References

Gordon, M. J. C. and Melham, T. F., 1993. Introduction to HOL, a theorem proving environment for higher order logic. Cambridge University Press.

Hudak, P.; Hughes, J.; Jones, S. L. P.; and Wadler, P., 2007. A history of haskell: being lazy with class. In Proceedings of the Third ACM SIGPLAN History of Programming Languages Conference (HOPL-III), San Diego, California, USA, 9-10 June 2007, 1–55. ACM. doi: 10.1145/ 1238844.1238856. URL https://doi.org/10.1145/1238844.1238856.

Kumar, R.; Myreen, M. O.; Norrish, M.; and Owens, S., 2014. Cakeml: a verified implementation of ml. ACM SIGPLAN Notices, 49, 1 (2014). doi: 10.1145/2578855.2535841.

Kunze, F.; Smolka, G.; and Forster, Y., 2018. Formal small-step ver- ification of a call-by-value lambda calculus machine. In Programming Lan- guages and Systems - 16th Asian Symposium, APLAS 2018, Wellington, New Zealand, December 2-6, 2018, Proceedings, vol. 11275 of Lecture Notes in Computer Science, 264–283. Springer. doi: 10.1007/978-3-030-02768-1\ _15. URL https://doi.org/10.1007/978-3-030-02768-1_15.

Lago, U. D. and Accattoli, B., 2017. Encoding turing machines into the deterministic lambda-calculus. CoRR, abs/1711.10078 (2017). URL http://arxiv.org/abs/1711.10078.

Levy, P. B., 1999. Call-by-push-value: A subsuming paradigm. In Typed 20

Lambda Calculi and Applications, 4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999, Proceedings, vol. 1581 of Lecture Notes in Computer Science, 228–242. Springer. doi: 10.1007/3-540-48959-2\ 17. URL https://doi.org/10.1007/3-540-48959-2_17.

Milner, R.; Tofte, M.; and Harper, R., 1990. Definition of standard ML. MIT Press. ISBN 978-0-262-63132-7.

Nipkow, T. and Klein, G., 2014. Concrete Semantics - With Isabelle/HOL. Springer. ISBN 978-3-319-10541-3. doi: 10.1007/978-3-319-10542-0. URL https://doi.org/10.1007/ 978-3-319-10542-0.

Nipkow, T.; Paulson, L. C.; and Wenzel, M., 2002. Isabelle/HOL - A Proof Assistant for Higher-Order Logic, vol. 2283 of Lecture Notes in Com- puter Science. Springer. ISBN 3-540-43376-7. doi: 10.1007/3-540-45949-9. URL https://doi.org/10.1007/3-540-45949-9.

Paulson, L. C., 1986. Natural deduction as higher-order resolution. J. Log. Program., 3, 3 (1986), 237–258. doi: 10.1016/0743-1066(86)90015-4. URL https://doi.org/10.1016/0743-1066(86)90015-4.

Slind, K. and Norrish, M., 2008. A brief overview of HOL4. In Theorem Proving in Higher Order Logics, 21st International Confer- ence, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceed- ings, vol. 5170 of Lecture Notes in Computer Science, 28–32. Springer. doi: 10.1007/978-3-540-71067-7\ 6. URL https://doi.org/10.1007/ 978-3-540-71067-7_6.

Turing, A. M. et al., 1936. On computable numbers, with an application to the entscheidungsproblem. J. of Math, 58, 345-363 (1936), 5.

Method - Why CBPV?

Evaluation Strategies for λ-Calculus

evaluates first

Evaluates first

call by value

fst(1+2, 1+3)
= fst(3, 1+3)
= fst(3, 4)
= 3
Evaluates the function argument before passing them into the function call

substitutes first

Evaluates after

call by name

fst(1+2, 1+3)
= 1+2
= 3
Copying the function argument into the function body without evaluating it first

lazy

Evaluates when needed

call by need

fst(1+2, 1+3)
1. let x = 1+2
    y = 1+3
  in fst(x, y)
2. let x = 1+2
    y = 1+3
  in x
3. let x = 3 in x
4. 3

weak

reduction doesn't go under abstraction

strong

reduction does go under abstraction

Method - Why CBPV?

fst(1+2, 1+3)
= fst(3, 1+3)
= fst(3, 4)
= 3

call by value

fst(1+2, 1+3)
= 1+2
= 3

call by name

fst(1+2, 1+3)
1. let x = 1+2
   y = 1+3
   in fst(x, y)
2. let x = 1+2
   y = 1+3
   in x
3. let x = 3 in x
4. 3

call by need

Evaluation Strategies for λ-Calculus

weak

reduction doesn't go under abstraction

strong