Guarded Kleene Algebra with Tests

Verification of Uninterpreted Programs in Nearly Linear Time

Tobias Kappé

Cornell University

Programming Languages Seminar, TU Delft

Joint work with. . .

Steffen Smolka

Nate Foster

Justin Hsu

Dexter Kozen

Alexandra Silva

Published @ POPL 2020; see `arxiv.org/abs/1907.05920`.

Introduction

```
while a and b do
    e;
end
while a do
    f;
    while a and b do
        e;
    end
end
```

Introduction

```
while a and b do
    e;
end
while a do
    f;
    while a and b do
        e;
    end
end
```

```
while a do
    if b then
        e;
    else
        f;
    end
end
```

Introduction

```
while a and b do
    e;
end
while a do
    f;
    while a and b do
        e;
    end
end
```

$\stackrel{?}{\equiv}$

```
while a do
    if b then
        e;
    else
        f;
    end
end
```

Introduction

Contributions:

- Nearly linear time decision procedure for equivalence.[1]

[1] For fixed number of tests.

Introduction

Contributions:

- Nearly linear time decision procedure for equivalence.[1]
- Axiomatization of uninterpreted program equivalence.

[1] For fixed number of tests.

Introduction

Contributions:

- Nearly linear time decision procedure for equivalence.[1]
- Axiomatization of uninterpreted program equivalence.
- Kleene theorem for uninterpreted programs.

[1] For fixed number of tests.

Syntax

$$\mathbf{a}, \mathbf{b} ::= t \in T \mid \mathbf{a} + \mathbf{b} \mid \mathbf{ab} \mid \overline{\mathbf{a}} \mid 0 \mid 1$$

$$\mathbf{e}, \mathbf{f} ::= \mathbf{a} \mid p \in \Sigma \mid \mathbf{ef} \mid \mathbf{e} +_{\mathbf{a}} \mathbf{f} \mid \mathbf{e}^{(\mathbf{a})}$$

Syntax

$$\mathsf{a}, \mathsf{b} ::= t \in T \mid \mathsf{a} + \mathsf{b} \mid \mathsf{ab} \mid \overline{\mathsf{a}} \mid 0 \mid 1$$

a or b

$$\mathsf{e}, \mathsf{f} ::= \mathsf{a} \mid p \in \Sigma \mid \mathsf{ef} \mid \mathsf{e} +_{\mathsf{a}} \mathsf{f} \mid \mathsf{e}^{(\mathsf{a})}$$

Syntax

$$\mathbf{a}, \mathbf{b} ::= t \in T \mid \mathbf{a} + \mathbf{b} \mid \mathbf{ab} \mid \overline{\mathbf{a}} \mid 0 \mid 1$$

a and b

$$\mathbf{e}, \mathbf{f} ::= \mathbf{a} \mid p \in \Sigma \mid \mathbf{ef} \mid \mathbf{e} +_{\mathbf{a}} \mathbf{f} \mid \mathbf{e}^{(\mathbf{a})}$$

Syntax

$$\mathbf{a}, \mathbf{b} ::= t \in T \mid \mathbf{a} + \mathbf{b} \mid \mathbf{ab} \mid \overline{\mathbf{a}} \mid 0 \mid 1$$

not **a**

$$\mathbf{e}, \mathbf{f} ::= \mathbf{a} \mid p \in \Sigma \mid \mathbf{ef} \mid \mathbf{e} +_{\mathbf{a}} \mathbf{f} \mid \mathbf{e}^{(\mathbf{a})}$$

Syntax

$$\mathbf{a}, \mathbf{b} ::= t \in T \mid \mathbf{a} + \mathbf{b} \mid \mathbf{ab} \mid \overline{\mathbf{a}} \mid 0 \mid 1$$

false

$$\mathbf{e}, \mathbf{f} ::= \mathbf{a} \mid p \in \Sigma \mid \mathbf{ef} \mid \mathbf{e} +_{\mathbf{a}} \mathbf{f} \mid \mathbf{e}^{(\mathbf{a})}$$

Syntax

$$\mathbf{a}, \mathbf{b} ::= t \in T \mid \mathbf{a} + \mathbf{b} \mid \mathbf{ab} \mid \overline{\mathbf{a}} \mid 0 \mid 1$$

true

$$\mathbf{e}, \mathbf{f} ::= \mathbf{a} \mid p \in \Sigma \mid \mathbf{ef} \mid \mathbf{e} +_{\mathbf{a}} \mathbf{f} \mid \mathbf{e}^{(\mathbf{a})}$$

Syntax

$$\mathsf{a}, \mathsf{b} ::= t \in T \mid \mathsf{a} + \mathsf{b} \mid \mathsf{ab} \mid \overline{\mathsf{a}} \mid 0 \mid 1$$

$$\mathsf{e}, \mathsf{f} ::= \mathsf{a} \mid p \in \Sigma \mid \mathsf{ef} \mid \mathsf{e} +_{\mathsf{a}} \mathsf{f} \mid \mathsf{e}^{(\mathsf{a})}$$

`assert` a

Syntax

$$\mathbf{a}, \mathbf{b} ::= t \in T \mid \mathbf{a} + \mathbf{b} \mid \mathbf{ab} \mid \overline{\mathbf{a}} \mid 0 \mid 1$$

$$\mathbf{e}, \mathbf{f} ::= \mathbf{a} \mid p \in \Sigma \mid \mathbf{ef} \mid \mathbf{e} +_{\mathbf{a}} \mathbf{f} \mid \mathbf{e}^{(\mathbf{a})}$$

$\mathbf{e}; \mathbf{f}$

Syntax

$$\mathsf{a}, \mathsf{b} ::= t \in T \mid \mathsf{a} + \mathsf{b} \mid \mathsf{ab} \mid \overline{\mathsf{a}} \mid 0 \mid 1$$

$$\mathsf{e}, \mathsf{f} ::= \mathsf{a} \mid p \in \Sigma \mid \mathsf{ef} \mid \mathsf{e} +_{\mathsf{a}} \mathsf{f} \mid \mathsf{e}^{(\mathsf{a})}$$

`if a then e else f`

Syntax

$$\mathbf{a}, \mathbf{b} ::= t \in T \mid \mathbf{a} + \mathbf{b} \mid \mathbf{ab} \mid \overline{\mathbf{a}} \mid 0 \mid 1$$

$$\mathbf{e}, \mathbf{f} ::= \mathbf{a} \mid p \in \Sigma \mid \mathbf{ef} \mid \mathbf{e} +_{\mathbf{a}} \mathbf{f} \mid \mathbf{e}^{(\mathbf{a})}$$

`while` **a** `do` **e**

Syntax

```
while a do
    if b then
        e;
    else
        f;
    end
end
```

$$\Downarrow$$

$$(e +_b f)^{(a)}$$

```
while a and b do
    e;
end
while a do
    f;
    while a and b do
        e;
    end
end
```

$$\Downarrow$$

$$e^{(ab)}(fe^{(ab)})^{(a)}$$

Semantics / relational

$$sat : T \to 2^{States}$$

Semantics / relational

$$sat : T \to 2^{States}$$

$$eval : \Sigma \to States \to 2^{States}$$

Semantics / relational

$$sat : T \to 2^{States}$$

$$eval : \Sigma \to States \to 2^{States}$$

$$i = (sat, eval)$$

Semantics / relational

$$sat : T \to 2^{States}$$

$$eval : \Sigma \to States \to 2^{States}$$

$$i = (sat, eval)$$

$$\mathcal{R}_i[\![\mathbf{e}]\!] : States \to 2^{States}$$

Semantics / uninterpreted

$$Atoms = 2^{T} \qquad \mathrm{GS}(\Sigma, T) = Atoms \cdot (\Sigma \cdot Atoms)^{*}$$

$$\mathit{Atoms} = 2^T \qquad \mathrm{GS}(\Sigma, T) = \mathit{Atoms} \cdot (\Sigma \cdot \mathit{Atoms})^*$$

$$L \diamond K = \{w\alpha x : w\alpha \in L, \alpha x \in K\} \qquad L^{(n)} = \underbrace{L \diamond \cdots \diamond L}_{n \text{ times}} \qquad L^{(*)} = \bigcup_{n \in \mathbb{N}} L^{(n)}$$

Semantics / uninterpreted

e	$[\![e]\!]$
$t \in T$	$\{\alpha \in \mathit{Atoms} : t \in \alpha\}$
$a + b$	$[\![a]\!] \cup [\![b]\!]$
ab	$[\![a]\!] \cap [\![b]\!]$
$\bar{a}$	$\mathit{Atoms} \setminus [\![a]\!]$
$p \in \Sigma$	$\{\alpha p \beta : \alpha, \beta \in \mathit{Atoms}\}$
$e +_a f$	$[\![a]\!] \diamond [\![e]\!] \cup [\![\bar{a}]\!] \diamond [\![f]\!]$
ef	$[\![e]\!] \diamond [\![f]\!]$
$e^{(a)}$	$([\![a]\!] \diamond [\![e]\!])^{(*)} \diamond [\![\bar{a}]\!]$

Decision procedure

Theorem

$$[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!] \iff \forall i.\ \mathcal{R}_i[\![\mathbf{e}]\!] = \mathcal{R}_i[\![\mathbf{f}]\!]$$

Decision procedure

Theorem

$$[\![e]\!] = [\![f]\!] \iff \forall i.\ \mathcal{R}_i[\![e]\!] = \mathcal{R}_i[\![f]\!]$$

How to check $[\![e]\!] = [\![f]\!]$:

1. Create automata that accept $[\![e]\!]$ and $[\![f]\!]$ [Thompson 1968]
2. Check automata for bisimilarity [Hopcroft and Karp 1971; Tarjan 1975]

Decision procedure

Theorem

$$[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!] \iff \forall i.\ \mathcal{R}_i[\![\mathbf{e}]\!] = \mathcal{R}_i[\![\mathbf{f}]\!]$$

How to check $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$:

1. Create automata that accept $[\![\mathbf{e}]\!]$ and $[\![\mathbf{f}]\!]$ [Thompson 1968]
2. Check automata for bisimilarity [Hopcroft and Karp 1971; Tarjan 1975]

Decidability

Axiomatization / without loops

$$e +_a e \equiv e$$

Axiomatization / without loops

$e +_a e \equiv e \qquad e +_a f \equiv f +_{\bar{a}} e$

$$e +_a e \equiv e \qquad e +_a f \equiv f +_{\bar{a}} e \qquad e +_a f \equiv ae +_a f$$

$$e +_a e \equiv e \qquad e +_a f \equiv f +_{\bar{a}} e \qquad e +_a f \equiv ae +_a f \qquad \bar{a}a \equiv 0$$

Axiomatization / without loops

$$e +_a e \equiv e \qquad e +_a f \equiv f +_{\bar{a}} e \qquad e +_a f \equiv ae +_a f \qquad \bar{a}a \equiv 0 \qquad 0e \equiv 0$$

Axiomatization / without loops

$$e +_a e \equiv e \qquad e +_a f \equiv f +_{\bar{a}} e \qquad e +_a f \equiv ae +_a f \qquad \bar{a}a \equiv 0 \qquad 0e \equiv 0$$

Example

$\texttt{if } a \texttt{ then } e \texttt{ else assert false} = e +_a 0$

Axiomatization / without loops

$e +_a e \equiv e \qquad e +_a f \equiv f +_{\bar{a}} e \qquad \boxed{e +_a f \equiv ae +_a f} \qquad \bar{a}a \equiv 0 \qquad 0e \equiv 0$

Example

$\texttt{if } a \texttt{ then } e \texttt{ else assert false} = e +_a 0 \equiv ae +_a 0$

Axiomatization / without loops

$e +_a e \equiv e$ $\quad$ $e +_a f \equiv f +_{\bar{a}} e$ $\quad$ $e +_a f \equiv ae +_a f$ $\quad$ $\bar{a}a \equiv 0$ $\quad$ $0e \equiv 0$

Example

$$\texttt{if } a \texttt{ then } e \texttt{ else assert false} = e +_a 0 \equiv ae +_a 0$$
$$\equiv 0 +_{\bar{a}} ae$$

Axiomatization / without loops

$$e +_a e \equiv e \qquad e +_a f \equiv f +_{\bar{a}} e \qquad e +_a f \equiv ae +_a f \qquad \bar{a}a \equiv 0 \qquad \boxed{0e \equiv 0}$$

Example

$$\begin{aligned}\texttt{if } a \texttt{ then } e \texttt{ else assert false} = e +_a 0 &\equiv ae +_a 0 \\ &\equiv 0 +_{\bar{a}} ae \\ &\equiv 0e +_{\bar{a}} ae\end{aligned}$$

Axiomatization / without loops

$$e +_a e \equiv e \qquad e +_a f \equiv f +_{\overline{a}} e \qquad e +_a f \equiv ae +_a f \qquad \boxed{\overline{a}a \equiv 0} \qquad 0e \equiv 0$$

Example

$$\begin{aligned}\texttt{if } a \texttt{ then } e \texttt{ else assert false} = e +_a 0 &\equiv ae +_a 0 \\ &\equiv 0 +_{\overline{a}} ae \\ &\equiv 0e +_{\overline{a}} ae \\ &\equiv \overline{a}ae +_{\overline{a}} ae\end{aligned}$$

Axiomatization / without loops

$$e +_a e \equiv e \qquad e +_a f \equiv f +_{\bar{a}} e \qquad \boxed{e +_a f \equiv ae +_a f} \qquad \bar{a}a \equiv 0 \qquad 0e \equiv 0$$

Example

$$\begin{aligned}
\texttt{if } a \texttt{ then } e \texttt{ else assert false} = e +_a 0 &\equiv ae +_a 0 \\
&\equiv 0 +_{\bar{a}} ae \\
&\equiv 0e +_{\bar{a}} ae \\
&\equiv \bar{a}ae +_{\bar{a}} ae \\
&\equiv ae +_{\bar{a}} ae
\end{aligned}$$

Axiomatization / without loops

$\boxed{e +_a e \equiv e}$ $\quad e +_a f \equiv f +_{\bar{a}} e \quad e +_a f \equiv ae +_a f \quad \bar{a}a \equiv 0 \quad 0e \equiv 0$

Example

$$
\begin{aligned}
\texttt{if } a \texttt{ then } e \texttt{ else assert false} = e +_a 0 &\equiv ae +_a 0 \\
&\equiv 0 +_{\bar{a}} ae \\
&\equiv 0e +_{\bar{a}} ae \\
&\equiv \bar{a}ae +_{\bar{a}} ae \\
&\equiv ae +_{\bar{a}} ae \\
&\equiv ae \qquad\qquad = \texttt{assert } a; e
\end{aligned}
$$

Axiomatization / with loops

$$\frac{e \equiv fe +_{a} g}{e \equiv f^{(a)} g}$$

$$\frac{e \equiv fe +_a g}{e \equiv f^{(a)}g}$$

Allows to derive $1 \equiv 1^{(1)}$, i.e.,

`assert` **true** $\equiv$ `while` **true** `do assert` **true**

Axiomatization / with loops

$$\frac{e \equiv fe +_a g \qquad f \text{ is productive}}{e \equiv f^{(a)} g}$$

$$\frac{e \equiv fe +_a g \qquad f \text{ is productive}}{e \equiv f^{(a)}g}$$

$$e^{(a)} \equiv ee^{(a)} +_a 1$$

Axiomatization / with loops

$$\frac{e \equiv fe +_{a} g \qquad f \text{ is productive}}{e \equiv f^{(a)} g}$$

$$e^{(a)} \equiv ee^{(a)} +_{a} 1$$

$$(e +_{a} 1)^{(b)} \equiv (ae)^{(b)}$$

$$\frac{e \equiv fe +_a g \qquad f \text{ is productive}}{e \equiv f^{(a)}g} \qquad e^{(a)} \equiv ee^{(a)} +_a 1 \qquad (e +_a 1)^{(b)} \equiv (ae)^{(b)}$$

Lemma

For every e, *there exists a productive* $\hat{e}$ *such that* $e^{(b)} \equiv \hat{e}^{(b)}$.

$$\frac{e \equiv fe +_a g \qquad f \text{ is productive}}{e \equiv f^{(a)} g} \qquad e^{(a)} \equiv ee^{(a)} +_a 1 \qquad (e +_a 1)^{(b)} \equiv (ae)^{(b)}$$

Lemma

For every e, *there exists a productive* $\hat{e}$ *such that* $e^{(b)} \equiv \hat{e}^{(b)}$.

Lemma

$$e^{(a)} \equiv e^{(a)}\overline{a} \qquad e^{(a)} \equiv (ae)^{(a)} \qquad e^{(ab)}e^{(b)} \equiv e^{(b)}$$

Theorem (Soundness)

If $\mathbf{e} \equiv \mathbf{f}$, *then* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$.

Theorem (Soundness)

If $\mathbf{e} \equiv \mathbf{f}$*, then* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$.

How about the converse?

1. $A \mapsto S(A)$ and $\mathbf{e} \mapsto A_{\mathbf{e}}$ with
$$\mathbf{e} \equiv S(A_{\mathbf{e}})$$
2. If $A \sim A'$, then $S(A) \equiv S(A')$.

Theorem (Soundness)

If $\mathbf{e} \equiv \mathbf{f}$*, then* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$.

How about the converse?

1. $A \mapsto S(A)$ and $\mathbf{e} \mapsto A_{\mathbf{e}}$ with
$$\mathbf{e} \equiv S(A_{\mathbf{e}})$$
2. If $A \sim A'$, then $S(A) \equiv S(A')$.

$$\begin{aligned} [\![\mathbf{e}]\!] = [\![\mathbf{f}]\!] &\implies L(A_{\mathbf{e}}) = L(A_{\mathbf{f}}) \\ &\implies A_{\mathbf{e}} \sim A_{\mathbf{f}} \\ &\implies S(A_{\mathbf{e}}) \equiv S(A_{\mathbf{f}}) \\ &\implies \mathbf{e} \equiv \mathbf{f} \end{aligned}$$

Theorem (Soundness)

If $\mathbf{e} \equiv \mathbf{f}$, *then* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$.

Theorem (Completeness)

If $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$, *then* $\mathbf{e} \equiv \mathbf{f}$.

How about the converse?

1. $A \mapsto S(A)$ and $\mathbf{e} \mapsto A_{\mathbf{e}}$ with

$$\mathbf{e} \equiv S(A_{\mathbf{e}})$$

2. If $A \sim A'$, then $S(A) \equiv S(A')$.

$$\begin{aligned} [\![\mathbf{e}]\!] = [\![\mathbf{f}]\!] &\implies L(A_{\mathbf{e}}) = L(A_{\mathbf{f}}) \\ &\implies A_{\mathbf{e}} \sim A_{\mathbf{f}} \\ &\implies S(A_{\mathbf{e}}) \equiv S(A_{\mathbf{f}}) \\ &\implies \mathbf{e} \equiv \mathbf{f} \end{aligned}$$

Theorem (Soundness)

If $e \equiv f$, *then* $[\![e]\!] = [\![f]\!]$.

Theorem (Completeness)

If $[\![e]\!] = [\![f]\!]$, *then* $e \equiv f$.

Axiomatization

How about the converse?

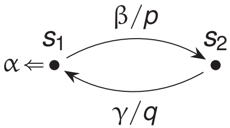
1. $A \mapsto S(A)$ and $e \mapsto A_e$ with

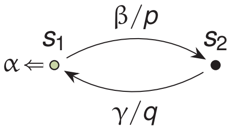
$$e \equiv S(A_e)$$

2. If $A \sim A'$, then $S(A) \equiv S(A')$.

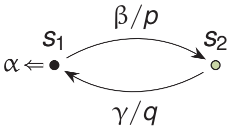
$$\begin{aligned}[\![e]\!] = [\![f]\!] &\implies L(A_e) = L(A_f) \\ &\implies A_e \sim A_f \\ &\implies S(A_e) \equiv S(A_f) \\ &\implies e \equiv f\end{aligned}$$

A Kleene theorem / automata model



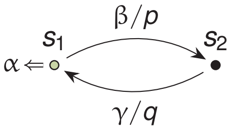

β / p
s1
s2
α ⇐
γ / q

A Kleene theorem / automata model



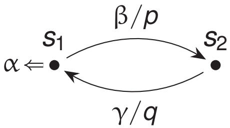
βp

A Kleene theorem / automata model



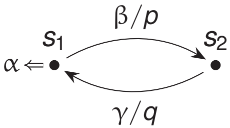
$\beta p \gamma q$

A Kleene theorem / automata model



$\beta p \gamma q \alpha \in L(A)$

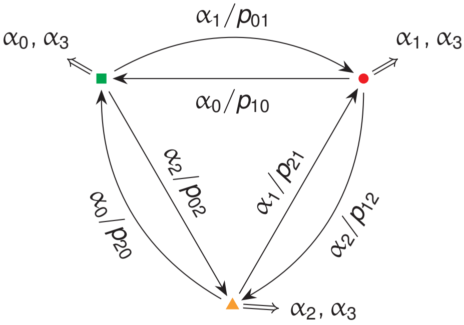
A Kleene theorem / automata model



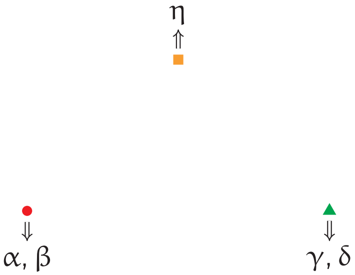
$\beta p \gamma q \alpha \in L(A)$

$$(X, \delta : X \to (2 + \Sigma \times X)^{Atoms})$$

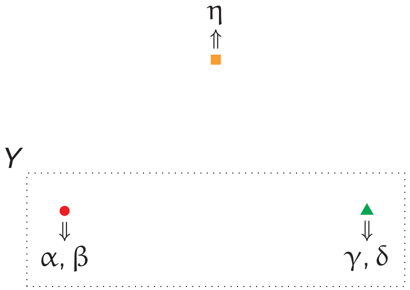
Not described by an expression **e**:

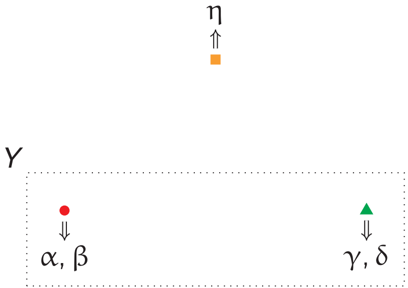


See [Kozen and Tseng 2008].

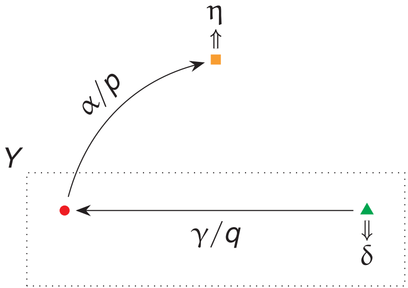

η
α, β
γ, δ

A Kleene theorem / automata model





$$h : \mathit{Atoms} \to 2 + \Sigma \times X$$
$$h(\alpha) = (p, \blacksquare)$$
$$h(\beta) = 0$$
$$h(\gamma) = (q, \bullet)$$
$$h(\delta) = 1$$
$$h(-) = 0$$



$h : Atoms \to 2 + \Sigma \times X$

$h(\alpha) = (p, ■)$

$h(\beta) = 0$

$h(\gamma) = (q, ●)$

$h(\delta) = 1$

$h(-) = 0$

A Kleene theorem / main result

Theorem

Let L be a language of guarded strings. The following are equivalent:

1. $L = [\![\mathbf{e}]\!]$ *for some* $\mathbf{e}$.
2. *L is accepted by a well-nested and finite automaton A.*

A Kleene theorem / main result

Theorem

Let L be a language of guarded strings. The following are equivalent:

1. $L = [\![\mathbf{e}]\!]$ *for some* $\mathbf{e}$.
2. *L is accepted by a well-nested and finite automaton A.*

- Both conversions are constructive.
- Automata are linear in size of expression.
- Side-conditions for completeness also hold.

A Kleene theorem / main result

Theorem

Let L be a language of guarded strings. The following are equivalent:

1. *$L = [\![\mathbf{e}]\!]$ for some $\mathbf{e}$.*
2. *L is accepted by a well-nested and finite automaton A.*

- Both conversions are constructive.
- Automata are linear in size of expression.
- Side-conditions for completeness also hold.

Kleene Theorem

Further work

- Which theories could we embed while keeping decidability?
- Are more parameterized semantics possible?
- How do we recover a small program from an automaton?
- Which extensions of the syntax would be interesting?

kap.pe/slides

arxiv.org/abs/1907.05920