Guarded Kleene Algebra with Tests

Verification of Uninterpreted Programs in Nearly Linear Time

Tobias Kappé

Cornell University

Programming Languages Discussion Group

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

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

a or 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

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

a and 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

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

not a

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

Syntax

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

`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

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

`true`

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

Syntax

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

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

`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

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

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

`if` **a** `then` **e** `else` **f**

Syntax

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

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

`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
```

$$\Downarrow$$

$$e^{(ab)}\left(fe^{(ab)}\right)^{(a)}$$

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

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

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

$$i = (sat, eval)$$

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

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

Semantics / probabilistic

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

$$eval : \Sigma \to States \to \mathcal{D}(States)$$

Semantics / probabilistic

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

$$eval : \Sigma \to States \to \mathcal{D}(States)$$

$$i = (sat, eval)$$

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

$$eval : \Sigma \to States \to \mathcal{D}(States)$$

$$i = (sat, eval)$$

$$\mathcal{P}_i[\![\mathbf{e}]\!] : States \to \mathcal{D}(States)$$

Semantics / uninterpreted

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

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

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

Semantics / uninterpreted

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

Theorem

The following are equivalent:

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

Theorem

The following are equivalent:

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

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

1. Create automata that accept $[\![\mathbf{e}]\!]$ and $[\![\mathbf{f}]\!]$.
2. Check automata for bisimilarity.

Axiomatization / without loops

$$e +_a e \equiv e$$

Axiomatization / without loops

$e +_{a} e \equiv e \qquad e +_{a} f \equiv f +_{\overline{a}} e$

Axiomatization / without loops

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

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

Axiomatization / without loops

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

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

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

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

Example

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

Axiomatization / without loops

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

Example

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

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

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

Example

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

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

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

Axiomatization / with loops

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

Axiomatization / with loops

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

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

$\texttt{assert}\ \textbf{true} \equiv \texttt{while}\ \textbf{true}\ \texttt{do}\ \texttt{assert}\ \textbf{true}$

Axiomatization / with loops

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

Axiomatization / with loops

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

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

Axiomatization / with loops

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

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

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

Axiomatization / with loops

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

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

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

Lemma

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

Axiomatization / with loops

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

Lemma

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

Lemma

$e^{(a)}$ $(ae)^{(a)}$

Axiomatization / with loops

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

Lemma

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

Lemma

$$e^{(a)} \equiv \hat{e}^{(a)} \qquad (ae)^{(a)}$$

Axiomatization / with loops

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

Lemma

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

Lemma

$$e^{(a)} \equiv \hat{e}^{(a)} \equiv \hat{e}\hat{e}^{(a)} +_a 1 \qquad (ae)^{(a)}$$

Axiomatization / with loops

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

Lemma

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

Lemma

$$e^{(a)} \equiv \hat{e}^{(a)} \equiv \hat{e}\hat{e}^{(a)} +_a 1 \equiv a\hat{e}\hat{e}^{(a)} +_a 1 \qquad (ae)^{(a)}$$

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

Lemma

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

Lemma

$$e^{(a)} \equiv \hat{e}^{(a)} \equiv \hat{e}\hat{e}^{(a)} +_a 1 \equiv a\hat{e}\hat{e}^{(a)} +_a 1 \equiv \widehat{ae}\hat{e}^{(a)} +_a 1 \qquad (ae)^{(a)}$$

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

Lemma

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

Lemma

$$e^{(a)} \equiv \hat{e}^{(a)} \equiv \hat{e}\hat{e}^{(a)} +_a 1 \equiv a\hat{e}\hat{e}^{(a)} +_a 1 \equiv \widehat{ae}\hat{e}^{(a)} +_a 1 \equiv (\widehat{ae})^{(a)} \equiv (ae)^{(a)}$$

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)

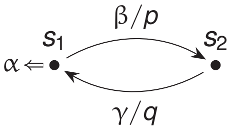
If $\mathbf{e} \equiv \mathbf{f}$*, then* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$.

How about the converse?

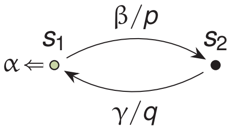
1. $A \mapsto S(A)$ and $\mathbf{e} \mapsto A_{\mathbf{e}}$ with $\mathbf{e} \equiv S(A_{\mathbf{e}})$.
2. If $A \sim A'$, then $S(A) \equiv S(A')$.

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

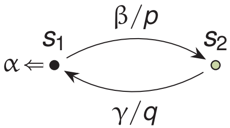
A Kleene theorem / automata model



A Kleene theorem / automata model

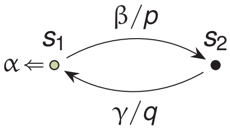


A Kleene theorem / automata model



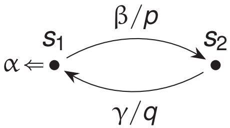
βp

A Kleene theorem / automata model



$\beta p \gamma q$

A Kleene theorem / automata model



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

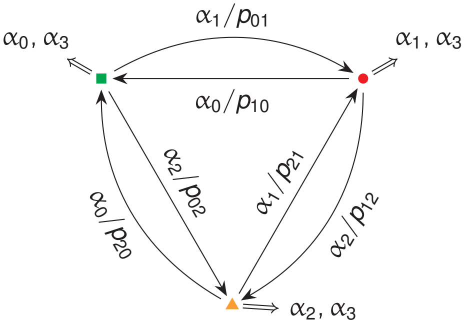
A Kleene theorem / automata model

$$\begin{tikzcd} \alpha \Leftarrow s_1 \arrow[r, bend left, "\beta / p"] & s_2 \arrow[l, bend left, "\gamma / q"] \end{tikzcd}$$

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

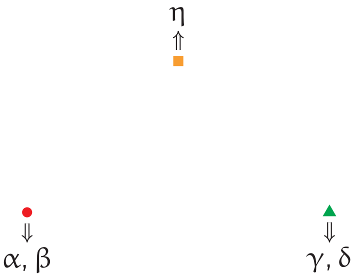
$$(X, \delta : X \to (2 + \Sigma \times X)^{Atoms})$$

Not described by an expression **e**:

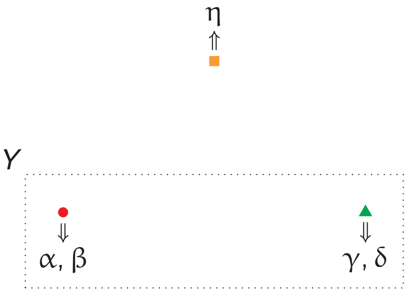


See [Kozen and Tseng 2008].

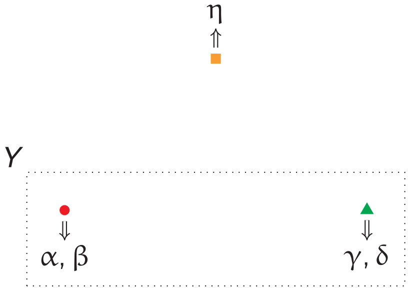
A Kleene theorem / automata model



A Kleene theorem / automata model

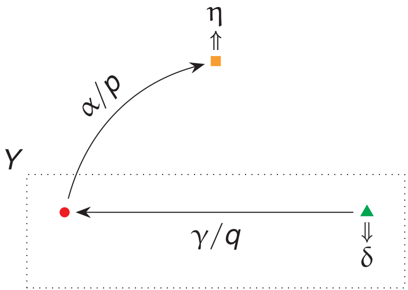


A Kleene theorem / automata model



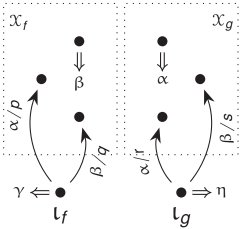
$$h : Atoms \to 2 + \Sigma \times X$$
$$h(\alpha) = (p, \blacksquare)$$
$$h(\beta) = 0$$
$$h(\gamma) = (q, \bullet)$$
$$h(\delta) = 1$$
$$h(-) = 0$$

NB: This slide had the wrong signature for *h* when I gave the talk; this is fixed now.



$$h : Atoms \to 2 + \Sigma \times X$$
$$h(\alpha) = (p, ■)$$
$$h(\beta) = 0$$
$$h(\gamma) = (q, ●)$$
$$h(\delta) = 1$$
$$h(-) = 0$$

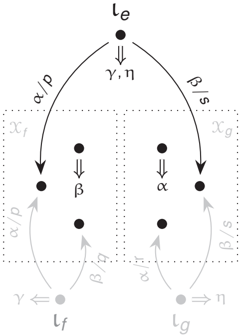
NB: This slide had the wrong signature for *h* when I gave the talk; this is fixed now.



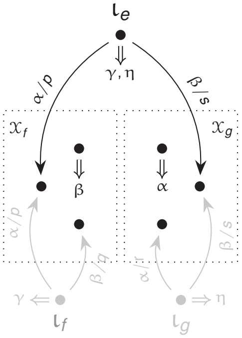
$\mathcal{X}_f$
$\mathcal{X}_g$
$\Downarrow$
β
α
α / p
β / q
α / r
β / s
$\gamma \Leftarrow$
$\Rightarrow \eta$
ι_f
ι_g

$$e = f +_a g$$

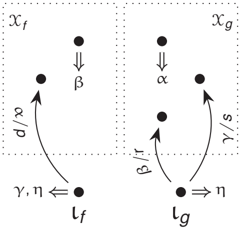
A Kleene theorem / expressions to automata



$$e = f +_a g$$

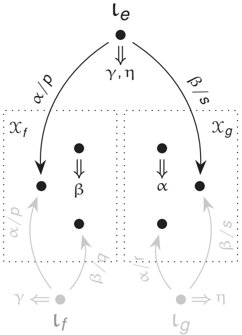


$\mathbf{e} = \mathbf{f} +_{\mathbf{a}} \mathbf{g}$

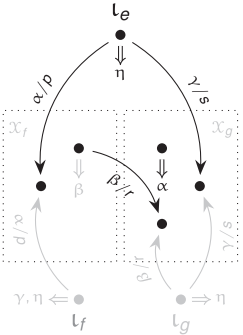


$\mathbf{e} = \mathbf{fg}$

A Kleene theorem / expressions to automata

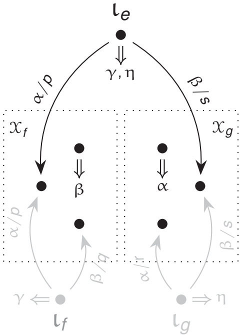


$\mathbf{e} = \mathbf{f} +_{\mathbf{a}} \mathbf{g}$

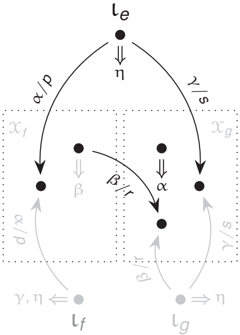


$\mathbf{e} = \mathbf{fg}$

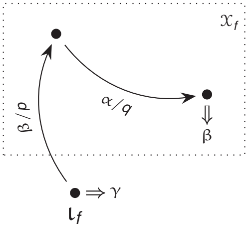
A Kleene theorem / expressions to automata



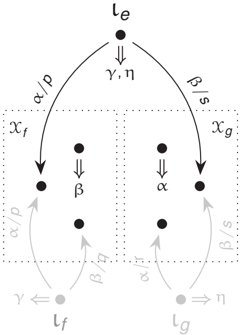
$\mathbf{e} = \mathbf{f} +_{\mathbf{a}} \mathbf{g}$



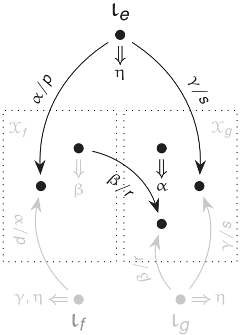
$\mathbf{e} = \mathbf{fg}$



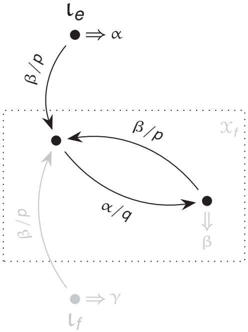
$\mathbf{e} = \mathbf{f}^{(\mathbf{a})}$



$\mathbf{e} = \mathbf{f} +_{\mathbf{a}} \mathbf{g}$

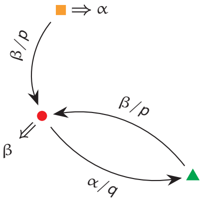


$\mathbf{e} = \mathbf{fg}$

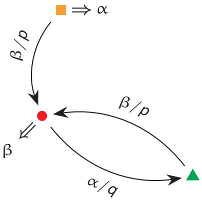


$\mathbf{e} = \mathbf{f}^{(\mathbf{a})}$

A Kleene theorem / automata to expressions

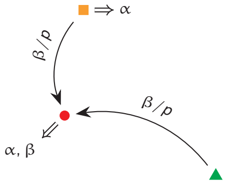


A Kleene theorem / automata to expressions



$$
\begin{aligned}
x_{\blacksquare} &\equiv 1 && +_\alpha\ p \cdot x_{\bullet} && +_\beta\ 0 \\
x_{\bullet} &\equiv q \cdot x_{\blacktriangle} && +_\alpha\ 1 && +_\beta\ 0 \\
x_{\blacktriangle} &\equiv 0 && +_\alpha\ p \cdot x_{\bullet} && +_\beta\ 0
\end{aligned}
$$

A Kleene theorem / automata to expressions



$$
\begin{aligned}
x_{\blacksquare} &\equiv 1 &&+_\alpha p \cdot x_{\bullet} &&+_\beta 0 \\
x_{\bullet} &\equiv 1 &&+_\alpha 1 &&+_\beta 0 \\
x_{\blacktriangle} &\equiv 0 &&+_\alpha p \cdot x_{\bullet} &&+_\beta 0
\end{aligned}
$$

Theorem

Let $L \subseteq \mathrm{GS}(\Sigma, T)$. The following are equivalent:

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

Discussion

- Which theories could we embed while keeping decidability?
- Parameterised semantics besides relational and probabilistic?
- 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