Guarded Kleene Algebra with Tests

Verification of Uninterpreted Programs in Nearly Linear Time

Steffen Smolka[1] Nate Foster[1] Justin Hsu[2]
Tobias Kappé[3] Dexter Kozen[1] Alexandra Silva[3]

[1]Cornell University

[2]University of Wisconsin-Madison

[3]University College London

POPL 2020

Introduction

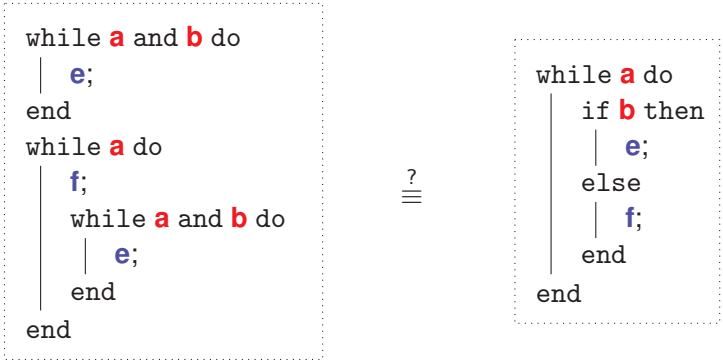
```
while a and b do
  | e;
end
while a do
  | f;
  | while a and b do
  |   | e;
  | end
end
```

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



Introduction

	KAT
Composition	choice, iteration [Kozen 1996]
Complexity	PSPACE-hard [Kozen and Smith 1996]
Axiomatization	Quasi-equational [Kozen and Smith 1996]
Automata	Automata on guarded strings [Kozen 2003; Kozen and Tseng 2008]

See also Ashcroft and Manna 1972; Böhm and Jacopini 1966; Kosaraju 1973; Oulsnam 1982; Peterson et al. 1973; Ramshaw 1988; Williams and Ossher 1978; Hendren et al. 1992; Morris et al. 1997

Introduction

	KAT	$\supseteq$	GKAT
Composition	choice, iteration		`if-then-else`, `while-do`
	[Kozen 1996]		
Complexity	PSPACE-hard		Nearly linear
	[Kozen and Smith 1996]		
Axiomatization	Quasi-equational		Quasi-equational†
	[Kozen and Smith 1996]		
Automata	Automata on guarded strings		Well-nested fragment
	[Kozen 2003; Kozen and Tseng 2008]		

See also Ashcroft and Manna 1972; Böhm and Jacopini 1966; Kosaraju 1973; Oulsnam 1982; Peterson et al. 1973; Ramshaw 1988; Williams and Ossher 1978; Hendren et al. 1992; Morris et al. 1997

Contributions

Nearly linear **decision procedure**.

Contributions

Nearly linear **decision procedure**.

Quasi-equational **axiomatization**.

Contributions

Nearly linear **decision procedure**.

Quasi-equational **axiomatization**.

Automata model with **Kleene Theorem**.

Syntax

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

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

Syntax

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

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

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

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

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

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

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

`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

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

`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

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

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

`assert` $\mathbf{a}$

Syntax

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

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

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

Syntax

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

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

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

Syntax

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

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

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

Syntax

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

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

Syntax

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

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

Syntax

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

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

Syntax

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

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

Semantics

Relational Semantics

$$i = \left(\mathit{sat} : T \to 2^{\mathit{States}}, \mathit{eval} : \Sigma \to 2^{\mathit{States}^2} \right)$$

Relational Semantics

$$i = \left(sat : T \to 2^{States}, eval : \Sigma \to 2^{States^2} \right)$$

e	$\mathcal{R}_i[\![e]\!]$
$t \in T$	$\{(s,s) : s \in sat(t)\}$
$\mathbf{a} + \mathbf{b}$	$\mathcal{R}_i[\![\mathbf{a}]\!] \cup \mathcal{R}_i[\![\mathbf{b}]\!]$
$\mathbf{ab}$	$\mathcal{R}_i[\![\mathbf{a}]\!] \cap \mathcal{R}_i[\![\mathbf{b}]\!]$
$\overline{\mathbf{a}}$	$\{(s,s) : s \in States\} \setminus \mathcal{R}_i[\![\mathbf{a}]\!]$
$p \in \Sigma$	$eval(p)$
$\mathbf{e} +_{\mathbf{a}} \mathbf{f}$	$\mathcal{R}_i[\![\mathbf{a}]\!] \circ \mathcal{R}_i[\![\mathbf{e}]\!] \cup \mathcal{R}_i[\![\overline{\mathbf{a}}]\!] \circ \mathcal{R}_i[\![\mathbf{f}]\!]$
$\mathbf{ef}$	$\mathcal{R}_i[\![\mathbf{e}]\!] \circ \mathcal{R}_i[\![\mathbf{f}]\!]$
$\mathbf{e}^{(\mathbf{a})}$	$(\mathcal{R}_i[\![\mathbf{a}]\!] \circ \mathcal{R}_i[\![\mathbf{e}]\!])^* \circ \mathcal{R}_i[\![\overline{\mathbf{a}}]\!]$

Relational Semantics

$$\mathit{Atoms} = 2^T$$

Relational Semantics

$$Atoms = 2^T$$

$$\alpha_0 p_0 \alpha_1 p_1 \cdots \alpha_{n-1} p_{n-1} \alpha_n \qquad \alpha_i \in Atoms \qquad p_i \in \Sigma$$

Relational Semantics

$$Atoms = 2^T$$

$$\alpha_0 p_0 \alpha_1 p_1 \cdots \alpha_{n-1} p_{n-1} \alpha_n \qquad \alpha_i \in Atoms \qquad p_i \in \Sigma$$

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

Language semantics

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

Decision Procedure

Theorem

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

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]

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]

Axiomatization

Axiomatization: `if-then-else`

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

Axiomatization: `if-then-else`

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

Example

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

Axiomatization: `if-then-else`

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

Example

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

Axiomatization: `if-then-else`

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

Example

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

Axiomatization: `if-then-else`

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

Example

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

Axiomatization: `if-then-else`

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

Example

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

Axiomatization: `if-then-else`

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

Example

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

Axiomatization: `if-then-else`

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

Example

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

Axiomatization: `if-then-else`

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

Example

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

Axiomatization: `while`

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

Axiomatization: `while`

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

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

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

Axiomatization: `while`

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

Axiomatization: `while`

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

Salomaa 1966

Axiomatization: `while`

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

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

Axiomatization: `while`

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

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

Axiomatization: `while`

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

Lemma

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

Axiomatization: `while`

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

Lemma

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

Lemma

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

Axioms versus semantics

Theorem (Soundness)

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

Axioms versus semantics

Theorem (Soundness)

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

How about the converse?

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

Axioms versus semantics

Theorem (Soundness)

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

How about the converse?

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

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

Axioms versus semantics

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

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

Axioms versus semantics

Theorem (Soundness)

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

Theorem (Completeness)

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

Axiomatization

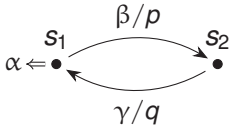
How about the converse?

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

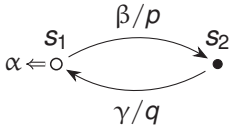
$$
\begin{aligned}
[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!] &\implies L(A_{\mathbf{e}}) = L(A_{\mathbf{f}}) \\
&\implies A_{\mathbf{e}} \sim A_{\mathbf{f}} \\
&\implies S(A_{\mathbf{e}}) \equiv S(A_{\mathbf{f}}) \\
&\implies \mathbf{e} \equiv \mathbf{f}
\end{aligned}
$$

Kleene Theorem

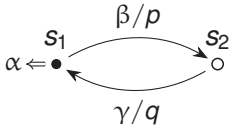
Automata model



Automata model

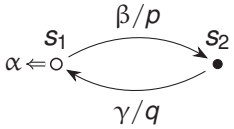


Automata model



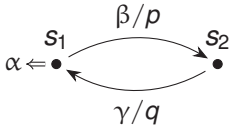
βp

Automata model



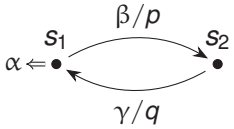
$\beta p \gamma q$

Automata model



$$\beta p \gamma q \alpha \in \mathcal{L}(s_1)$$

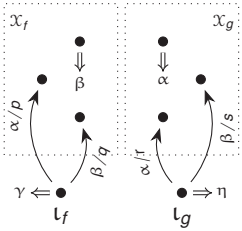
Automata model



$$\beta p \gamma q \alpha \in \mathcal{L}(s_1)$$

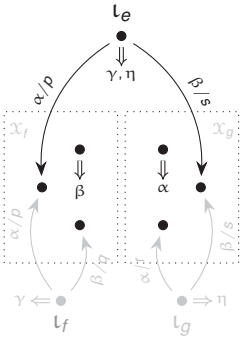
$$(X, \delta : X \to (2 + \Sigma \times X)^{Atoms})$$

Kleene Theorem: expressions to automata



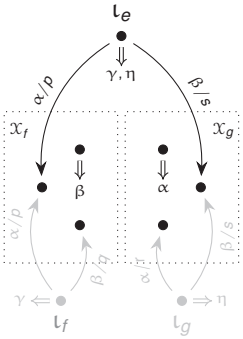
$$\mathbf{e} = \mathbf{f} +_{\mathbf{a}} \mathbf{g}$$

Kleene Theorem: expressions to automata

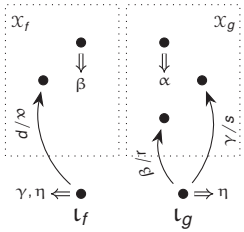


$$\mathbf{e} = \mathbf{f} +_{\mathbf{a}} \mathbf{g}$$

Kleene Theorem: expressions to automata

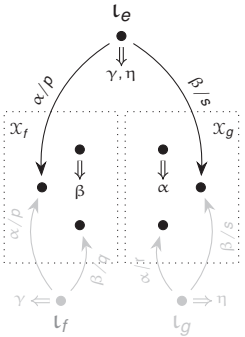


$\mathbf{e = f +_a g}$

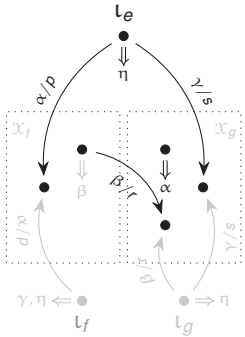


$\mathbf{e = fg}$

Kleene Theorem: expressions to automata

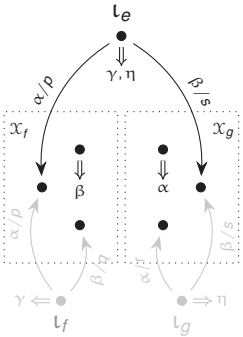


$\mathbf{e = f +_a g}$

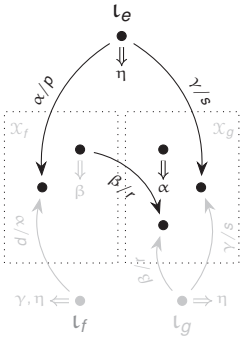


$\mathbf{e = fg}$

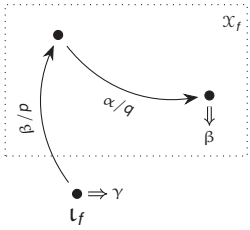
Kleene Theorem: expressions to automata



$\mathbf{e = f +_a g}$

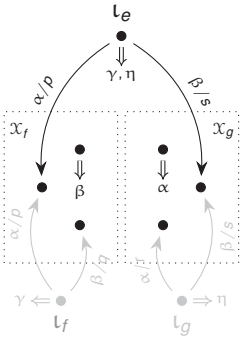


$\mathbf{e = fg}$

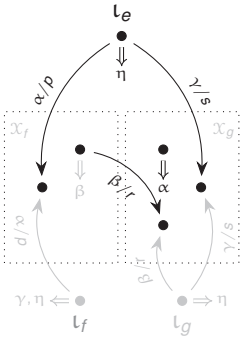


$\mathbf{e = f^{(a)}}$

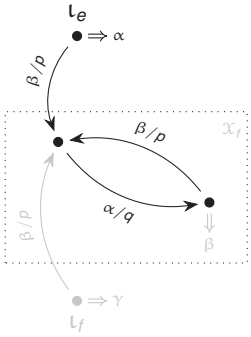
Kleene Theorem: expressions to automata



$\mathbf{e = f +_a g}$



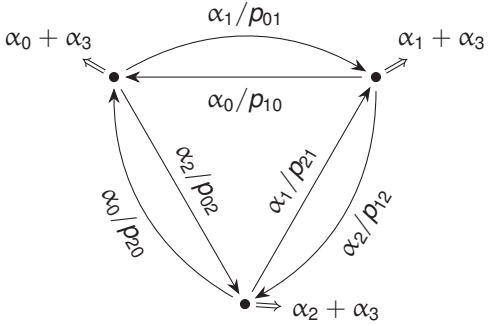
$\mathbf{e = fg}$



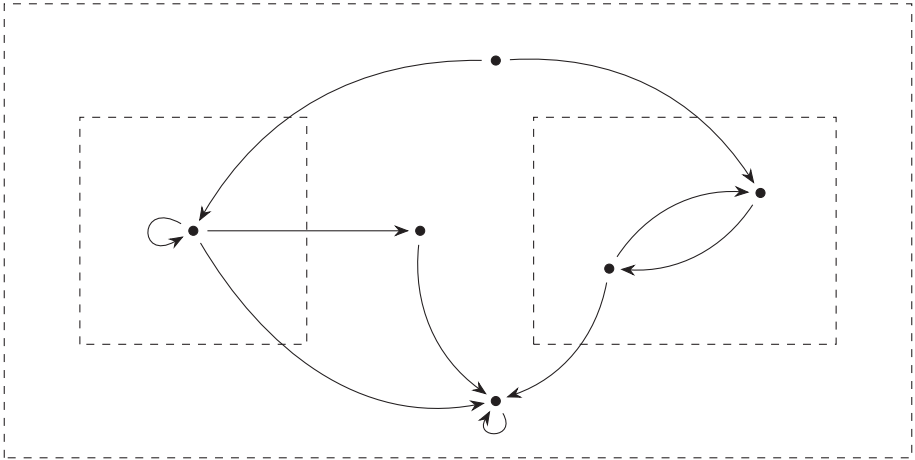
$\mathbf{e = f^{(a)}}$

Kleene Theorem: automata to expressions

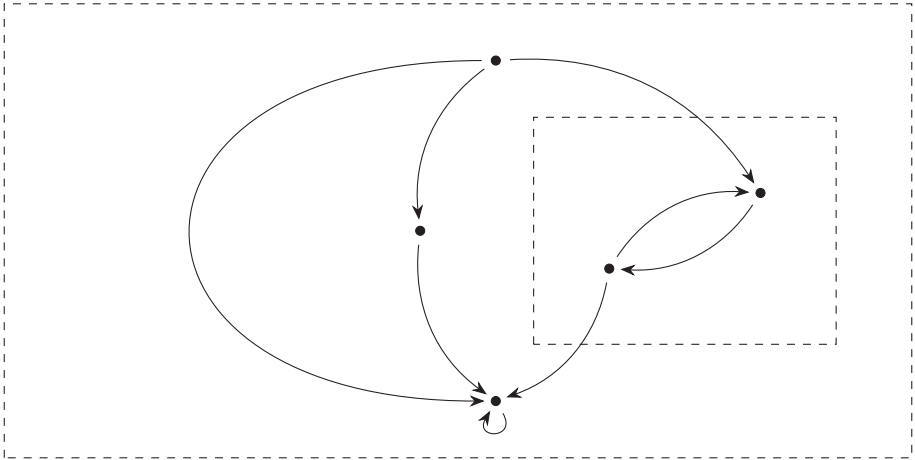
From [Kozen and Tseng 2008]:



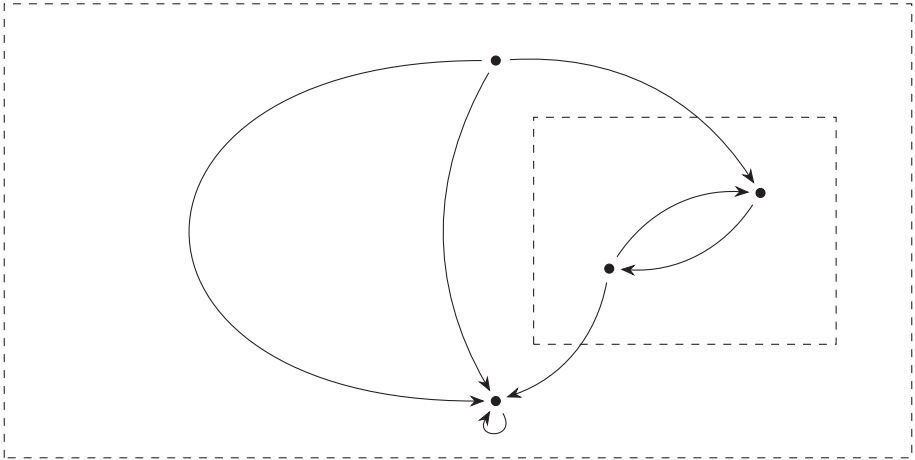
Kleene Theorem: automata to expressions



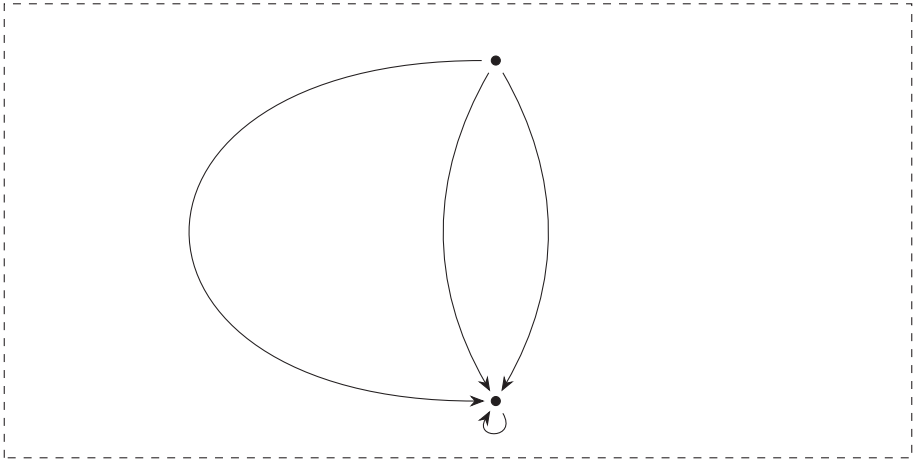
Kleene Theorem: automata to expressions



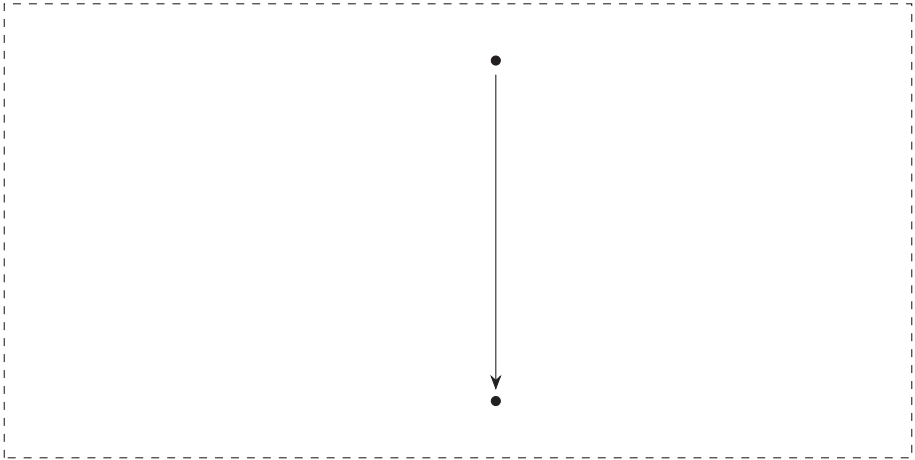
Kleene Theorem: automata to expressions



Kleene Theorem: automata to expressions



Kleene Theorem: automata to expressions



Kleene Theorem

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.

Kleene Theorem

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

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.

Further work

- Coalgebraic perspective, coequations
- Instantiation framework; hypotheses
- Fully algebraic axiomatization

Laurie J. Hendren (1958–2019)

https://kap.pe/slides

https://arxiv.org/abs/1907.05920