Reasoning about Program Equivalence using (Prob)GKAT

Tobias Kappé

Open University of The Netherlands
Institute for Logic, Language and Computation, University of Amsterdam

OUrsi Seminar — October 22, 2022

Joint work with . . .

Alexandra Silva

Dexter Kozen

Justin Hsu

Nate Foster

Steffen Smolka

Todd Schmid

Wojciech Rozowski

Motivation: comparing programs

```
if not a then
    e;
else
    f;
```

$\equiv$

```
if a then
    f;
else
    e;
```

Motivation: comparing programs

```
if a then
   e;
   while a do
      e;
```

$\equiv$

```
while a do
   e;
```

A more complicated equivalence

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

$\equiv$

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

Open Universiteit

Initial questions

- What is the minimal set of axioms?
- Are those axioms complete w.r.t. some model?
- Can we decide axiomatic equivalence?

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

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

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

$\mathbf{a}$ or $\mathbf{b}$

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

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

$\mathbf{a}$ `and` $\mathbf{b}$

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

Open Universiteit

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

`not` $\mathbf{a}$

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

Open Universiteit

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

`false`

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

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

`true`

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

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

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

`assert` $\mathbf{a}$

Open Universiteit

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

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

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

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

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

Open Universiteit

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

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

`if` $\mathbf{a}$ `then` $\mathbf{e}$ `else` $\mathbf{f}$

Condensing the syntax

Treat `while`-programs as expressions — c.f. (Kozen and Tseng 2008).

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

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

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

Open Universiteit

Some example axioms

$$\mathbf{e} +_{\mathbf{a}} \mathbf{e} \equiv \mathbf{e}$$

Some example axioms

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

Some example axioms

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

Open Universiteit

Some example axioms

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

Open Universiteit

Some example axioms

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

Some example axioms

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

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

Some example axioms

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

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

Open Universiteit

Some example axioms

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

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

Some example axioms

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

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

Some example axioms

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

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

Some example axioms

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

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

Some example axioms

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

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

Open Universiteit

Guarded Kleene Algebra with Tests

$$e +_a e \equiv e \qquad e +_a f \equiv f +_{\overline{a}} e \qquad (e +_a f) +_b g \equiv e +_{ab} (f +_b g)$$

$$e +_a f \equiv ae +_a f \qquad eg +_a fg \equiv (e +_a f)g \qquad (ef)g \equiv e(fg) \qquad 0e \equiv 0$$

$$e0 \equiv 0 \qquad 1e \equiv e \qquad e1 \equiv e \qquad e^{(a)} \equiv ee^{(a)} +_a 1 \qquad (e +_a 1)^{(b)} \equiv (ae)^{(b)}$$

Guarded Kleene Algebra with Tests

Fixpoints: If $\mathbf{fe} +_{\mathbf{b}} \mathbf{g} \equiv \mathbf{e}$ and $\mathbf{e}$ is productive, then $\mathbf{f}^{(\mathbf{b})}\mathbf{g} \equiv \mathbf{e}$.

Guarded Kleene Algebra with Tests

Fixpoints: If $\mathbf{fe} +_{\mathbf{b}} \mathbf{g} \equiv \mathbf{e}$ and $\mathbf{e}$ is productive, then $\mathbf{f}^{(\mathbf{b})}\mathbf{g} \equiv \mathbf{e}$.

Unique solutions: affine systems of equations, i.e., of the form

$$\begin{array}{cccccccccc}
\mathbf{e}_{1,1} \cdot x_1 & +_{\mathbf{a}_{1,1}} & \mathbf{e}_{1,2} \cdot x_2 & +_{\mathbf{a}_{1,2}} & \cdots & +_{\mathbf{a}_{1,n}} & \mathbf{b}_1 & \equiv x_1 \\
\vdots & & & & \ddots & & & \vdots \\
\mathbf{e}_{n,1} \cdot x_1 & +_{\mathbf{a}_{n,1}} & \mathbf{e}_{n,2} \cdot x_2 & +_{\mathbf{a}_{n,2}} & \cdots & +_{\mathbf{a}_{n,n}} & \mathbf{b}_n & \equiv x_n
\end{array}$$

have at most one solution (up to $\equiv$) — provided the $\mathbf{e}_{i,j}$ are *productive*.

Guarded Kleene Algebra with Tests

Theorem (Smolka et al. (2020))

- $\equiv$ *is sound and complete w.r.t. a natural model.*
- $\equiv$ *is decidable in nearly-linear time (for a fixed number of tests).*

A more complicated equivalence

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

$$\mathsf{e}^{(\mathsf{ab})} \cdot (\mathsf{fe}^{(\mathsf{ab})})^{(\mathsf{a})}$$

$\equiv$

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

$$(\mathsf{e} +_{\mathsf{b}} \mathsf{f})^{(\mathsf{a})}$$

Followup questions

- What if we drop the axiom $\mathbf{e}0 \equiv 0$?
- How expressive is this syntax?
- Can we simplify the last axiom?

Followup questions

- What if we drop the axiom $\mathbf{e}0 \equiv 0$?
- How expressive is this syntax?
- Can we simplify the last axiom?

Third question remains open!

The axiom $\mathbf{e}0 \equiv 0$

Intuition: "failing now is the same as failing later" . . .

The axiom $\mathbf{e}0 \equiv 0$

Intuition: "failing now is the same as failing later" ...

... but what if the actions before failure matter?

Open Universiteit

But wait, there's more

Provable in GKAT: $\mathbf{e}^{(\mathbf{a})} \equiv \mathbf{e}^{(\mathbf{a})}\overline{\mathbf{a}}$.

But wait, there's more

Provable in GKAT: $\mathbf{e}^{(\mathbf{a})} \equiv \mathbf{e}^{(\mathbf{a})}\overline{\mathbf{a}}$.

In particular,

```
while true do e end
```

But wait, there's more

Provable in GKAT: $\mathbf{e}^{(\mathsf{a})} \equiv \mathbf{e}^{(\mathsf{a})}\overline{\mathsf{a}}$.

In particular,

$$\texttt{while true do } \mathbf{e} \texttt{ end} = \mathbf{e}^{(1)}$$

Open Universiteit

But wait, there's more

Provable in GKAT: $\mathbf{e}^{(\mathsf{a})} \equiv \mathbf{e}^{(\mathsf{a})}\overline{\mathsf{a}}$.

In particular,

$$\begin{aligned}\texttt{while true do } \mathbf{e} \texttt{ end} &= \mathbf{e}^{(1)} \\ &\equiv \mathbf{e}^{(1)} \cdot \overline{1}\end{aligned}$$

But wait, there's more

Provable in GKAT: $\mathbf{e}^{(\mathsf{a})} \equiv \mathbf{e}^{(\mathsf{a})}\overline{\mathsf{a}}$.

In particular,

$$
\begin{aligned}
\texttt{while true do } \mathbf{e} \texttt{ end} &= \mathbf{e}^{(1)} \\
&\equiv \mathbf{e}^{(1)} \cdot \overline{1} \\
&\equiv \mathbf{e}^{(1)} \cdot 0
\end{aligned}
$$

But wait, there's more

Provable in GKAT: $\mathbf{e}^{(\mathbf{a})} \equiv \mathbf{e}^{(\mathbf{a})}\overline{\mathbf{a}}$.

In particular,

$$
\begin{aligned}
\texttt{while true do } \mathbf{e} \texttt{ end} &= \mathbf{e}^{(1)} \\
&\equiv \mathbf{e}^{(1)} \cdot \overline{1} \\
&\equiv \mathbf{e}^{(1)} \cdot 0 \\
&\equiv 0 \qquad = \texttt{assert false}
\end{aligned}
$$

But wait, there's more

Provable in GKAT: $\mathbf{e}^{(\mathsf{a})} \equiv \mathbf{e}^{(\mathsf{a})}\overline{\mathsf{a}}$.

In particular,

$$\begin{aligned}
\texttt{while true do } \mathbf{e} \texttt{ end} &= \mathbf{e}^{(1)} \\
&\equiv \mathbf{e}^{(1)} \cdot \overline{1} \\
&\equiv \mathbf{e}^{(1)} \cdot 0 \\
&\equiv 0 \qquad = \texttt{assert false}
\end{aligned}$$

🤔

Open Universiteit

But wait, there's more

Provable in GKAT: $\mathbf{e}^{(\mathsf{a})} \equiv \mathbf{e}^{(\mathsf{a})}\overline{\mathsf{a}}$.

In particular,

$$
\begin{aligned}
\texttt{while true do } \mathbf{e} \texttt{ end} &= \mathbf{e}^{(1)} \\
&\equiv \mathbf{e}^{(1)} \cdot \overline{1} \\
&\equiv \mathbf{e}^{(1)} \cdot 0 \\
&\equiv 0 \qquad\qquad = \texttt{assert false}
\end{aligned}
$$

🤔

See also (Mamouras 2017).

Mission statement

Question

Let $\equiv_0$ be like $\equiv$, but without relating $\mathbf{e}0$ *to* 0.

Can we recover the same results for this finer equivalence?

Open Universiteit

Mission statement

Question

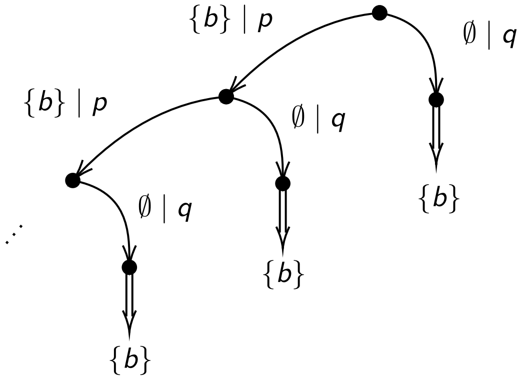
Let $\equiv_0$ be like $\equiv$, but without relating $\mathbf{e}0$ to 0.

Can we recover the same results for this finer equivalence?

Roadmap:

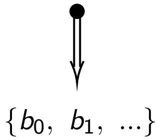
1. Find a model satisfying the axioms.
2. Prove soundness and completeness.
3. Decide equivalence within that model.

Guarded trees — example

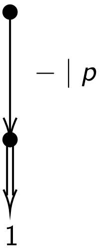


Expressions to trees — base case

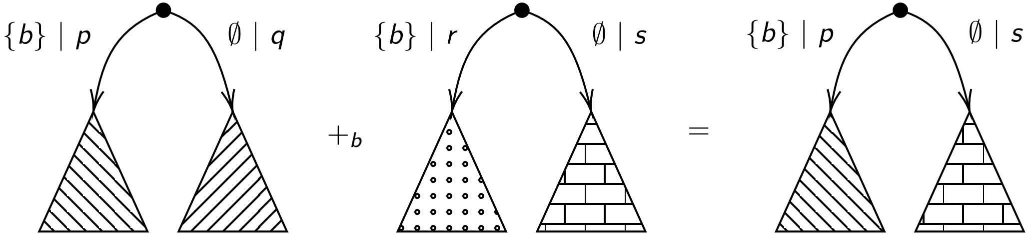
$a = \{b_0,\ b_1,\ \dots\} \mapsto$



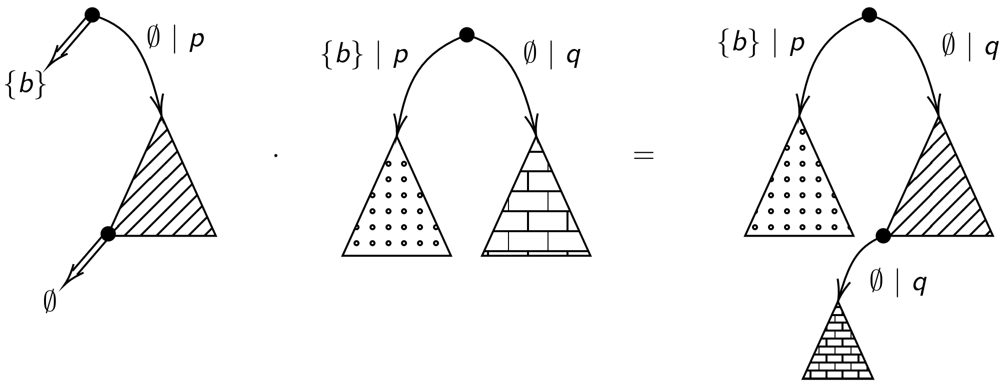
$p \in \Sigma \mapsto$



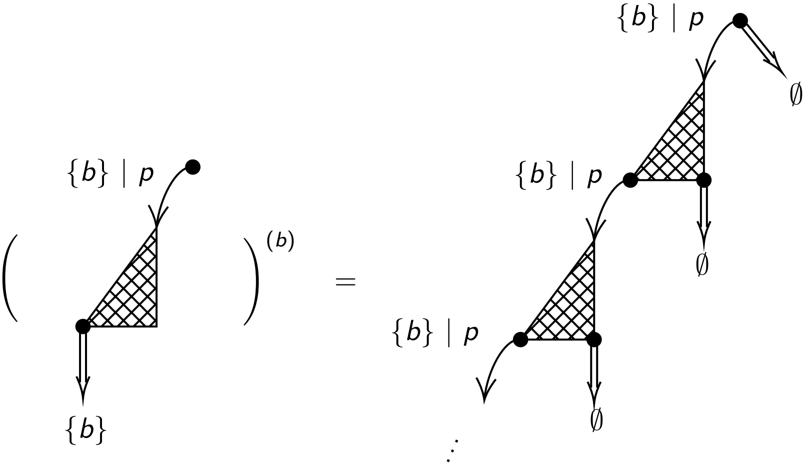
Expressions to trees — Party hat diagrams



Expressions to trees — Party hat diagrams



Expressions to trees — Party hat diagrams



A model in terms of guarded trees

Every expression $\mathbf{e}$ has an associated guarded tree $[\![\mathbf{e}]\!]$.

Open Universiteit

A model in terms of guarded trees

Every expression $\mathbf{e}$ has an associated guarded tree $[\![\mathbf{e}]\!]$.

The early termination axiom does *not* hold: $[\![\mathbf{e}0]\!] \neq [\![0]\!]$.

A model in terms of guarded trees

Every expression $\mathbf{e}$ has an associated guarded tree $[\![\mathbf{e}]\!]$.

The early termination axiom does *not* hold: $[\![\mathbf{e}0]\!] \neq [\![0]\!]$.

Question (Soundness & Completeness)

Is $\mathbf{e} \equiv_0 \mathbf{f}$ *equivalent to* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$*?*

A model in terms of guarded trees

Every expression $\mathbf{e}$ has an associated guarded tree $[\![\mathbf{e}]\!]$.

The early termination axiom does *not* hold: $[\![\mathbf{e}0]\!] \neq [\![0]\!]$.

Question (Soundness & Completeness)

Is $\mathbf{e} \equiv_0 \mathbf{f}$ *equivalent to* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$*?*

Question (Decidability)

Can we decide whether $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$*?*

Establishing completeness and decidability

From (Schmid et al. 2021):

Establishing completeness and decidability

From (Schmid et al. 2021):

Theorem (Soundness & Completeness)

$\mathbf{e} \equiv_0 \mathbf{f}$ *if and only if* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$

Establishing completeness and decidability

From (Schmid et al. 2021):

Theorem (Soundness & Completeness)

$\mathbf{e} \equiv_0 \mathbf{f}$ *if and only if* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$

Theorem (Decidability for trees)

It is decidable whether $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$ *(proof is coalgebraic!)*

Open Universiteit

Establishing completeness and decidability

From (Schmid et al. 2021):

Theorem (Soundness & Completeness)

$\mathbf{e} \equiv_0 \mathbf{f}$ *if and only if* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$

Theorem (Decidability for trees)

It is decidable whether $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$ *(proof is coalgebraic!)*

Corollary (Decidability for terms)

It is decidable whether $\mathbf{e} \equiv_0 \mathbf{f}$

Establishing completeness and decidability

From (Schmid et al. 2021):

Theorem (Soundness & Completeness)

$\mathbf{e} \equiv_0 \mathbf{f}$ *if and only if* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$

Theorem (Decidability for trees)

It is decidable whether $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$ *(proof is coalgebraic!)*

Corollary (Decidability for terms)

It is decidable whether $\mathbf{e} \equiv_0 \mathbf{f}$

Note: decision procedures are *nearly-linear* — actually feasible!

Open Universiteit

Establishing completeness and decidability

From (Schmid et al. 2021):

Theorem (Soundness & Completeness)

$\mathbf{e} \equiv_0 \mathbf{f}$ *if and only if* $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$

Theorem (Decidability for trees)

It is decidable whether $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$ *(proof is coalgebraic!)*

Corollary (Decidability for terms)

It is decidable whether $\mathbf{e} \equiv_0 \mathbf{f}$

Note: decision procedures are *nearly-linear* — actually feasible!

The "old" results from (Smolka et al. 2020) can be recovered from these.

Expressiveness

Question

Let $\mathtt{t}$ *be a guarded tree with finitely many distinct subtrees.*

Is there an $\mathbf{e}$ *such that* $[\![\mathbf{e}]\!] = \mathtt{t}$*?*

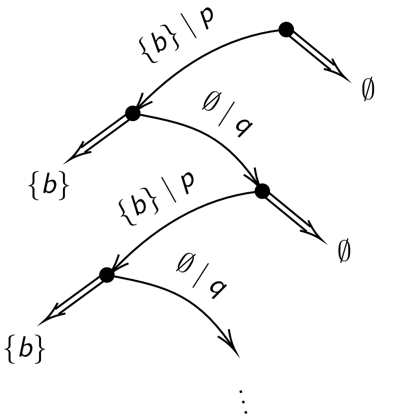
Expressiveness

Question

Let $\mathtt{t}$ *be a guarded tree with finitely many distinct subtrees.*

Is there an $\mathbf{e}$ *such that* $[\![\mathbf{e}]\!] = \mathtt{t}$*?*

Not in general — for instance:



See also (Kozen and Tseng 2008).

Expressiveness

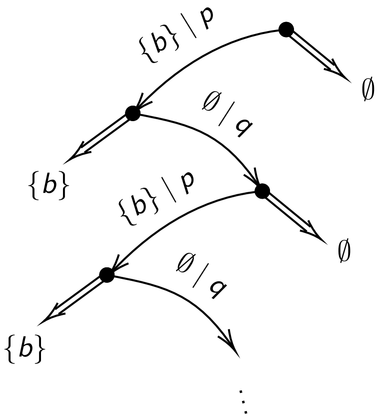
Question

Let $\mathtt{t}$ *be a guarded tree with finitely many distinct subtrees.*

Is there an $\mathbf{e}$ *such that* $[\![\mathbf{e}]\!] = \mathtt{t}$*?*

Reason: our syntax does not have `goto`.
Only *structured* programs!

Not in general — for instance:



See also (Kozen and Tseng 2008).

Expressiveness

Question

Let t *be a guarded tree with finitely many distinct subtrees.*

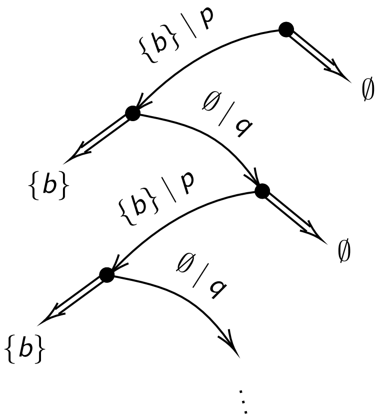
Is there an **e** *such that* $[\![\mathbf{e}]\!] = \mathtt{t}$*?*

Reason: our syntax does not have goto.
Only *structured* programs!

ℓ_0 :if $\mathbf{b}$ then p; goto ℓ_1 else accept

ℓ_1 :if $\overline{\mathbf{b}}$ then q; goto ℓ_0 else accept

Not in general — for instance:



See also (Kozen and Tseng 2008).

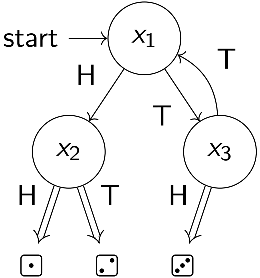
Knuth-Yao algorithm

How to simulate  using   ?

Knuth-Yao algorithm

How to simulate  using   ?

```
while true do
    if flip(0.5) then
        if flip(0.5) then
            return 1 // heads-heads
        else
            return 2 // heads-tails
    else
        if flip(0.5) then
            return 3 // tails-heads
        else
            skip // tails-tails
```

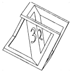


Correctness of Knuth-Yao in ProbGKAT

```
while true do
    if flip(0.5) then
        if flip(0.5) then
            return 1 // heads-heads
        else
            return 2 // heads-tails
    else
        if flip(0.5) then
            return 3 // tails-heads
        else
            skip // tails-tails
```

$$((r_1 \oplus_{\frac{1}{2}} r_2) \oplus_{\frac{1}{2}} (r_3 \oplus_{\frac{1}{2}} 1))^{(1)}$$

$$\stackrel{?}{\equiv}$$



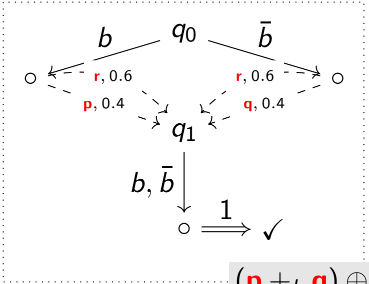
```
if flip(1/3) then
    return 1
else
    if flip(0.5) then
        return 2
    else
        return 3
```

$$r_1 \oplus_{\frac{1}{3}} (r_2 \oplus_{\frac{1}{2}} r_3)$$

Operational model

Automata with the transition function of the type

$$Q \times \mathtt{At} \to \mathcal{D}_\omega(\{\checkmark, \text{✗}\} + V + \mathtt{Act} \times Q)$$

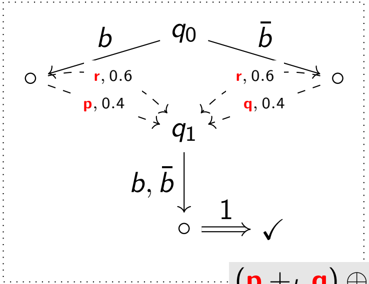


$(\mathbf{p} +_b \mathbf{q}) \oplus_{0.4} \mathbf{r}$

Operational model

Automata with the transition function of the type

$$Q \times \mathtt{At} \to \mathcal{D}_\omega(\{\checkmark, \text{✗}\} + V + \mathtt{Act} \times Q)$$



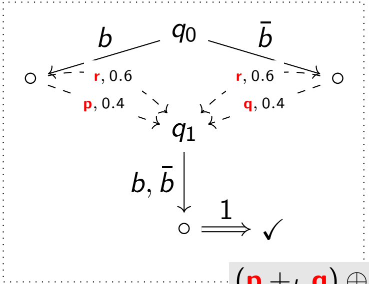
$(\mathbf{p} +_b \mathbf{q}) \oplus_{0.4} \mathbf{r}$

- Notion of equivalence: bisimulation associated with the type functor

Operational model

Automata with the transition function of the type

$$Q \times \mathtt{At} \to \mathcal{D}_\omega(\{\checkmark, \text{✗}\} + V + \mathtt{Act} \times Q)$$



$$(\mathbf{p} +_b \mathbf{q}) \oplus_{0.4} \mathbf{r}$$

- Notion of equivalence: bisimulation associated with the type functor
- Can be decided in $O(n^2 \log(n))$ using a generic minimization algorithm (Wißmann et al, 2020)

Open Universiteit

Overview

- GKAT describes general equivalences of programs.
- It admits a complete axiomatization and is decidable.
- There is a model for the theory without $\mathbf{e}0 \equiv 0$.
- Soundness and completeness can be recovered.
- Lack of GOTO means not every tree is expressible.
- A probabilistic extension is in the works.

`https://kap.pe/slides` `https://kap.pe/papers`

Open Universiteit

Bonus — What is “nearly-linear”?

Nearly-linear complexity is $O(\alpha(n) \cdot n)$, where α is the *inverse Ackermann function*.

Fun fact: $\alpha(n) \leq 5$ for most numbers you can think of:

- Grains of sand in the Sahara.
- The number of DNA base pairs on earth.
- Number of protons in the observable universe.

See also (Tarjan 1975).

Bonus — Reduction to KAT

Syntax is special case of Kleene Algebra with Tests (KAT):

$$\texttt{if } \mathbf{a} \texttt{ then } \mathbf{e} \texttt{ else } \mathbf{f} \texttt{ end} \mapsto \mathbf{a} \cdot \mathbf{e} + \overline{\mathbf{a}} \cdot \mathbf{f}$$

$$\texttt{while } \mathbf{a} \texttt{ do } \mathbf{e} \texttt{ end} \mapsto (\mathbf{a} \cdot \mathbf{e})^* \cdot \overline{\mathbf{a}}$$

Bonus — Reduction to KAT

Syntax is special case of Kleene Algebra with Tests (KAT):

$$\texttt{if } \mathbf{a} \texttt{ then } \mathbf{e} \texttt{ else } \mathbf{f} \texttt{ end} \mapsto \mathbf{a} \cdot \mathbf{e} + \overline{\mathbf{a}} \cdot \mathbf{f}$$

$$\texttt{while } \mathbf{a} \texttt{ do } \mathbf{e} \texttt{ end} \mapsto (\mathbf{a} \cdot \mathbf{e})^* \cdot \overline{\mathbf{a}}$$

Known results:

- There is a "nice" set of axioms for KAT.
- Soundness & completeness for a straightforward model.
- Equivalence according to these axioms is decidable.

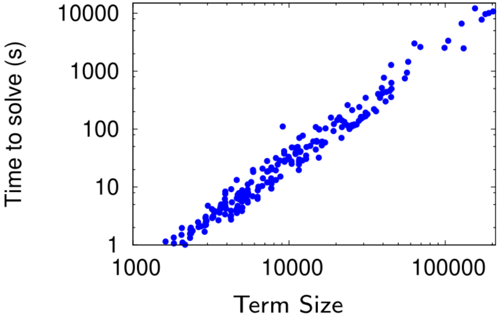
Bonus — Reduction to KAT

Equivalence in KAT is PSPACE-complete (Cohen, Kozen, and Smith 1996).

Bonus — Reduction to KAT

Equivalence in KAT is PSPACE-complete (Cohen, Kozen, and Smith 1996).

But for practical inputs, good algorithms scale well — e.g., (Foster et al. 2015):



Open Universiteit

References

Ernie Cohen, Dexter Kozen, and Frederick Smith (July 1996). *The Complexity of Kleene Algebra with Tests*. Tech. rep. TR96-1598. Cornell University. handle: 1813/7253.

Nate Foster et al. (2015). "A Coalgebraic Decision Procedure for NetKAT". In: *POPL*, pp. 343–355. DOI: 10.1145/2676726.2677011.

Dexter Kozen and Wei-Lung (Dustin) Tseng (2008). "The Böhm-Jacopini Theorem is False, Propositionally". In: *MPC*, pp. 177–192. DOI: 10.1007/978-3-540-70594-9_11.

Konstantinos Mamouras (2017). "Equational Theories of Abnormal Termination Based on Kleene Algebra". In: *FOSSACS*. Vol. 10203. Lecture Notes in Computer Science, pp. 88–105. DOI: 10.1007/978-3-662-54458-7_6.

Todd Schmid et al. (2021). "Guarded Kleene Algebra with Tests: Coequations, Coinduction, and Completeness". In: *ICALP*, 142:1–142:14. DOI: 10.4230/LIPIcs.ICALP.2021.142.

Steffen Smolka et al. (Jan. 2020). "Guarded Kleene Algebra with Tests: Verification of Uninterpreted Programs in Nearly Linear Time". In: *POPL*. DOI: 10.1145/3371129.

Robert Endre Tarjan (1975). "Efficiency of a Good But Not Linear Set Union Algorithm". In: 22.2, pp. 215–225. DOI: 10.1145/321879.321884.