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 ...
Motivation: comparing programs

\[
\begin{align*}
\text{if not } a \text{ then} & \quad \text{if } a \text{ then} \\
\quad e; & \quad f; \\
\text{else} & \quad \text{else} \\
\quad f; & \quad e;
\end{align*}
\]
Motivation: comparing programs

\[
\text{if } a \text{ then }
\begin{align*}
  & e; \\
  & \text{while } a \text{ do }
  \begin{align*}
    & e; \\
  \end{align*}
\end{align*}
\]

\[
\equiv
\begin{align*}
  & \text{while } a \text{ do }
  \begin{align*}
    & e;
  \end{align*}
\end{align*}
\]
A more complicated equivalence

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

while a do
  if b then
    e;
  else
    f;
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).

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

\[ e, f ::= a \mid p \in \Sigma \mid ef \mid e + a \mid f \mid e^{(a)} \]
Condensing the syntax

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

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

\[ a \text{ or } b \]

\[ e, f ::= a \mid p \in \Sigma \mid ef \mid e +_a f \mid e^{(a)} \]
Condensing the syntax

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

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

\[
e, f ::= a \mid p \in \Sigma \mid ef \mid e +_a f \mid e^{(a)}
\]
Condensing the syntax

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

\[
\begin{align*}
\text{a, b} & ::= \ t \in T \mid \text{a + b} \mid \text{ab} \mid \overline{\text{a}} \mid 0 \mid 1 \\
\text{not a} \\
\text{e, f} & ::= \text{a} \mid p \in \Sigma \mid \text{ef} \mid \text{e + a f} \mid e^{(a)}
\end{align*}
\]
Condensing the syntax

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

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

\[ e, f ::= a \mid p \in \Sigma \mid ef \mid e +_a f \mid e^{(a)} \]
Condensing the syntax

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

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

\[ e, f ::= a \mid p \in \Sigma \mid ef \mid e + a f \mid e^{(a)} \]
Condensing the syntax

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

\begin{align*}
a, b & : = t \in \mathcal{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)}
\end{align*}
Condensing the syntax

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

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

\[ e, f ::= a \mid p \in \Sigma \mid ef \mid e + a f \mid e^{(a)} \]
Condensing the syntax

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

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

\[ e, f ::= a \mid p \in \Sigma \mid ef \mid e +_a f \mid e^{(a)} \]
Condensing the syntax

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

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

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

if \( a \) then \( e \) else \( f \)
Condensing the syntax

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

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

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

while \( a \) do \( e \)
Some example axioms

\[ e +_a e \equiv e \]
Some example axioms

\[ e +_a e \equiv e \quad e +_a f \equiv f +_a e \]
Some example axioms

\[ e +_a e \equiv e \quad e +_a f \equiv f +_\bar{a} e \quad e +_a f \equiv ae +_a f \]
Some example axioms

\[
\begin{align*}
e + a \, e & \equiv e \\
e + a \, f & \equiv f + a \, e \\
e + a \, f & \equiv a \, e + a \, f \\
\overline{a}a & \equiv 0
\end{align*}
\]
Some example axioms

\[ e + a \ e \equiv e \quad e + a \ f \equiv f + a \ e \quad e + a \ f \equiv a \ e + a \ f \quad \bar{a}a \equiv 0 \quad 0e \equiv 0 \]
Some example axioms

\[ e + a e \equiv e \quad e + a f \equiv f + a e \quad e + a f \equiv a e + a f \quad \overline{a} a \equiv 0 \quad 0 e \equiv 0 \]

\[ \text{if } a \text{ then } e \text{ else assert false } = e + a 0 \]
Some example axioms

\[ e +_a e \equiv e \quad e +_a f \equiv f +_a e \quad e +_a f \equiv ae +_a f \quad \overline{aa} \equiv 0 \quad 0e \equiv 0 \]

if \( a \) then \( e \) else assert false = \( e +_a 0 \equiv ae +_a 0 \)
Some example axioms

\[ e + a e \equiv e \quad \boxed{e + a f \equiv f + \overline{a} e} \quad e + a f \equiv a e + a f \quad \overline{a} a \equiv 0 \quad 0 e \equiv 0 \]

if \( a \) then \( e \) else assert false = \( e + a 0 \equiv a e + a 0 \)

\[ \equiv 0 + \overline{a} a e \]
Some example axioms

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

if \ a \ then \ e \ else \ assert \ false = e +_a 0 \equiv ae +_a 0
\equiv 0 +_{\bar{a}} ae
\equiv 0e +_{\bar{a}} ae
Some example axioms

\[
e_a e \equiv e \quad e_a f \equiv f + \overline{a} e \quad e_a f \equiv ae + af \quad \overline{aa} \equiv 0 \quad 0e \equiv 0
\]

if a then e else assert false = e + a 0 \equiv ae + a 0

\[
\equiv 0 + \overline{a} ae \\
\equiv 0e + \overline{a} ae \\
\equiv \overline{aa}e + \overline{a} ae
\]
Some example axioms

\[
\begin{align*}
e + a \cdot e & \equiv e \\
e + a \cdot f & \equiv f + a \cdot e \\
e + a \cdot f & \equiv a \cdot e + a \cdot f \\
\overline{a} \cdot a & \equiv 0 \\
0 \cdot e & \equiv 0
\end{align*}
\]

\[
\begin{align*}
& \text{if } a \text{ then } e \text{ else assert false} = e + a \cdot 0 \\
& \equiv a \cdot e + a \cdot 0 \\
& \equiv 0 + a \cdot ae \\
& \equiv 0 \cdot e + a \cdot ae \\
& \equiv \overline{a} \cdot ae + a \cdot ae \\
& \equiv a \cdot e + a \cdot ae
\end{align*}
\]
Some example axioms

\[
\begin{align*}
e +_a e & \equiv e \\
e +_a f & \equiv f +_a e \\
e +_a f & \equiv ae +_a f \\
aa & \equiv 0 \\
0e & \equiv 0
\end{align*}
\]

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

\[
\begin{align*}
& \equiv 0 +_a ae \\
& \equiv 0e +_a ae \\
& \equiv aae +_a ae \\
& \equiv ae +_a ae \\
& \equiv ae = \text{assert } a; e
\end{align*}
\]
Guarded Kleene Algebra with Tests

\[ e +_a e \equiv e \quad e +_a f \equiv f +_a e \quad (e +_a f) +_b g \equiv e +_{ab} (f +_b g) \]

\[ e +_a f \equiv ae +_a f \quad eg +_a fg \equiv (e +_a f)g \quad (ef)g \equiv e(fg) \quad 0e \equiv 0\]

\[ e0 \equiv 0 \quad 1e \equiv e \quad e1 \equiv e \quad e^{(a)} \equiv ee^{(a)} +_a 1 \quad (e +_a 1)^{(b)} \equiv (ae)^{(b)} \]
Fixpoints: If $fe +_b g \equiv e$ and $e$ is productive, then $f^{(b)}g \equiv e$. 
Guarded Kleene Algebra with Tests

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

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

\[
\begin{align*}
  e_{1,1} \cdot x_1 &+ a_{1,1} e_{1,2} \cdot x_2 + a_{1,2} \cdots + a_{1,n} b_1 &\equiv x_1 \\
  \vdots &\hspace{1cm} \vdots &\hspace{1cm} \vdots \\
  e_{n,1} \cdot x_1 &+ a_{n,1} e_{n,2} \cdot x_2 + a_{n,2} \cdots + a_{n,n} b_n &\equiv x_n
\end{align*}
\]

have at most one solution (up to \( \equiv \)) — provided the \( e_{i,j} \) 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;

\( e^{(ab)} \cdot (f e^{(ab)})^{(a)} \)

\[ \equiv \]

while a do
  if b then
    e;
  else
    f;

\((e +_{b} f)^{(a)}\)
Followup questions

- What if we drop the axiom $e_0 \equiv 0$?
- How expressive is this syntax?
- Can we simplify the last axiom?
Followup questions

▶ What if we drop the axiom $e0 \equiv 0$?
▶ How expressive is this syntax?
▶ Can we simplify the last axiom?

Third question remains open!
The axiom $e_0 \equiv 0$

Intuition: “failing now is the same as failing later” ...
The axiom $e_0 \equiv 0$

Intuition: “failing now is the same as failing later” . . .

. . . but what if the actions before failure matter?
But wait, there’s more

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

See also (Mamouras 2017).
But wait, there’s more

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

In particular,

```plaintext
while true do e end
```

See also (Mamouras 2017).
But wait, there's more

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

In particular,

while true do e end $= e^{(1)}$
But wait, there's more

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

In particular,

\[
\text{while true do e end} = e^{(1)} \\
\equiv e^{(1)} \cdot \overline{1}
\]
But wait, there’s more

Provable in GKAT: \( e^{(a)} \equiv e^{(a)\bar{a}} \).

In particular,

\[
\text{while true do } e \text{ end} = e^{(1)} \\
\equiv e^{(1)} \cdot \bar{I} \\
\equiv e^{(1)} \cdot 0
\]
But wait, there's more

Provable in GKAT: \( e^{(a)} \equiv e^{(a)\bar{a}}. \)

In particular,

\[
\text{while true do } e \text{ end} = e^{(1)} \\
\equiv e^{(1)} \cdot \bar{1} \\
\equiv e^{(1)} \cdot 0 \\
\equiv 0 \quad = \text{assert false}
\]
But wait, there’s more

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

In particular,

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

$$\equiv e^{(1)} \cdot \bar{1}$$

$$\equiv e^{(1)} \cdot 0$$

$$\equiv 0 \quad = \text{assert false}$$
But wait, there’s more

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

In particular,

\[
\text{while true do } e \text{ end} = e^{(1)} \\
\equiv e^{(1)} \cdot \bar{I} \\
\equiv e^{(1)} \cdot 0 \\
\equiv 0 = \text{assert false}
\]

See also (Mamouras 2017).
Mission statement

Question
Let $\equiv_0$ be like $\equiv$, but without relating $e_0$ to 0.

Can we recover the same results for this finer equivalence?
Mission statement

Question

Let $\equiv_0$ be like $\equiv$, but without relating $e0$ 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

\[ a = \{ b_0, b_1, \ldots \} \mapsto \{ b_0, b_1, \ldots \} \]

\[ p \in \Sigma \mapsto 1 \]
Expressions to trees — Party hat diagrams

\[ \{b\} \mid p \lor \emptyset \mid q \quad +_b \quad \{b\} \mid r \lor \emptyset \mid s = \{b\} \mid p \lor \emptyset \mid s \]
Expressions to trees — Party hat diagrams

\[
\{ b \} \mid p \quad \cdot \quad \{ b \} \mid p \quad \emptyset \mid q
\]

= 

\[
\{ b \} \mid p \quad \emptyset \mid q
\]
Expressions to trees — Party hat diagrams

\[(b)\]
A model in terms of guarded trees

Every expression $e$ has an associated guarded tree $[e]$. 

The early termination axiom does not hold: $J_e e_0 K \neq J_0 e K$.

Question (Soundness & Completeness)

Is $e \equiv 0 f$ equivalent to $J e K = J f K$?

Question (Decidability)

Can we decide whether $J e K = J f K$?
A model in terms of guarded trees

Every expression $e$ has an associated guarded tree $[e]$. The early termination axiom does not hold: $[e0] \neq [0]$. 
A model in terms of guarded trees

Every expression $e$ has an associated guarded tree $[e]$.
The early termination axiom does not hold: $[e0] \neq [0]$.

Question (Soundness & Completeness)
Is $e \equiv_0 f$ equivalent to $[e] = [f]$?
A model in terms of guarded trees

Every expression $e$ has an associated guarded tree $[e]$. The early termination axiom does not hold: $[e0] \neq [0]$.

Question (Soundness & Completeness)
Is $e \equiv_0 f$ equivalent to $[e] = [f]$?

Question (Decidability)
Can we decide whether $[e] = [f]$?
Establishing completeness and decidability

From (Schmid et al. 2021):

\[ e \equiv f \iff J_e K = J_f K \]

Theorem (Decidability for trees)

It is decidable whether \( J_e K = J_f K \) (proof is coalgebraic!)

Corollary (Decidability for terms)

It is decidable whether \( e \equiv f \)

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

The “old” results from (Smolka et al. 2020) can be recovered from these.
Establishing completeness and decidability

From (Schmid et al. 2021):

Theorem (Soundness & Completeness)

\[ e \equiv_0 f \text{ if and only if } \llbracket e \rrbracket = \llbracket f \rrbracket \]
Establishing completeness and decidability

From (Schmid et al. 2021):

Theorem (Soundness & Completeness)
\[ e \equiv_0 f \text{ if and only if } \llbracket e \rrbracket = \llbracket f \rrbracket \]

Theorem (Decidability for trees)
It is decidable whether \( \llbracket e \rrbracket = \llbracket f \rrbracket \) (proof is coalgebraic!)
Establishing completeness and decidability

From (Schmid et al. 2021):

**Theorem (Soundness & Completeness)**
\[ e \equiv_0 f \text{ if and only if } \llbracket e \rrbracket = \llbracket f \rrbracket \]

**Theorem (Decidability for trees)**
*It is decidable whether* \( \llbracket e \rrbracket = \llbracket f \rrbracket \) (*proof is coalgebraic!*)

**Corollary (Decidability for terms)**
*It is decidable whether* \( e \equiv_0 f \)
Establishing completeness and decidability

From (Schmid et al. 2021):

**Theorem (Soundness & Completeness)**

\[ e \equiv_0 f \text{ if and only if } \llbracket e \rrbracket = \llbracket f \rrbracket \]

**Theorem (Decidability for trees)**

*It is decidable whether \( \llbracket e \rrbracket = \llbracket f \rrbracket \) (proof is coalgebraic!)*

**Corollary (Decidability for terms)**

*It is decidable whether \( e \equiv_0 f \)*

Note: decision procedures are *nearly-linear* — actually feasible!
Establishing completeness and decidability

From (Schmid et al. 2021):

**Theorem (Soundness & Completeness)**

\( e \equiv_0 f \) if and only if \( \left[ e \right] = \left[ f \right] \)

**Theorem (Decidability for trees)**

*It is decidable whether \( \left[ e \right] = \left[ f \right] \) (proof is coalgebraic!)*

**Corollary (Decidability for terms)**

*It is decidable whether \( e \equiv_0 f \)*

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

The “old” results from (Smolka et al. 2020) can be recovered from these.
Expressiveness

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

*Is there an $e$ such that $[e] = t$?*
Expressiveness

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

Is there an $e$ such that $\llbracket e \rrbracket = t$?

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 $\llbracket e \rrbracket = t$?

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 $\llbracket e \rrbracket = t$?

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

\[
\ell_0 : \text{if } b \text{ then } p; \text{ goto } \ell_1 \text{ else accept}
\]

\[
\ell_1 : \text{if } \overline{b} \text{ then } q; \text{ goto } \ell_0 \text{ else accept}
\]

Not in general — for instance:

See also (Kozen and Tseng 2008).
Knuth-Yao algorithm

How to simulate using 🍀 🍀 ?

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

x 1 2 3
T H T H T H
```
Knuth-Yao algorithm

How to simulate using "heads" and "tails"?

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

\[
\text{while } \textbf{true} \text{ do }
\begin{align*}
\text{if flip}(0.5) & \text{ then} \\
& \quad \text{if flip}(0.5) \text{ then} \\
& \quad \quad \text{return 1 // heads-heads} \\
& \quad \quad \text{else} \\
& \quad \quad \text{return 2 // heads-tails} \\
& \quad \text{else} \\
& \quad \text{if flip}(0.5) \text{ then} \\
& \quad \quad \text{return 3 // tails-heads} \\
& \quad \quad \text{else} \\
& \quad \quad \text{skip // tails-tails}
\end{align*}
\]

\[
((r_1 \oplus_2 r_2) \oplus_2 (r_3 \oplus_2 1))^{(1)}
\]

? \equiv

\[
\text{if flip}(1/3) \text{ then} \\
& \quad \text{return 1} \\
\text{else} \\
& \quad \text{if flip}(0.5) \text{ then} \\
& \quad \quad \text{return 2} \\
\text{else} \\
& \quad \text{return 3}
\]

\[
r_1 \oplus_{1/3} (r_2 \oplus_{1/2} r_3)
\]
Operational model

Automata with the transition function of the type
\[ Q \times \text{At} \rightarrow D_\omega(\{\checkmark, \times\} + V + \text{Act} \times Q) \]

\[ b \quad q_0 \quad \bar{b} \]

\[ \begin{array}{c}
q_1 \\
\downarrow b, \bar{b}
\end{array} \]

\[ (p + b q) \oplus_{0.4} r \]
Operational model

Automata with the transition function of the type
\[ Q \times \text{At} \rightarrow D_\omega(\{\sqrt{\cdot}, X\} + V + \text{Act} \times Q) \]

Notion of equivalence: bisimulation associated with the type functor

\[ (p + b \cdot q) \oplus_{0.4} r \]
Operational model

Automata with the transition function of the type
\[ Q \times \text{At} \to D_\omega(\{\checkmark, \times\} + V + \text{Act} \times Q) \]

- 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)
Overview

- GKAT describes general equivalences of programs.
- It admits a complete axiomatization and is decidable.
- There is a model for the theory without $e0 \equiv 0$.
- 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
Nearly-linear complexity is $O(\alpha(n) \cdot n)$, where $\alpha$ 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):

\[
\text{if } a \text{ then } e \text{ else } f \text{ end } \mapsto a \cdot e + \bar{a} \cdot f
\]

\[
\text{while } a \text{ do } e \text{ end } \mapsto (a \cdot e)^* \cdot \bar{a}
\]

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

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

\[
\text{if } a \text{ then } e \text{ else } f \text{ end } \mapsto a \cdot e + \overline{a} \cdot f
\]

\[
\text{while } a \text{ do } e \text{ end } \mapsto (a \cdot e)^* \cdot \overline{a}
\]

Known results:

▶ There is a “nice” set of axioms for KAT.
▶ Soundness & completeness for a straightforward model.
▶ Equivalence according to these axioms is decidable.
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):
References


