Joint work with . . .

Todd Schmid (UCL)

Dexter Kozen (Cornell)

Alexandra Silva (Cornell, UCL)
Motivation: comparing programs

if not a then
  e;
else
  f;
end

≡

if a then
  f;
else
  e;
end
Motivation: comparing programs

if a then
  e;
  while a do
    e;
  end
end

≡

while a do
  e;
end
A more complicated equivalence

\[
\begin{align*}
\text{while } &a\text{ and }b \text{ do } \\
& e; \\
\text{end} \\
\text{while } &a \text{ do} \\
& f; \\
& \text{while } a\text{ and }b \text{ do} \\
& e; \\
& \text{end} \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\text{while } &a \text{ do} \\
& \text{if } b \text{ then} \\
& \quad e; \\
& \text{else} \\
& \quad f; \\
& \text{end} \\
\text{end}
\end{align*}
\]
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).

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

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

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

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

\[ \text{a or b} \]

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

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

\[
\begin{align*}
a, b & ::= t \in T \mid a + b \mid ab \mid a \mid 0 \mid 1 \\
e, f & ::= a \mid p \in \Sigma \mid ef \mid e + a f \mid e^{(a)}
\end{align*}
\]
Condensing the syntax

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

\[ \begin{align*}
    a, b & ::= t \in T \mid a + b \mid ab \mid \overline{a} \mid 0 \mid 1 \\
    e, f & ::= a \mid p \in \Sigma \mid ef \mid e + a f \mid e(a)
\end{align*} \]
Condensing the syntax

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

\[
\begin{align*}
a, b & ::= t \in T \mid a + b \mid ab \mid \overline{a} \mid 0 \mid 1 \\
e, f & ::= a \mid p \in \Sigma \mid ef \mid e + a f \mid e^{(a)}
\end{align*}
\]
Condensing the syntax

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

\[
\begin{align*}
\mathbf{a, b} &::= t \in T \mid a + b \mid ab \mid \overline{a} \mid 0 \mid 1 \\
\mathbf{e, f} &::= a \mid p \in \Sigma \mid ef \mid e +_a f \mid e^{(a)}
\end{align*}
\]
Condensing the syntax

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

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

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

assert a
Condensing the syntax

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

\[
\begin{align*}
a, b & \ ::= \ t \in T \mid a + b \mid ab \mid \overline{a} \mid 0 \mid 1 \\
e, f & \ ::= \ a \mid p \in \Sigma \mid ef \mid e + a f \mid e^{(a)}
\end{align*}
\]
Condensing the syntax

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

\[
\begin{align*}
\text{a, b} & ::= t \in T \mid a + b \mid ab \mid \overline{a} \mid 0 \mid 1 \\
\text{e, f} & ::= a \mid p \in \Sigma \mid ef \mid e +_a f \mid e^{(a)}
\end{align*}
\]
Condensing the syntax

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

\[
\begin{align*}
a, b &::= t \in T \mid a + b \mid ab \mid \overline{a} \mid 0 \mid 1 \\
e, f &::= a \mid p \in \Sigma \mid ef \mid e + a f \mid e^{(a)}
\end{align*}
\]

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

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

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

\[
\begin{align*}
  e + ae & \equiv e \\
  e + af & \equiv f + ae \\
  e + af & \equiv ae + af \\
  \bar{a}a & \equiv 0 \\
  0e & \equiv 0
\end{align*}
\]
Some example axioms

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

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

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

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

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

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

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

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

\[ \equiv 0 +_\overline{a} a e \]

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

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

\[
\text{if } a \text{ then } e \text{ else assert false} = e +_a 0 \equiv ae +_a 0 \\
\equiv 0 +_a ae \\
\equiv 0e +_a ae \\
\equiv \overline{aa}e +_a ae
\]
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 = \( e +_a 0 \equiv ae +_a 0 \)
\[ \equiv 0 +_a ae \]
\[ \equiv 0e +_a ae \]
\[ \equiv \overline{aa}e +_a ae \]
\[ \equiv ae +_a ae \]
Some example axioms

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

```
if \ a \ then \ e \ else \ assert \ false \ = \ e +_a 0 \equiv ae +_a 0 \\
\quad \equiv 0 +_a ae \\
\quad \equiv 0e +_a ae \\
\quad \equiv \overline{a}ae +_a ae \\
\quad \equiv ae +_a ae \\
\quad \equiv ae = assert a; e 
```
Guarded Kleene Algebra with Tests

\[ e + a e \equiv e \quad e + a f \equiv f + a e \quad (e + a f) + b g \equiv e + ab (f + b g) \]

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

\[ e0 \equiv 0 \quad 1e \equiv e \quad e1 \equiv e \quad e^{(a)} \equiv ee^{(a)} + a 1 \quad (e + a 1)^{(b)} \equiv (ae)^{(b)} \]
Guarded Kleene Algebra with Tests

**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

$$e_{1,1} \cdot x_1 + a_{1,1} \quad e_{1,2} \cdot x_2 + a_{1,2} \quad \cdots \quad e_{1,n} \cdot x_1 + a_{1,n} \quad b_1 \equiv x_1$$
$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$e_{n,1} \cdot x_1 + a_{n,1} \quad e_{n,2} \cdot x_2 + a_{n,2} \quad \cdots \quad e_{n,n} \cdot x_1 + a_{n,n} \quad b_n \equiv x_n$$

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 almost-linear time.
A more complicated equivalence

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

\[
\text{while } a \text{ and } b \text{ do}
\]
\[
\quad e;
\]
\[
\text{end}
\]
\[
\text{while } a \text{ do}
\]
\[
\quad f;
\]
\[
\quad \text{while } a \text{ and } b \text{ do}
\]
\[
\quad e;
\]
\[
\text{end}
\]
\[
\text{end}
\]

\[
\equiv
\]

\[
\text{while } a \text{ do}
\]
\[
\quad \text{if } b \text{ then}
\]
\[
\quad \quad e;
\]
\[
\quad \text{else}
\]
\[
\quad \quad f;
\]
\[
\text{end}
\]
\[
\text{end}
\]

\[
(e +_b f)^{(a)}
\]
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?

This talk:
- Answer to the first question.
- Progress on the second question.
Followup questions

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

This talk:
- Answer to the first question.
- Progress on the second question.

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

In particular,

```
while true do e end
```
But wait, there's more

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

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,

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

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

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 $= e^{(1)}$

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

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

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

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

In particular,

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

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

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

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

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

In particular,

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

\[ \equiv e^{(1)} \cdot \bar{1} \]

\[ \equiv e^{(1)} \cdot 0 \]

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

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

In particular,

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

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

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

$$\equiv 0 \quad = \text{assert false}$$

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
\[ \equiv_0 \text{ be like } \equiv, \text{ but without relating } \epsilon_0 \text{ to } 0. \]

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.
A tree where, for each set of tests $\alpha \subseteq T$, a node either...

- ...transitions to an “accept” or “reject” leaf node, or
- ...transitions to another internal node, executing an action $p \in \Sigma$. 

Note: guarded trees may be infinite!
Guarded trees — informal description

A tree where, for each set of tests $\alpha \subseteq T$, a node either . . .

▶ . . . transitions to an “accept” or “reject” leaf node, or
▶ . . . transitions to another internal node, executing an action $p \in \Sigma$.

Note: guarded trees may be infinite!
Guaraded trees — example
Expressions to trees — base case

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

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

\[
\begin{align*}
\{b\} & \mid p \\
\emptyset & \mid q \\
\{b\} & \mid r \\
\emptyset & \mid s \\
+_{b} & = \\
\{b\} & \mid p \\
\emptyset & \mid s
\end{align*}
\]
Expressions to trees — Party hat diagrams

\[
\begin{align*}
\{b\} & \quad \emptyset \mid p \\
\emptyset & \quad \{b\} \mid p
\end{align*}
\]
Expressions to trees — Party hat diagrams

\[
\left( \begin{array}{c}
\{b\} | p \\
\{b\}
\end{array} \right)^{(b)} = \left( \begin{array}{c}
\{b\} | p \\
\{b\} | p \\
\{b\} | p
\end{array} \right)
\]
A model in terms of guarded trees

Every expression $e$ has an associated guarded tree $[e]$. 
A model in terms of guarded trees

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

Theorem (Soundness & Completeness)

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

Theorem (Soundness & Completeness)
\( e \equiv_0 f \) if and only if \( \llbracket e \rrbracket = \llbracket f \rrbracket \)

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

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* \( \llbracket e \rrbracket = \llbracket f \rrbracket \).*

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

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

Theorem (Decidability for trees)

*It is decidable whether* \([e] = [f] \).

Corollary (Decidability for terms)

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

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

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* \[ \llbracket e \rrbracket = \llbracket f \rrbracket. \]

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 $\llbracket e \rrbracket = t$?
Expressiveness

Question

Let \( t \) be a guarded tree with finitely many distinct subtrees.

Is there an \( e \) such that \( [e] = t \)?

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 \([e] = t\)?

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

Not in general — for instance:

\[
\begin{aligned}
\{b\} & | p \\
\emptyset & | q \\
\{b\} & | p \\
\emptyset & | q \\
\{b\} & \\
\vdots
\end{aligned}
\]

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!

\[
\begin{align*}
\ell_0 : & \text{if } b \text{ then } p; \text{ goto } \ell_1 \text{ else accept} \\
\ell_1 : & \text{if } \overline{b} \text{ then } q; \text{ goto } \ell_0 \text{ else accept}
\end{align*}
\]

Not in general — for instance:

See also (Kozen and Tseng 2008).
Further work

Question

*Is it decidable whether, given a tree $t$, there exists an $e$ such that $[e] = t$?*
Further work

Question

Is it decidable whether, given a tree $t$, there exists an $e$ such that $[e] = t$?

Question

Can we identify rejection and looping without identifying early/late rejection?

What would be the appropriate axioms for such a semantics?
Overview

- GKAT describes general equivalences of programs.
- It admits a complete axiomatization and is decidable.
- The axiom $e_0 \equiv 0$ may not be what you want.
- There is a model for the theory without this axiom.
- Soundness and completeness can be recovered.
- Lack of GOTO means not every tree is expressible.

https://kap.pe/slides

https://doi.org/10.4230/LIPIcs.ICALP.2021.142
Syntax is special case of Kleene Algebra with Tests (KAT):

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

\[
\text{while } a \text{ do } e \text{ end } \mapsto (a \cdot e)^* \cdot \overline{a}
\]
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.
Bonus — Reduction to KAT

Equivalence in KAT is \textsc{PSPACE}-complete (Cohen, Kozen, and Smith 1996).
Bonus — Reduction to KAT

Equivalence in KAT is \textsc{pspace}-complete (Cohen, Kozen, and Smith 1996).

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

![Time to solve vs. Term Size](image_url)
References


