Joint work with . . .

Todd Schmid  
(UCL, Nijmegen)

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

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
Research questions

▶ What is the minimal set of axioms?
▶ Are those axioms sound and complete for a 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).

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

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

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

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

false

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

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

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

assert a
Condensing the syntax

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

\[ a, b ::= \begin{array}{l} t \in T \ | \ a + b \ | \ ab \ | \ \overline{a} \ | \ 0 \ | \ 1 \\
\end{array} \]

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

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

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

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

\[ e; f \]
Condensing the syntax

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

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

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

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

\[
\begin{align*}
e +_a e & \equiv e \\
e +_a f & \equiv f +_a e
\end{align*}
\]
Some example axioms

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

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

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

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

\[ e + a \ e \equiv e \]

\[ e + a \ f \equiv f + a \ e \]

\[ e + a \ f \equiv a e + a \ f \]

\[ \overline{a} a \equiv 0 \]

\[ 0 e \equiv 0 \]

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

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

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

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

\[
\text{if } a \text{ then } e \text{ else assert false} = e + a 0 \equiv ae + a 0
\]

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

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

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

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

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

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

\[
\begin{align*}
  e + a \ e &\equiv e & e + a \ f &\equiv f + a \ e & (e + a \ f) + b \ g &\equiv e + a b (f + b \ g) & e + a \ f &\equiv a e + a f \\
  eg + a \ fg &\equiv (e + a \ f)g & (ef)g &\equiv e (fg) & 0e &\equiv 0 & e0 &\equiv 0 & 1e &\equiv e & e1 &\equiv e \\
  e^{(a)} &\equiv ee^{(a)} + a \ 1 & (e + a \ 1)^{(b)} &\equiv (ae)^{(b)} & g &\equiv eg + a \ f & \Rightarrow & g &\equiv e^{(a)} f
\end{align*}
\]
Guarded Kleene Algebra with Tests

\[
\begin{align*}
e + a\ e & \equiv e & e + a\ f & \equiv f + a\ e & (e + a\ f) + b\ g & \equiv e + a\ b\ (f + b\ g) & e + a\ f & \equiv a\ e + a\ f \\
& & eg + a\ fg & \equiv (e + a\ f)g & (ef)g & \equiv e(fg) & 0e & \equiv 0 & e0 & \equiv 0 & 1e & \equiv e & e1 & \equiv e \\
& & e^{(a)} & \equiv ee^{(a)} + a\ 1 & (e + a\ 1)^{(b)} & \equiv (ae)^{(b)} & g & \equiv eg + a\ f & \Rightarrow & g & \equiv e^{(a)}f
\end{align*}
\]

it’s a bit more subtle than this…
Guarded Kleene Algebra with Tests

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

\[ eg +_a fg \equiv (e +_a f)g \quad (ef)g \equiv e(fg) \quad 0e \equiv 0 \quad e0 \equiv 0 \quad 1e \equiv e \quad e1 \equiv e \]

\[ e^{(a)} \equiv ee^{(a)} +_a 1 \quad (e +_a 1)^{(b)} \equiv (ae)^{(b)} \quad g \equiv eg +_a f \quad \Rightarrow \quad g \equiv e^{(a)}f \]

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

\[ e^{(ab)} \cdot (fe^{(ab)})^{(a)} \equiv (e + _b f)^{(a)} \]
Open questions

- What if we drop the axiom $e_0 \equiv 0$?
- How expressive is this syntax?
- Funny business with the last axiom.
Open questions

- What if we drop the axiom $e^0 \equiv 0$?
- How expressive is this syntax?
- Funny business with the last axiom.

This talk:
- Answer to the first question.
- Progress towards answering the second question.
- Third problem is very hard...
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: $e^{(a)} \equiv e^{(a)\overline{a}}$.

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,

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

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

In particular,

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

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

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

In particular,

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

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

In particular,

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

See also (Mamouras 2017).
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 $e_0$ 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 — 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!
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!
Guarded trees — example
Expressions to trees — base case

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

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

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

\[
\left( \left\{ b \right\} \mid p \right) \overset{(b)}{=} \left\{ b \right\} \mid \emptyset
\]
A model in terms of guarded trees

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

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 \textit{not} hold: \([e0]\) \(\neq\) \([0]\).
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: $\llbracket e 0 \rrbracket \neq \llbracket 0 \rrbracket$.

Question (Soundness & Completeness)

Is $e \equiv_0 f$ equivalent to $\llbracket e \rrbracket = \llbracket f \rrbracket$?
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)
\[ e \equiv_0 f \text{ if and only if } \llbracket e \rrbracket = \llbracket f \rrbracket \]

Theorem (Decidability for trees)
*It is decidable whether* \( \llbracket e \rrbracket = \llbracket f \rrbracket \).
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)
\( e \equiv_0 f \) 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 \( \llbracket e \rrbracket = t \)?

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 $[[e]] = t$?

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!

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

Not in general — for instance:

\[
\begin{align*}
\{ b \} | p \\
\emptyset | q \\
\{ b \} | p \\
\emptyset | q \\
\{ b \} \\
\vdots
\end{align*}
\]

See also (Kozen and Tseng 2008).
Further work

Question

Is it decidable whether, given a tree $t$, there exists an $e$ such that $\llbracket e \rrbracket = t$?
Further work

Question
Is it decidable whether, given a tree $t$, there exists an $e$ such that $\llbracket e \rrbracket = t$?

Question
Can we identify rejection and looping without identifying early/late rejection?
What would be the appropriate axioms for such a semantics?
Thank you

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.
Equivalence in KAT is \textit{PSPACE}-complete (Cohen, Kozen, and Smith 1996).
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):


