Proving Liveness of Parameterized Programs

Zachary Kincaid
University of Toronto & Princeton University

July 5, 2016

Joint work with:
Azadeh Farzan, University of Toronto

Andreas Podelski, University of Freiburg

```
global t : int              // ticket counter
global s : int              // service counter
local m : int               // my ticket
init s = t

do forever {
    m := t++                // acquire ticket
    do {
                            // busy wait
    } until (m <= s)
    // critical section
    s++                     // bump service counter
}
```

```
global t : int            // ticket counter
global s : int            // service counter
local m : int             // my ticket
init s = t

do forever {
    m := t++              // acquire ticket
    do {
                          // busy wait
    } until (m <= s)
    // critical section
    s++                   // bump service counter
}
```

Goal: Prove that no thread starves

```
global t : int            // ticket counter
global s : int            // service counter
local m : int             // my ticket
init s = t

do forever {
    m := t++              // acquire ticket
    do {
                          // busy wait
    } until (m <= s)
    // critical section
    s++                   // bump service counter
}
```

Goal: Prove that no thread starves

- no matter how many threads there are

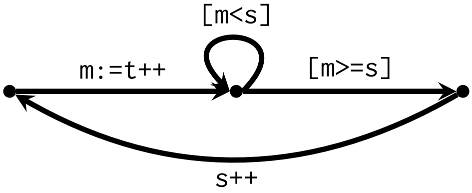
```
global t : int                  // ticket counter
global s : int                  // service counter
local m : int                   // my ticket
init s = t

do forever {
    m := t++                    // acquire ticket
    do {
                                // busy wait
    } until (m <= s)
    // critical section
    s++                         // bump service counter
}
```

Goal: Prove that no thread starves

- no matter how many threads there are
- automatically

A parameterized concurrent program, P:

- *thread template* = finite directed graph with edges labeled by instructions (in some programming language). Call the set of instructions Σ.
- For any $N \in \mathbb{N}$, $P(N)$ denotes the program with N identical threads, all of which execute P.



Thread identifiers

A **trace** is a sequence $\tau = \langle \sigma_1 : i_1 \rangle \langle \sigma_2 : i_2 \rangle ... \in (\Sigma \times \mathbb{N})^\omega$

Program instructions

A **trace** is a sequence $\tau = \langle \sigma_1 : i_1 \rangle \langle \sigma_2 : i_2 \rangle ... \in (\Sigma \times \mathbb{N})^\omega$

- Associate linear-time property Φ w/ set of traces $\mathcal{L}(\Phi)$ that satisfy it.

A **trace** is a sequence $\tau = \langle \sigma_1 : i_1 \rangle \langle \sigma_2 : i_2 \rangle ... \in (\Sigma \times \mathbb{N})^\omega$

- Associate linear-time property Φ w/ set of traces $\mathcal{L}(\Phi)$ that satisfy it.
- Associate $P(N)$ w/ set of traces $\mathcal{L}(P(N)) \subseteq (\Sigma \times \{1, ..., N\})^\omega$ corresponding to interleaved paths through the thread template

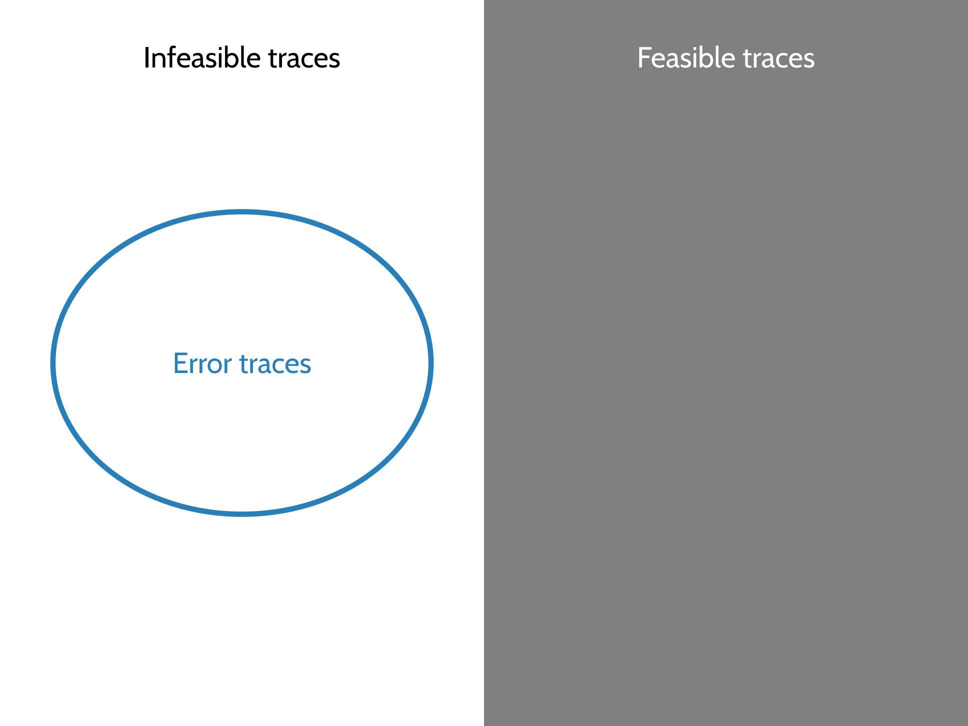
A **trace** is a sequence $\tau = \langle \sigma_1 : i_1 \rangle \langle \sigma_2 : i_2 \rangle ... \in (\ \Sigma \times \mathbb{N}\ )^\omega$

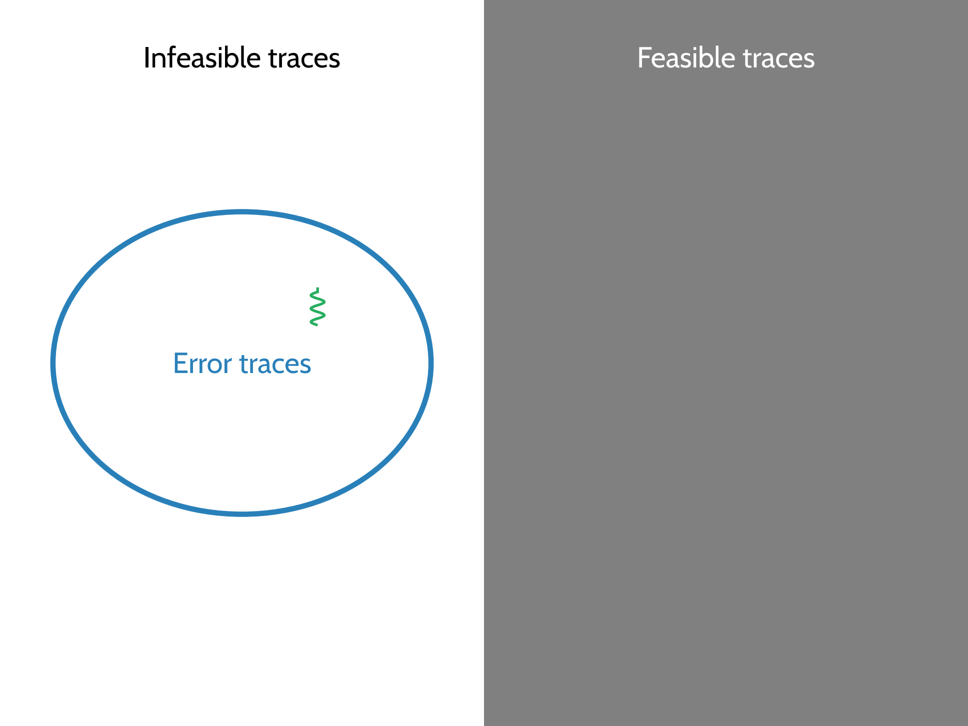
- Associate linear-time property Φ w/ set of traces $\mathcal{L}(\Phi)$ that satisfy it.
- Associate $P(N)$ w/ set of traces $\mathcal{L}(P(N)) \subseteq (\Sigma \times \{1, ..., N\})^\omega$ corresponding to interleaved paths through the thread template
- Program traces $\mathcal{L}(P) = \bigcup_N \mathcal{L}(P(N))$

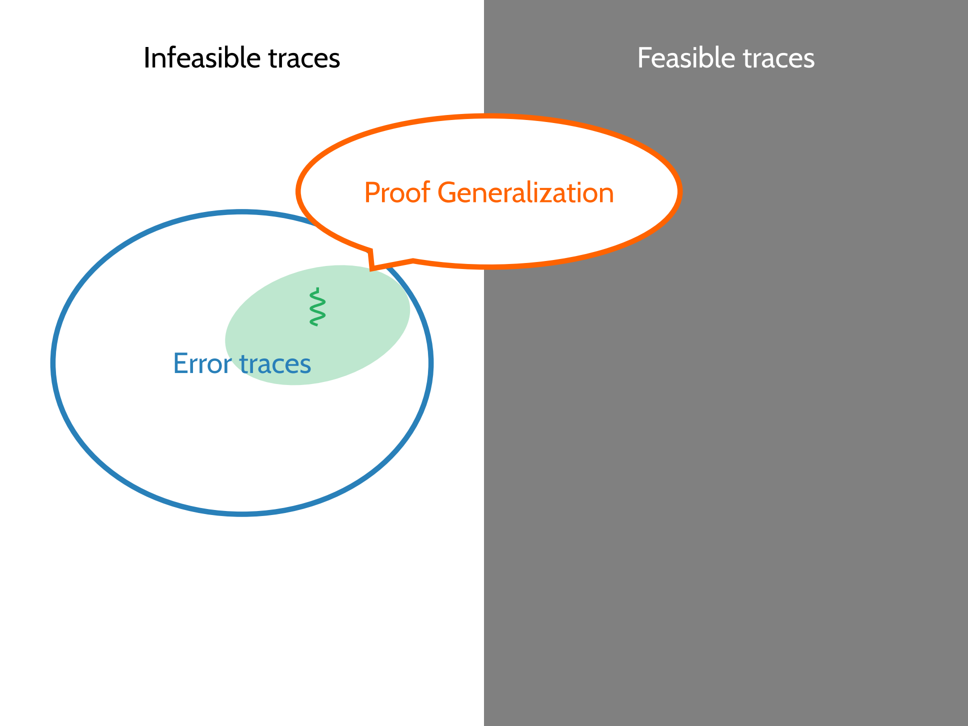
A **trace** is a sequence $\tau = \langle \sigma_1 : i_1 \rangle \langle \sigma_2 : i_2 \rangle ... \in (\Sigma \times \mathbb{N})^\omega$

- Associate linear-time property Φ w/ set of traces $\mathcal{L}(\Phi)$ that satisfy it.
- Associate $P(N)$ w/ set of traces $\mathcal{L}(P(N)) \subseteq (\Sigma \times \{1, ..., N\})^\omega$ corresponding to interleaved paths through the thread template
- Program traces $\mathcal{L}(P) = \bigcup_N \mathcal{L}(P(N))$
- P correct $\iff$ every **error trace** in $\mathcal{L}(P) \setminus \mathcal{L}(\Phi)$ is infeasible.

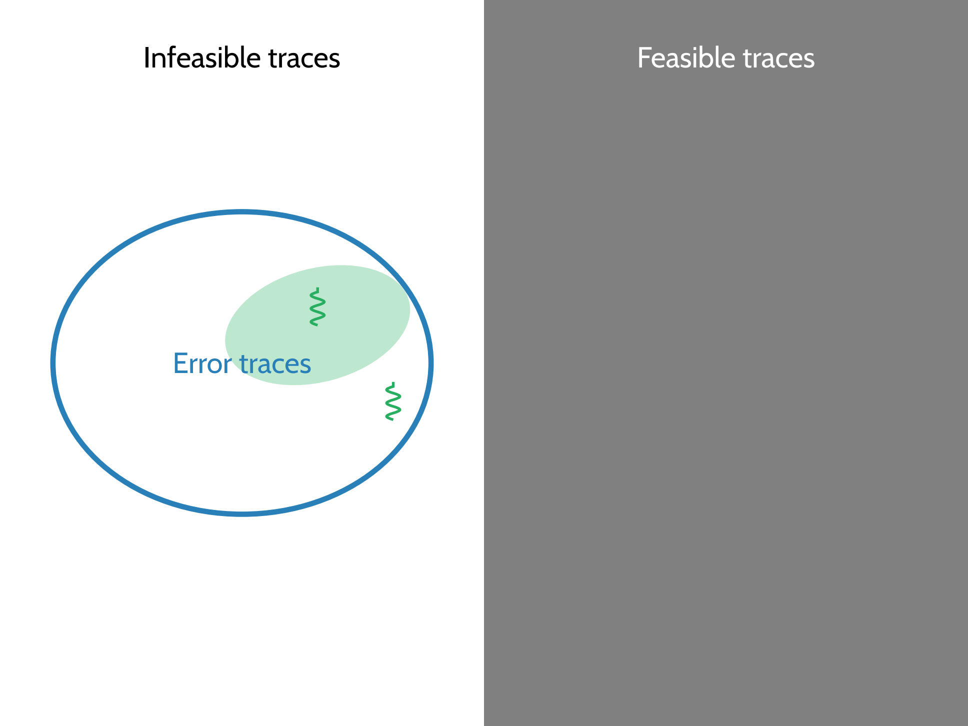
Infeasible traces	Feasible traces
No corresponding executions	At least one corresponding execution

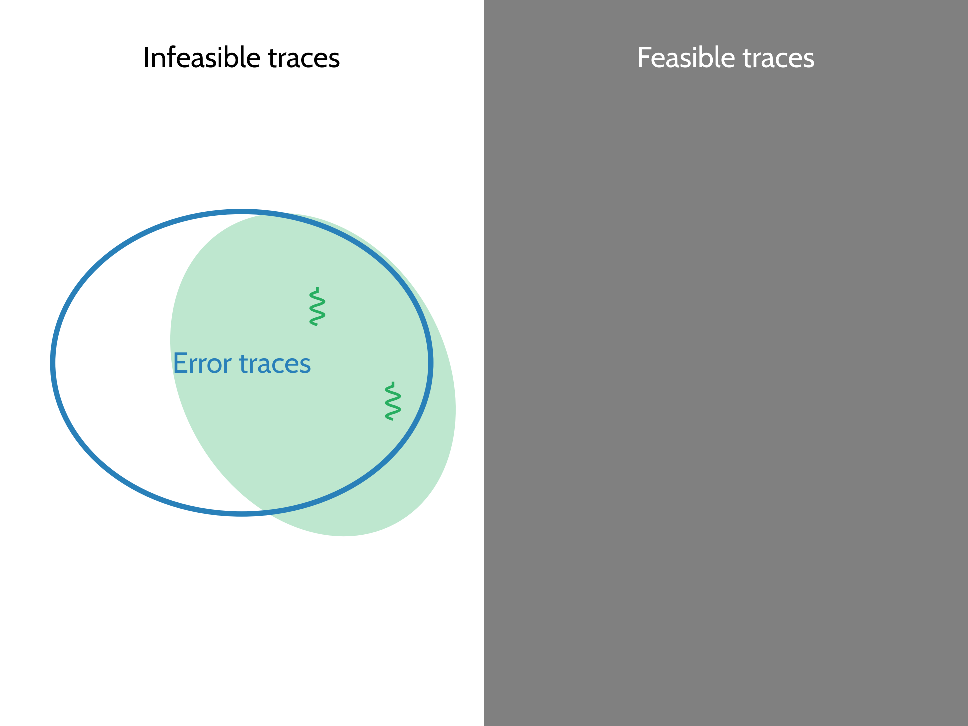

Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Error traces

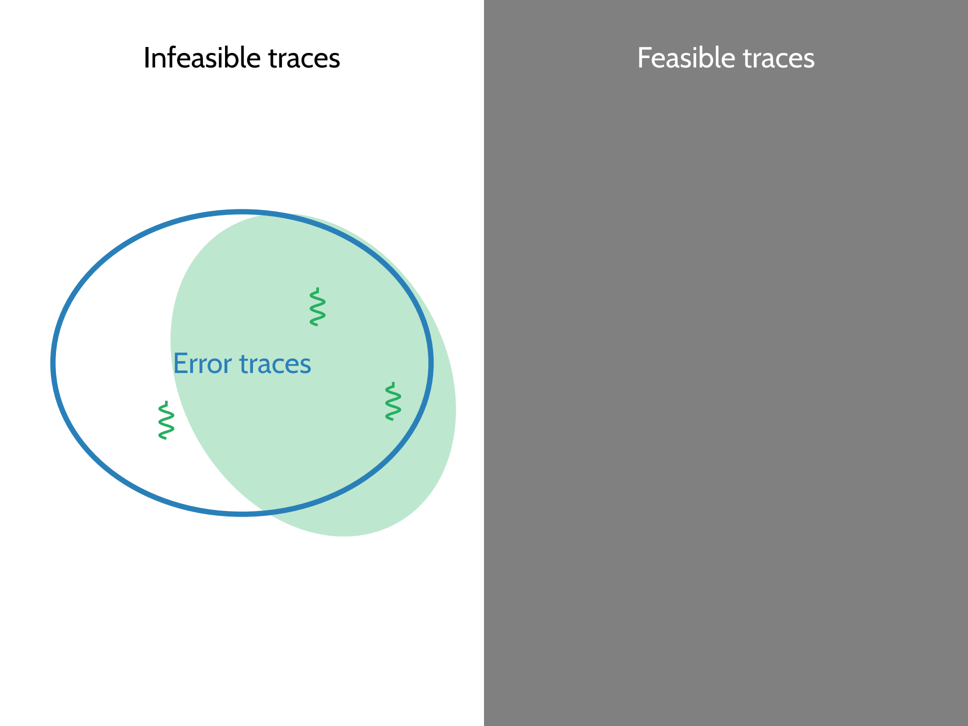


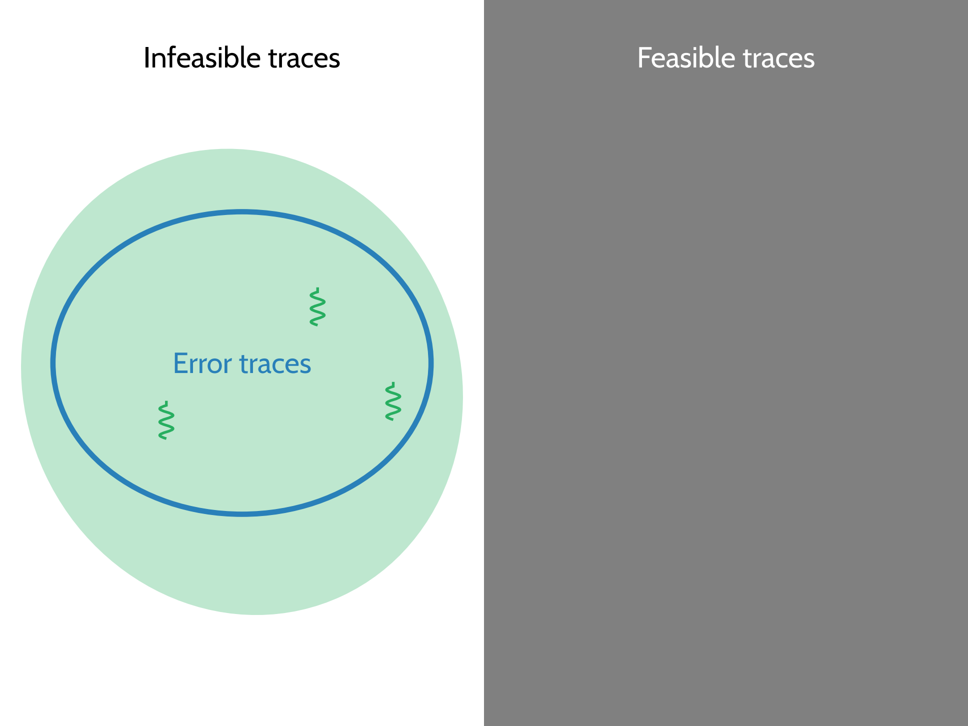
Infeasible traces
Feasible traces
Proof Generalization
Error traces

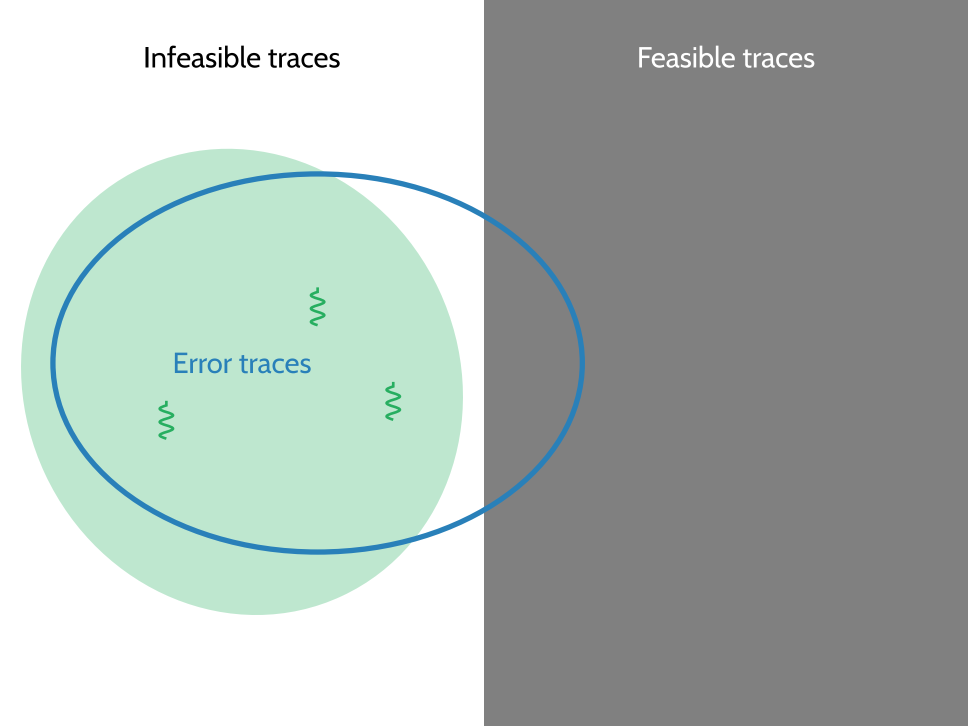

Infeasible traces
Feasible traces
Error traces

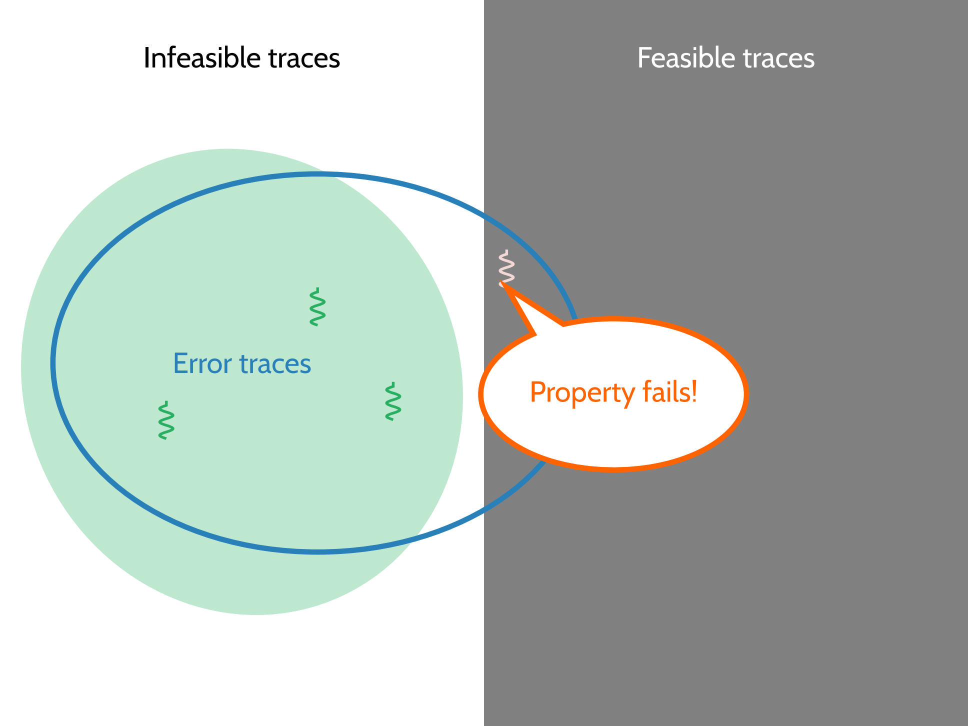


Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Error traces
Property fails!

Two key problems:

1. How do we generalize proofs?

2. How do we check that a proof is complete?

Two key problems:

1. How do we generalize proofs?
    - Concurrency: Same proof applies to many interleavings.

2. How do we check that a proof is complete?

Two key problems:

1. How do we generalize proofs?
   - Concurrency: Same proof applies to many interleavings.
   - Parameterization: Same proof applies to many instantiations.
2. How do we check that a proof is complete?

$$\underbrace{\langle \texttt{m:=t++} : 1\rangle\langle \texttt{m:=t++} : 2\rangle}_{\textsf{Stem}}\ (\underbrace{\langle \texttt{[m>s]} : 2\rangle\langle \texttt{[m<=s]} : 1\rangle\langle \texttt{s++} : 1\rangle\langle \texttt{m:=t++} : 1\rangle}_{\textsf{Loop}})^{\omega}$$

$$\underbrace{\langle \texttt{m:=t++} : 1\rangle\langle \texttt{m:=t++} : 2\rangle}_{\text{Stem}}\ \big(\underbrace{\langle \texttt{[m>s]} : 2\rangle\langle \texttt{[m<=s]} : 1\rangle\langle \texttt{s++} : 1\rangle\langle \texttt{m:=t++} : 1\rangle}_{\text{Loop}}\big)^{\omega}$$

$$\begin{gathered}
\{\textit{old}(\mathtt{s}) = \mathtt{s}\} \\
\langle \texttt{[m>s]} : 2\rangle \\
\{\textit{old}(\mathtt{s}) = \mathtt{s} \wedge \mathtt{m}(2) \geq \textit{old}(\mathtt{s})\} \\
\langle \texttt{[m<=s]} : 1\rangle \\
\{\textit{old}(\mathtt{s}) = \mathtt{s} \wedge \mathtt{m}(2) \geq \textit{old}(\mathtt{s})\} \\
\langle \texttt{s++} : 1\rangle \\
\{\textit{old}(\mathtt{s}) < \mathtt{s} \wedge \mathtt{m}(2) \geq \textit{old}(\mathtt{s})\} \\
\langle \texttt{m:=t++} : 1\rangle \\
\{\ \textit{old}(\mathtt{s}) < \mathtt{s} \wedge \mathtt{m}(2) \geq \textit{old}(\mathtt{s})\ \}
\end{gathered}$$

Ranking formula

Variance proof

$$\underbrace{\langle \texttt{m:=t++} : 1\rangle\langle \texttt{m:=t++} : 2\rangle}_{\text{Stem}} \; (\underbrace{\langle \texttt{[m>s]} : 2\rangle\langle \texttt{[m<=s]} : 1\rangle\langle \texttt{s++} : 1\rangle\langle \texttt{m:=t++} : 1\rangle}_{\text{Loop}})^{\omega}$$

Variance proof

$$\{\mathit{old}(\texttt{s}) = \texttt{s}\}$$
$$\langle \texttt{[m>s]} : 2\rangle$$
$$\{\mathit{old}(\texttt{s}) = \texttt{s} \wedge \texttt{m}(2) \geq \mathit{old}(\texttt{s})\}$$
$$\langle \texttt{[m<=s]} : 1\rangle$$
$$\{\mathit{old}(\texttt{s}) = \texttt{s} \wedge \texttt{m}(2) \geq \mathit{old}(\texttt{s})\}$$
$$\langle \texttt{s++} : 1\rangle$$
$$\{\mathit{old}(\texttt{s}) < \texttt{s} \wedge \texttt{m}(2) \geq \mathit{old}(\texttt{s})\}$$
$$\langle \texttt{m:=t++} : 1\rangle$$
$$\{\; \mathit{old}(\texttt{s}) < \texttt{s} \wedge \texttt{m}(2) \geq \mathit{old}(\texttt{s}) \;\}$$

Invariance proof

$$\{s = t\}$$
$$\langle \texttt{m:=t++} : 1\rangle$$
$$\{\mathit{true}\}$$
$$\langle \texttt{m:=t++} : 2\rangle$$
$$\{\mathit{true}\}$$
$$\langle \texttt{[m>s]} : 2\rangle$$
$$\{\mathit{true}\}$$
$$\langle \texttt{[m<=s]} : 1\rangle$$
$$\{\mathit{true}\}$$
$$\langle \texttt{s++} : 1\rangle$$
$$\{\mathit{true}\}$$
$$\langle \texttt{m:=t++} : 1\rangle$$
$$\{\mathit{true}\}$$

$$
\begin{array}{ccc}
\{\textit{\textbf{old}}(\mathsf{s}) = \mathsf{s}\} & \{\textit{\textbf{old}}(\mathsf{s}) = \mathsf{s}\} & \{\varphi\} \\
\langle \texttt{[m>s]} : 2 \rangle & \langle \mathsf{s{+}{+}} : 1 \rangle & \langle \sigma : i \rangle \\
\{\texttt{m}(2) \geq \textit{\textbf{old}}(\mathsf{s})\} & \{\textit{\textbf{old}}(\mathsf{s}) < \mathsf{s}\} & \{\varphi\}
\end{array}
$$

Sequencing

$$\{s \leq t\}$$
$$\texttt{m := t++} : \mathbf{1}$$
$$\{m(1) < t\}$$

$$\{m(1) < t\}$$
$$\texttt{m := t++} : \mathbf{2}$$
$$\{m(1) < m(2)\}$$

Sequencing

$\{\mathsf{s} \leq \mathsf{t}\}$
`m := t++ : 1`
$\{\mathtt{m}(1) < \mathsf{t}\}$

$\{\mathtt{m}(1) < \mathsf{t}\}$
`m := t++ : 2`
$\{\mathtt{m}(1) < \mathtt{m}(2)\}$

$\{\mathsf{s} \leq \mathsf{t}\}$
`m := t++ : 1;`
`m := t++ : 2`
$\{\mathtt{m}(1) < \mathtt{m}(2)\}$

Symmetry

$$P(N) = \underbrace{P \parallel P \parallel \cdots \parallel P}_{N \text{ times}}$$

$$\{\mathsf{s} \leq \mathtt{m}(1) \wedge \mathtt{m}(1) < \mathtt{m}(2)\}$$

`[m <= s]`: **2**

$\{$*false*$\}$

Symmetry

$$P(N) = \underbrace{P \parallel P \parallel \cdots \parallel P}_{N \text{ times}}$$

$$\{\mathsf{s} \le \mathtt{m}(1) \wedge \mathtt{m}(1) < \mathtt{m}(2)\} \quad \xrightarrow[{[2 \mapsto 1]}]{[1 \mapsto 2]} \quad \{\mathsf{s} \le \mathtt{m}(2) \wedge \mathtt{m}(2) < \mathtt{m}(1)\}$$

`[m <= s]` : 2 → `[m <= s]` : 1

{*false*} → {*false*}

Symmetry

$$P(N) = \underbrace{P \parallel P \parallel \cdots \parallel P}_{N \text{ times}}$$

$$\{\mathsf{s} \leq \mathtt{m}(1) \wedge \mathtt{m}(1) < \mathtt{m}(2)\}$$
`[m <= s]` : **2**
$\{\textit{false}\}$

$$\xrightarrow[{[2 \mapsto 3]}]{[1 \mapsto 2]}$$

$$\{\mathsf{s} \leq \mathtt{m}(2) \wedge \mathtt{m}(2) < \mathtt{m}(3)\}$$
`[m <= s]` : **3**
$\{\textit{false}\}$

Conjunction

$\{\mathtt{m}(1) < t\}$
`m := t++:3`
$\{\mathtt{m}(1) < \mathtt{m}(3)\}$

$\{\mathtt{m}(2) < t\}$
`m := t++:3`
$\{\mathtt{m}(2) < \mathtt{m}(3)\}$

Conjunction

$\{\mathtt{m}(1) < t\}$
`m := t++` : 3
$\{\mathtt{m}(1) < \mathtt{m}(3)\}$

$\{\mathtt{m}(2) < t\}$
`m := t++` : 3
$\{\mathtt{m}(2) < \mathtt{m}(3)\}$

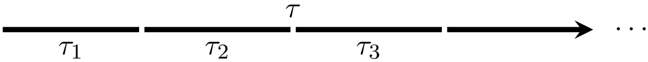
$\{\mathtt{m}(1) < t \wedge \mathtt{m}(2) < t\}$
`m := t++` : 3
$\{\mathtt{m}(1) < \mathtt{m}(3) \wedge \mathtt{m}(2) < \mathtt{m}(3)\}$

A *Well-founded proof space* (WFPS) $\langle H, R \rangle$ is a set of valid Hoare triples H which is closed under sequencing, symmetry, and conjunction, along with a set of ranking formulas R which is closed under symmetry.

A *Well-founded proof space* (WFPS) $\langle H, R \rangle$ is a set of valid Hoare triples H which is closed under sequencing, symmetry, and conjunction, along with a set of ranking formulas R which is closed under symmetry.

H is a set of theorems about *finite* traces. How do we prove infeasibility of *infinite* traces?

A WFPS $\langle H, R\rangle$ proves a trace τ infeasible if there is some ranking formula $r \in R$, some decomposition of τ:

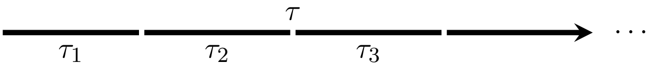


and some sequence of "intermediate formulas" $\varphi_1, \varphi_2, \ldots$ such that

$$\{\textsf{pre}\}\tau_1\{\varphi_1\} \qquad \{\varphi_1 \land \textit{old}(\mathbf{x}) = \mathbf{x}\}\tau_2\{r\}$$
$$\{\textsf{pre}\}\tau_1\tau_2\{\varphi_2\} \qquad \{\varphi_2 \land \textit{old}(\mathbf{x}) = \mathbf{x}\}\tau_3\{r\}$$
$$\vdots$$
$$\{\textsf{pre}\}\tau_1\tau_2\ldots\tau_i\{\varphi_i\} \qquad \{\varphi_i \land \textit{old}(\mathbf{x}) = \mathbf{x}\}\tau_{i+1}\{r\}$$

all belong to H.

A WFPS $\langle H, R \rangle$ proves a trace τ infeasible if there is some ranking formula $r \in R$, some decomposition of τ:



and some sequence of "intermediate formulas" $\varphi_1, \varphi_2, ...$ such that

$$\{\textsf{pre}\}\tau_1\{\varphi_1\} \qquad \{\varphi_1 \land \textit{old}(\mathbf{x}) = \mathbf{x}\}\tau_2\{r\}$$
$$\{\textsf{pre}\}\tau_1\tau_2\{\varphi_2\} \qquad \{\varphi_2 \land \textit{old}(\mathbf{x}) = \mathbf{x}\}\tau_3\{r\}$$
$$\vdots$$
$$\{\textsf{pre}\}\tau_1\tau_2...\tau_i\{\varphi_i\} \qquad \{\varphi_i \land \textit{old}(\mathbf{x}) = \mathbf{x}\}\tau_{i+1}\{r\}$$

all belong to H.
The set of traces $\langle H, R \rangle$ proves infeasible is denoted $\omega(H, R)$.

Two key problems:

1. How do we generalize proofs?
   - Concurrency: Same proof applies to many interleavings.
   - Parameterization: Same proof applies to many instantiations.
2. How do we check that a proof is complete?

Two key problems:

1. How do we generalize proofs?
   - Concurrency: Same proof applies to many interleavings.
   - Parameterization: Same proof applies to many instantiations.
2. How do we check that a proof is complete?
   - $\mathcal{L}(P) \setminus \mathcal{L}(\Phi) \subseteq \omega(H, R)$: inclusion between infinite sets of infinite words over an infinite alphabet

Infinite traces $\to$ finite traces

An **ultimately periodic trace** is a trace of the form $\pi\rho\rho\rho\cdots$
Every ultimately periodic trace can be written (*not uniquely*) as a **lasso** $\pi\$\rho$.
Given a language $L \subseteq \Sigma^\omega$, define its *lasso language* $\$(L)$ as:

$$\$(L) = \{\pi\$\rho : \pi\rho^\omega \in L\}$$

Infinite traces $\rightarrow$ finite traces

An **ultimately periodic trace** is a trace of the form $\pi\rho\rho\rho\cdots$
Every ultimately periodic trace can be written (*not uniquely*) as a **lasso** $\pi\$\rho$.
Given a language $L \subseteq \Sigma^\omega$, define its *lasso language* $\$(L)$ as:

$$\$(L) = \{\pi\$\rho : \pi\rho^\omega \in L\}$$

Theorem

If $\$(\mathcal{L}(P)) \setminus \$(\mathcal{L}(\Phi)) \subseteq \$(\omega(H, R))$*, then* $\mathcal{L}(P) \setminus \mathcal{L}(\Phi) \subseteq \omega(H, R)$.

Infinite traces $\to$ finite traces

An **ultimately periodic trace** is a trace of the form $\pi\rho\rho\rho\cdots$
Every ultimately periodic trace can be written (*not uniquely*) as a **lasso** $\pi\$\rho$.
Given a language $L \subseteq \Sigma^\omega$, define its *lasso language* $\$(L)$ as:

$$\$(L) = \{\pi\$\rho : \pi\rho^\omega \in L\}$$

Theorem

If $\$(\mathcal{L}(P)) \setminus \$(\mathcal{L}(\Phi)) \subseteq \$(\omega(H, R))$, *then* $\mathcal{L}(P) \setminus \mathcal{L}(\Phi) \subseteq \omega(H, R)$.

- For any $N \in \mathbb{N}$, $\mathcal{L}(P) \cap (\Sigma \times \{1, ..., N\})^\omega$ is ω-regular. Same for $\mathcal{L}(\Phi)$ and $\omega(H, R)$.
- Fact: If L_1 and L_2 are ω-regular, then $UP(L_1) \subseteq L_2$ implies $L_1 \subseteq L_2$.

Infinite language $\to$ automaton

Quantified Predicate Automata (QPA): a class of infinite-state automata that recognize words over an infinite alphabet.

Infinite language $\rightarrow$ automaton

Quantified Predicate Automata (QPA): a class of infinite-state automata that recognize words over an infinite alphabet.

- There is a QPA that recognizes $\$(\mathcal{L}(P))$.

Infinite language $\rightarrow$ automaton

Quantified Predicate Automata (QPA): a class of infinite-state automata that recognize words over an infinite alphabet.

- There is a QPA that recognizes $\$(\mathcal{L}(P))$.
- There is a QPA that recognizes $\$(\mathcal{L}(\Phi))$.

Infinite language $\rightarrow$ automaton

Quantified Predicate Automata (QPA): a class of infinite-state automata that recognize words over an infinite alphabet.

- There is a QPA that recognizes $\$(\mathcal{L}(P))$.
- There is a QPA that recognizes $\$(\mathcal{L}(\Phi))$.
- There is *not* a QPA that recognizes $\$(\omega(H, R))$.

But...

There *is* a QPA that recognizes all lassos $\pi\$\rho$ such that there exists some intermediate assertion φ and some ranking formula $r \in R$ such that

$$\{\text{pre}\}\pi\{\varphi\} \qquad \text{and} \qquad \{\varphi \wedge \textit{old}(\mathbf{x}) = \mathbf{x}\}\rho\{r\}$$

belong to H. Call this language $\$(H, R)$.

But...

There *is* a QPA that recognizes all lassos $\pi\$\rho$ such that there exists some intermediate assertion φ and some ranking formula $r \in R$ such that

$$\{\text{pre}\}\pi\{\varphi\} \qquad \text{and} \qquad \{\varphi \wedge \textit{old}(\mathbf{x}) = \mathbf{x}\}\rho\{r\}$$

belong to H. Call this language $\$(H, R)$.

Membership of $\pi\$\rho$ in $\$(H, R)$ does not imply that $\pi\rho^\omega \in \omega(H, R)$. It does not even imply that $\pi\rho^\omega$ is infeasible!

But...

There *is* a QPA that recognizes all lassos $\pi\$\rho$ such that there exists some intermediate assertion φ and some ranking formula $r \in R$ such that

$$\{\text{pre}\}\pi\{\varphi\} \qquad \text{and} \qquad \{\varphi \land \textit{old}(\mathbf{x}) = \mathbf{x}\}\rho\{r\}$$

belong to H. Call this language $\$(H, R)$.

Membership of $\pi\$\rho$ in $\$(H, R)$ does not imply that $\pi\rho^\omega \in \omega(H, R)$. It does not even imply that $\pi\rho^\omega$ is infeasible!

Theorem

If $\$(\mathcal{L}(P)) \setminus \$(\mathcal{L}(\Phi)) \subseteq \$(H, R)$*, then* $\mathcal{L}(P) \setminus \mathcal{L}(\Phi) \subseteq \omega(H, R)$.

But...

There *is* a QPA that recognizes all lassos $\pi\$\rho$ such that there exists some intermediate assertion φ and some ranking formula $r \in R$ such that

$$\{\text{pre}\}\pi\{\varphi\} \qquad \text{and} \qquad \{\varphi \wedge \textit{old}(\mathbf{x}) = \mathbf{x}\}\rho\{r\}$$

belong to H. Call this language $\$(H, R)$.

Membership of $\pi\$\rho$ in $\$(H, R)$ does not imply that $\pi\rho^\omega \in \omega(H, R)$. It does not even imply that $\pi\rho^\omega$ is infeasible!

Theorem

If $\$(\mathcal{L}(P)) \setminus \$(\mathcal{L}(\Phi)) \subseteq \$(H, R)$, *then* $\mathcal{L}(P) \setminus \mathcal{L}(\Phi) \subseteq \omega(H, R)$.

- $\pi\rho^\omega \in \mathcal{L}(P) \setminus \mathcal{L}(\Phi) \Rightarrow \pi\rho^n\$\rho^k \in \$(\mathcal{L}(P)) \setminus \$(\mathcal{L}(\Phi))$ for all $n \geq 0, k \geq 1$.
- H contains $\{\text{pre}\}\pi\rho^n\{\varphi_{n,k}\}$ and $\{\varphi_{n,k} \wedge \textit{old}(\mathbf{x}) = \mathbf{x}\}\rho^k\{r_{n,k}\}$. Ramsey!

But...

There *is* a QPA that recognizes all lassos $\pi\$\rho$ such that there exists some intermediate assertion φ and some ranking formula $r \in R$ such that

$$\{\text{pre}\}\pi\{\varphi\} \qquad \text{and} \qquad \{\varphi \wedge \textit{old}(\mathbf{x}) = \mathbf{x}\}\rho\{r\}$$

belong to H. Call this language $\$(H, R)$.

Membership of $\pi\$\rho$ in $\$(H, R)$ does not imply that $\pi\rho^\omega \in \omega(H, R)$. It does not even imply that $\pi\rho^\omega$ is infeasible!

Theorem

If $\$(\mathcal{L}(P)) \setminus \$(\mathcal{L}(\Phi)) \subseteq \$(H, R)$*, then* $\mathcal{L}(P) \setminus \mathcal{L}(\Phi) \subseteq \omega(H, R)$.

QPA language containment can be used to check proofs

Summary

Two key problems:

1. How do we generalize proofs?
   - **Well-founded proof spaces**
2. How do we check that a proof is complete?
   - **Lassos** + **Quantified Predicate Automata**

Thanks!