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
    }

Goal: Prove that no thread starves
• no matter how many threads there are
• automatically
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 \( t : \text{int} \)  // ticket counter
global \( s : \text{int} \)  // service counter
local \( m : \text{int} \)  // my ticket
init \( s = t \)

\[
\text{do forever} \ {\{}
\quad m := t++ \quad \text{// acquire ticket}
\quad \text{do} \ {\{}
\qquad \text{// busy wait}
\quad \text{until} \ (m <= s) \quad \text{// critical section}
\quad s++ \quad \text{// bump service counter}
\quad \text{\}}
\text{\}}
\]

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 $\Sigma$.
- For any $N \in \mathbb{N}$, $P(N)$ denotes the program with $N$ identical threads, all of which execute $P$.

![Diagram](image-url)
A *trace* is a sequence $\tau = \langle \sigma_1 : i_1 \rangle \langle \sigma_2 : i_2 \rangle \cdots \in (\Sigma \times \mathbb{N})^\omega$.
A trace is a sequence $\tau = \langle \sigma_1 : i_1 \rangle \langle \sigma_2 : i_2 \rangle \cdots \in (\Sigma \times \mathbb{N})^\omega$

- Associate linear-time property $\Phi$ w/ set of traces $\mathcal{L}(\Phi)$ that satisfy it.
A trace is a sequence $\tau = \langle \sigma_1 : i_1 \rangle \langle \sigma_2 : i_2 \rangle \cdots \in (\Sigma \times \mathbb{N})^\omega$

- Associate linear-time property $\Phi$ w/ set of traces $\mathcal{L}(\Phi)$ that satisfy it.
- Associate $P(N)$ w/ set of traces $\mathcal{L}(P(N)) \subseteq (\Sigma \times \{1, \ldots, N\})^\omega$ corresponding to interleaved paths through the thread template.
A **trace** is a sequence \( \tau = \langle \sigma_1 : i_1 \rangle \langle \sigma_2 : i_2 \rangle \cdots \in (\Sigma \times \mathbb{N})^\omega \)

- Associate linear-time property \( \Phi \) w/ set of traces \( L(\Phi) \) that satisfy it.
- Associate \( P(N) \) w/ set of traces \( L(P(N)) \subseteq (\Sigma \times \{1, \ldots, N\})^\omega \) corresponding to interleaved paths through the thread template
- Program traces \( L(P) = \bigcup_{N} L(P(N)) \)
A trace is a sequence $\tau = \langle \sigma_1 : i_1 \rangle \langle \sigma_2 : i_2 \rangle \ldots \in (\Sigma \times \mathbb{N})^\omega$

- Associate linear-time property $\Phi$ w/ set of traces $\mathcal{L}(\Phi)$ that satisfy it.
- Associate $P(N)$ w/ set of traces $\mathcal{L}(P(N)) \subseteq (\Sigma \times \{1, \ldots, N\})^\omega$ 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**

No corresponding executions

**Feasible traces**

At least one corresponding execution
Infeasible traces

Feasible traces

Error traces
Infeasible traces

Error traces

Feasible 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

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?
\[ \langle m:=t++ : 1 \rangle \langle m:=t++ : 2 \rangle (\langle [m>s] : 2 \rangle \langle [m<=s] : 1 \rangle \langle s++ : 1 \rangle \langle m:=t++ : 1 \rangle)^\omega \]

Stem

Loop
\[
\begin{align*}
\langle m := t++ : 1 \rangle \langle m := t++ : 2 \rangle & \quad \langle [m > s] : 2 \rangle \langle [m <= s] : 1 \rangle \langle s++ : 1 \rangle \langle m := t++ : 1 \rangle^\omega \\
\text{Stem} & \quad \text{Loop}
\end{align*}
\]

\[
\begin{align*}
\{ \text{old}(s) &= s \} \\
\langle [m > s] : 2 \rangle \\
\{ \text{old}(s) &= s \land m(2) \geq \text{old}(s) \} \\
\langle [m <= s] : 1 \rangle \\
\{ \text{old}(s) &= s \land m(2) \geq \text{old}(s) \} \\
\langle s++ : 1 \rangle \\
\{ \text{old}(s) < s \land m(2) \geq \text{old}(s) \} \\
\langle m := t++ : 1 \rangle \\
\{ \text{old}(s) < s \land m(2) \geq \text{old}(s) \}
\end{align*}
\]

Variance proof

Ranking formula
\[
\langle m:=t++ : 1 \rangle \langle m:=t++ : 2 \rangle
\]

**Stem**

\[
\langle [m>s] : 2 \rangle \langle [m<=s] : 1 \rangle \langle s++ : 1 \rangle \langle m:=t++ : 1 \rangle
\]^\omega
\]

**Loop**

\[
\{ s = t \}
\]

\[
\langle m:=t++ : 1 \rangle
\]

\[
\{ true \}
\]

\[
\langle m:=t++ : 2 \rangle
\]

\[
\{ true \}
\]

\[
\langle [m>s] : 2 \rangle
\]

\[
\{ true \}
\]

\[
\langle [m<=s] : 1 \rangle
\]

\[
\{ true \}
\]

\[
\langle s++ : 1 \rangle
\]

\[
\{ true \}
\]

\[
\langle m:=t++ : 1 \rangle
\]

\[
\{ true \}
\]

\[
\langle m:=t++ : 1 \rangle
\]

\[
\{ true \}
\]

**Variance proof**

\[
\{ old(s) = s \}
\]

\[
\langle [m>s] : 2 \rangle
\]

\[
\{ old(s) = s \land m(2) \geq old(s) \}
\]

\[
\langle [m<=s] : 1 \rangle
\]

\[
\{ old(s) = s \land m(2) \geq old(s) \}
\]

\[
\langle s++ : 1 \rangle
\]

\[
\{ old(s) < s \land m(2) \geq old(s) \}
\]

\[
\langle m:=t++ : 1 \rangle
\]

\[
\{ old(s) < s \land m(2) \geq old(s) \}
\]

**Invariance proof**
\[
\begin{align*}
\{ \text{old}(s) = s \} & \quad \langle [m>s] : 2 \rangle & \quad \{ \text{old}(s) = s \} \\
\langle s++ : 1 \rangle & \quad \{ \text{old}(s) < s \} \\
\{ m(2) \geq \text{old}(s) \} & \quad \{ \varphi \} \\
\{ \varphi \} & \quad \{ \sigma : i \}
\end{align*}
\]
Sequencing

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

\{m(1) < t\}
m := t++ : 2
\{m(1) < m(2)\}
Sequencing

\[
\begin{align*}
\{s \leq t\} \\
& \quad m := t++ : 1 \\
& \quad \{m(1) < t\}
\end{align*}
\]

\[
\begin{align*}
\{m(1) < t\} \\
& \quad m := t++ : 2 \\
& \quad \{m(1) < m(2)\}
\end{align*}
\]

\[
\begin{align*}
\{s \leq t\} \\
& \quad m := t++ : 1; \\
& \quad m := t++ : 2 \\
& \quad \{m(1) < m(2)\}
\end{align*}
\]
Symmetry

\[ P(N) = P \parallel P \parallel \cdots \parallel P \]

\( N \text{ times} \)

\[
\{ s \leq m(1) \land m(1) < m(2) \}
\]

\[
[m \leq s] : 2
\]

\{ \textit{false} \}
Symmetry

\[ P(N) = P \parallel P \parallel \cdots \parallel P \]

\( N \) times

\[
\begin{align*}
\{ s \leq m(1) \land m(1) < m(2) \} \quad [m \leq s]: 2 \\
\{ false \}
\end{align*}
\]

\[
\begin{align*}
\{ s \leq m(2) \land m(2) < m(1) \} \quad [m \leq s]: 1 \\
\{ false \}
\end{align*}
\]
Symmetry

\[ P(N) = P \parallel P \parallel \cdots \parallel P \]
\[ \text{\textit{N times}} \]

\[
\begin{align*}
\{ s \leq m(1) \land m(1) < m(2) \} & \quad [m \leq s] : 2 \\
\{ false \} & \quad [2 \leq 3] \\
& \quad \{ s \leq m(2) \land m(2) < m(3) \} & \quad [m \leq s] : 3 \\
& \quad \{ false \}
\end{align*}
\]
Conjunction

\[
\begin{align*}
\{ m(1) < t \} \\
m := t++ : 3 \\
\{ m(1) < m(3) \}
\end{align*}
\]

\[
\begin{align*}
\{ m(2) < t \} \\
m := t++ : 3 \\
\{ m(2) < m(3) \}
\end{align*}
\]
Conjunction

\[
\{ m(1) < t \} \\
\text{m} := t++ : 3 \\
\{ m(1) < m(3) \}
\]

\[
\{ m(2) < t \} \\
\text{m} := t++ : 3 \\
\{ m(2) < m(3) \}
\]

\[
\{ m(1) < t \land m(2) < t \} \\
\text{m} := t++ : 3 \\
\{ m(1) < m(3) \land m(2) < m(3) \}
\]
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 $(H, R)$ proves a trace $\tau$ infeasible if there is some ranking formula $r \in R$, some decomposition of $\tau$:

$$
\tau = \tau_1 \tau_2 \tau_3 \cdots
$$

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

$${\text{pre}}\tau_1 \{\varphi_1\}$$

$${\text{pre}}\tau_1 \tau_2 \{\varphi_2\}$$

$$\vdots$$

$${\text{pre}}\tau_1 \tau_2 \cdots \tau_i \{\varphi_i\}$$

$${\varphi}_1 ^{\text{old}}(x) = x \} \tau_2 \{ r \}$$

$${\varphi}_2 ^{\text{old}}(x) = x \} \tau_3 \{ r \}$$

$$\vdots$$

$${\varphi}_i ^{\text{old}}(x) = x \} \tau_{i+1} \{ r \}$$

all belong to $H$. 

The set of traces $(H, R)$ proves infeasible is denoted $! (H, R)$. 
A WFPS \( \langle H, R \rangle \) proves a trace \( \tau \) infeasible if there is some ranking formula \( r \in R \), some decomposition of \( \tau \):

\[
\tau_1 \quad \tau_2 \quad \tau_3 \quad \cdots
\]

and some sequence of “intermediate formulas” \( \varphi_1, \varphi_2, \ldots \) such that

\[
\begin{align*}
\{ \text{pre} \} \tau_1 \{ \varphi_1 \} & \quad \{ \varphi_1 \land \text{old}(x) = x \} \tau_2 \{ r \} \\
\{ \text{pre} \} \tau_1 \tau_2 \{ \varphi_2 \} & \quad \{ \varphi_2 \land \text{old}(x) = x \} \tau_3 \{ r \} \\
\vdots & \\
\{ \text{pre} \} \tau_1 \tau_2 \cdots \tau_i \{ \varphi_i \} & \quad \{ \varphi_i \land \text{old}(x) = x \} \tau_{i+1} \{ r \}
\end{align*}
\]

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 → 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:

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

•
For any $N \in \mathbb{N}$, $(P \setminus (f_1; \cdots; f_n))$ is $!$-regular. Same for $(L)$ and $(H; R)$.

•
Fact: If $L_1$ and $L_2$ are $!$-regular, then $UP(L_1) \subseteq L_2$ implies $L_1 \subseteq L_2$. 
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 → finite traces

An ultimately periodic trace is a trace of the form $\pi\rho\rho\rho\ldots$

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 $N \in \mathbb{N}$, $\mathcal{L}(P) \cap (\Sigma \times \{1, \ldots, N\})^\omega$ is $\omega$-regular. Same for $\mathcal{L}(\Phi)$ and $\omega(H, R)$.
- Fact: If $L_1$ and $L_2$ are $\omega$-regular, then $UP(L_1) \subseteq L_2$ implies $L_1 \subseteq L_2$. 
Quantified Predicate Automata (QPA): a class of infinite-state automata that recognize words over an infinite alphabet.
Quantified Predicate Automata (QPA): a class of infinite-state automata that recognize words over an infinite alphabet.

- There is a QPA that recognizes \( L(P) \).
- There is a QPA that recognizes \( L(\phi) \).
- There is not a QPA that recognizes \( L(H; R) \).
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 $(\mathcal{L}(\Phi))$.
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 $\varphi$ and some ranking formula $r \in R$ such that

$$\{\operatorname{pre}\} \pi \{\varphi\} \quad \text{and} \quad \{\varphi \land \operatorname{old}(x) = x\} \rho \{r\}$$

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 $\varphi$ and some ranking formula $r \in R$ such that

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

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 $\varphi$ and some ranking formula $r \in R$ such that

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

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 $\varphi$ and some ranking formula $r \in R$ such that

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

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 $\mathcal{L}(P) \setminus \mathcal{L}(\Phi) \subseteq \omega(H, R)$, 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 $\{\varphi_{n,k} \land \text{old}(x) = x\} \rho^k \{r_{n,k}\}$. Ramsey!
But...

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

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

belong to $H$. Call this language $\left((H, R)\right)$. Membership of $\pi \rho$ in $\left((H, R)\right)$ does not imply that $\pi \rho^\omega \in \omega(H, R)$. It does not even imply that $\pi \rho^\omega$ is infeasible!

**Theorem**

If $\left(\mathcal{L}(P)\right) \setminus \left(\mathcal{L}(\Phi)\right) \subseteq \left((H, R)\right)$, 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!