Proof Spaces for Unbounded Parallelism

Zachary Kincaid
University of Toronto

January 16, 2015

Joint work with:
Azadeh Farzan, University of Toronto

Andreas Podelski, University of Freiburg

Multi-threaded program verification

- Unbounded/unknown number of threads
  - E.g., webservers, computations parallelized over N processors, ...

Multi-threaded program verification

- Unbounded/unknown number of threads
  - E.g., webservers, computations parallelized over N processors, ...
  - Single template T executed by every thread

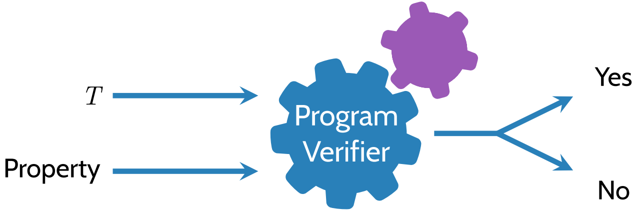
$$T^N = \underbrace{T \parallel T \parallel \cdots \parallel T}_{N \text{ times}}$$

Multi-threaded program verification

- Unbounded/unknown number of threads
  - E.g., webservers, computations parallelized over N processors, ...
  - Single template T executed by every thread

$$T^N = \underbrace{T \parallel T \parallel \cdots \parallel T}_{N \text{ times}}$$

- Goal: *prove* that a given program is free of (certain types of) errors.

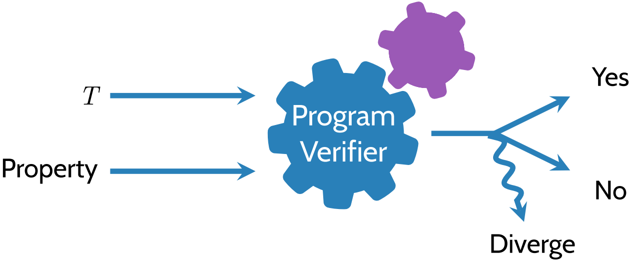


Multi-threaded program verification

- Unbounded/unknown number of threads
  - E.g., webservers, computations parallelized over N processors, ...
  - Single template T executed by every thread

$$T^N = \underbrace{T \parallel T \parallel \cdots \parallel T}_{N \text{ times}}$$

- Goal: *prove* that a given program is free of (certain types of) errors.



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

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

Proving correctness of a multi-threaded program is hard.

$$\forall i, j \in \text{Thread}.\mathtt{pc}(i) \neq \text{init} \land \mathtt{pc}(j) \neq \text{init} \land \mathtt{m}(i) = \mathtt{m}(j) \Rightarrow i = j$$

Proving correctness of a multi-threaded program is hard.

$$\forall i, j \in \text{Thread}.\texttt{pc}(i) \neq \text{init} \wedge \texttt{pc}(j) \neq \text{init} \wedge \texttt{m}(i) = \texttt{m}(j) \Rightarrow i = j$$

Proving correctness of a trace of a multi-threaded program is easy.

- Re-use sequential verification!

Proving correctness of a multi-threaded program is hard.

$$\forall i, j \in \text{Thread}.\mathtt{pc}(i) \neq \text{init} \wedge \mathtt{pc}(j) \neq \text{init} \wedge \mathtt{m}(i) = \mathtt{m}(j) \Rightarrow i = j$$

Proving correctness of a trace of a multi-threaded program is easy.

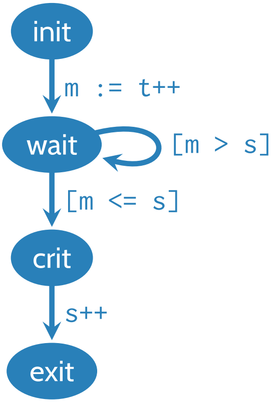
- Re-use sequential verification!

Program is correct $\iff$ *each of its traces are correct.*

Proof Spaces

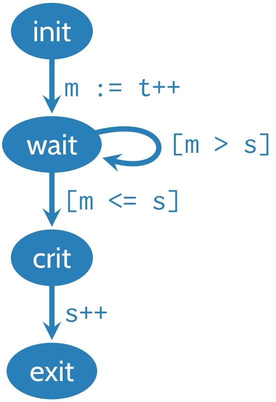
global `t` : int *// ticket counter*
global `s` : int *// service counter*
local `m` : int *// my ticket*
init `s` $\leq$ `t`

```
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` $\leq$ `t`

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




```
m := t++ : 1
m := t++ : 2
[m <= s] : 1
[m <= s] : 2
```

Commands

Error trace $\in (\Sigma \times \mathbb{N})^*$

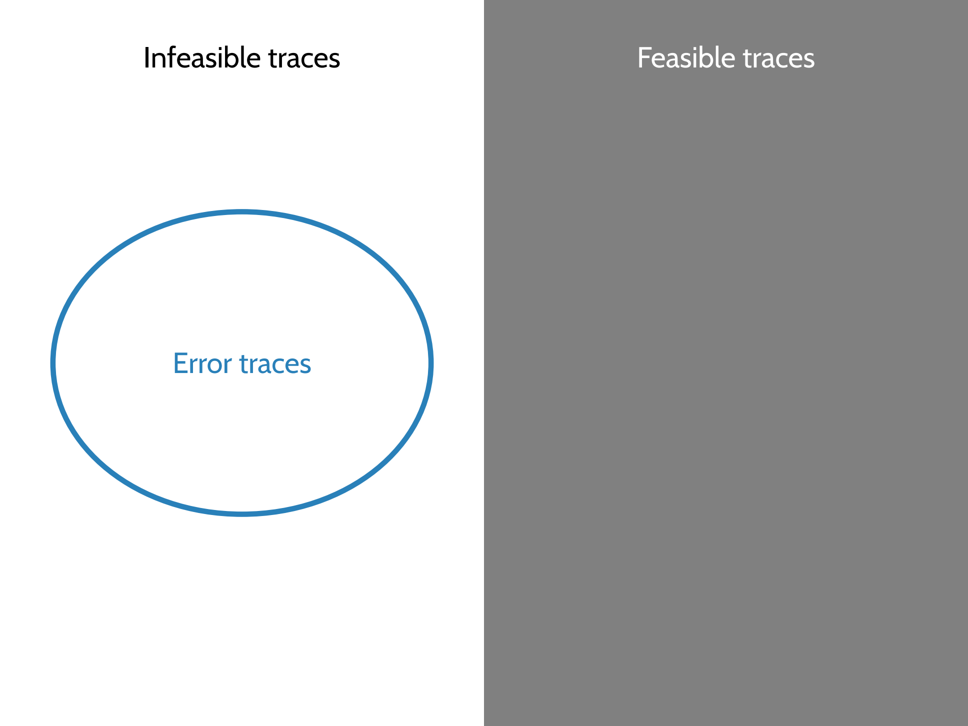
Thread IDs

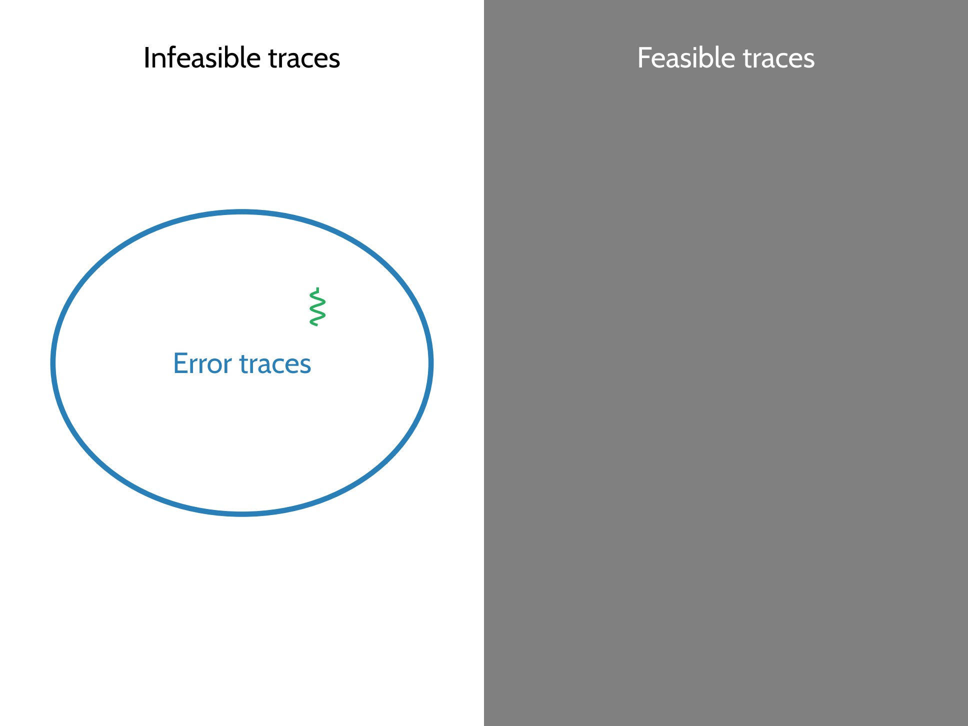
Infeasible traces

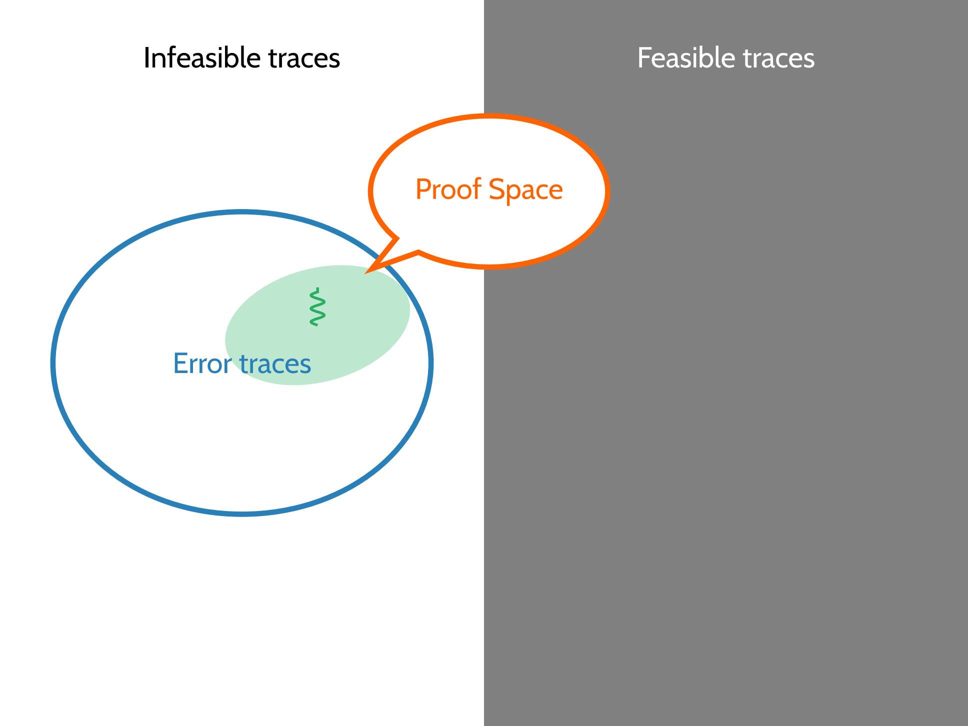
No corresponding executions

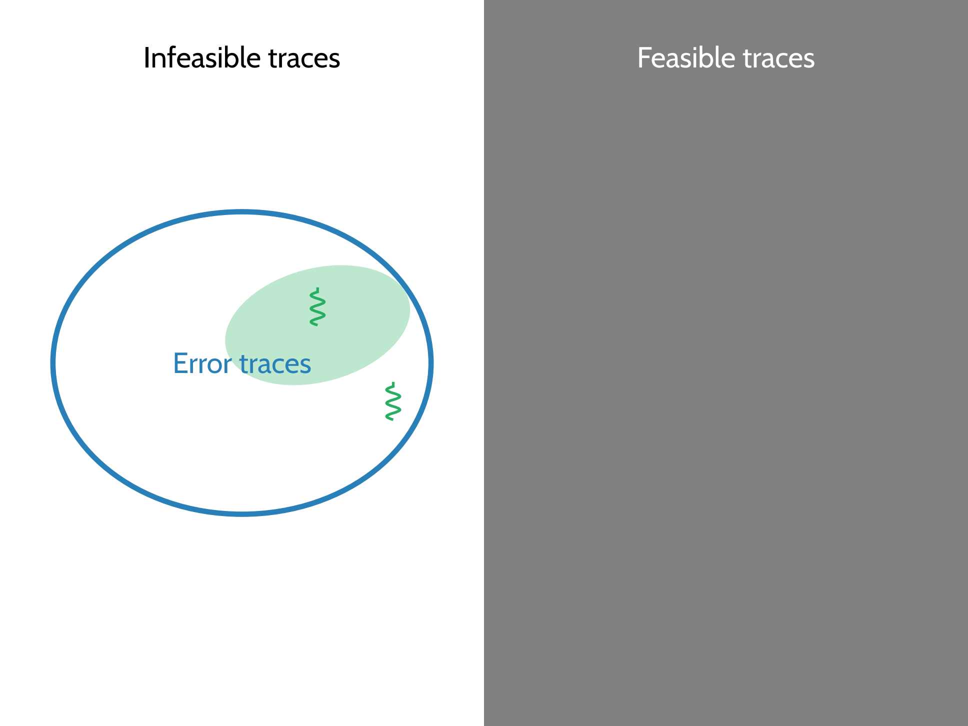
Feasible traces

At least one corresponding execution

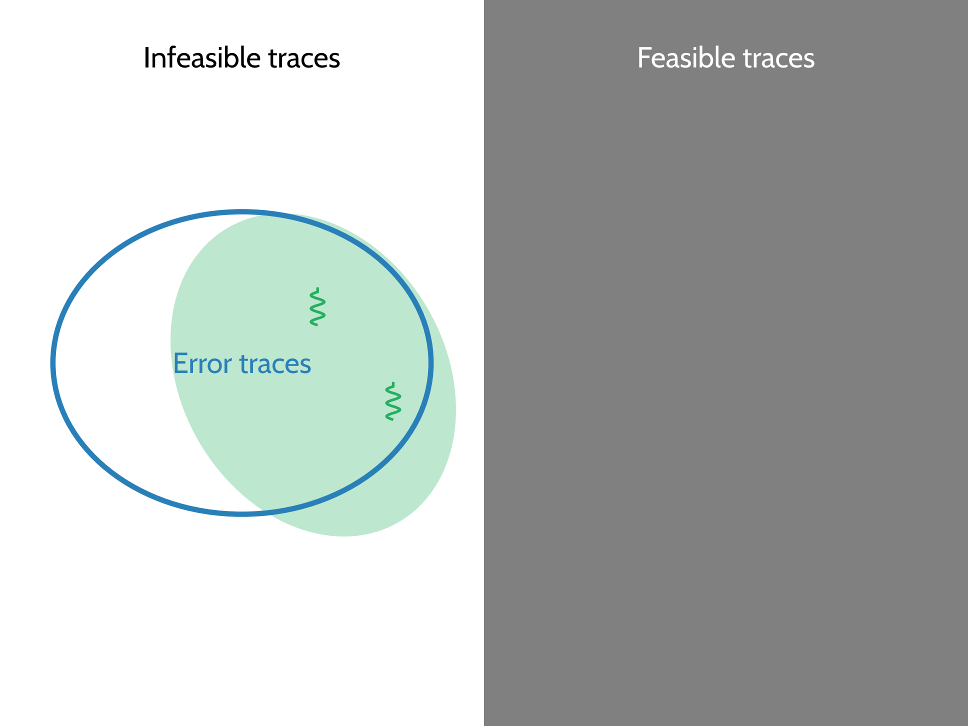

Infeasible traces
Feasible traces
Error traces

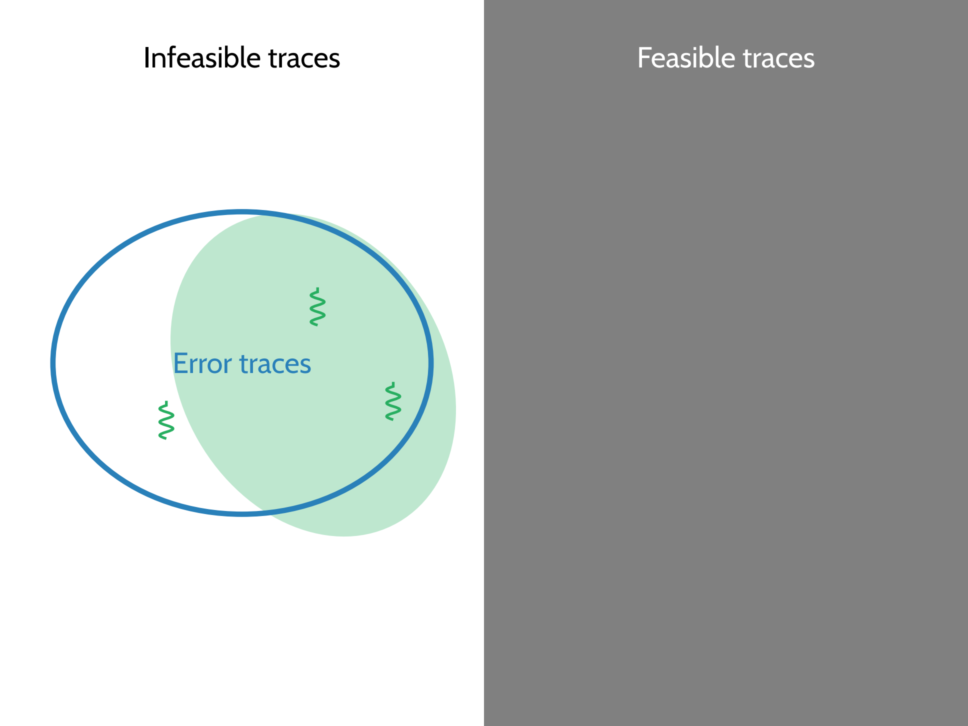

Infeasible traces
Feasible traces
Error traces

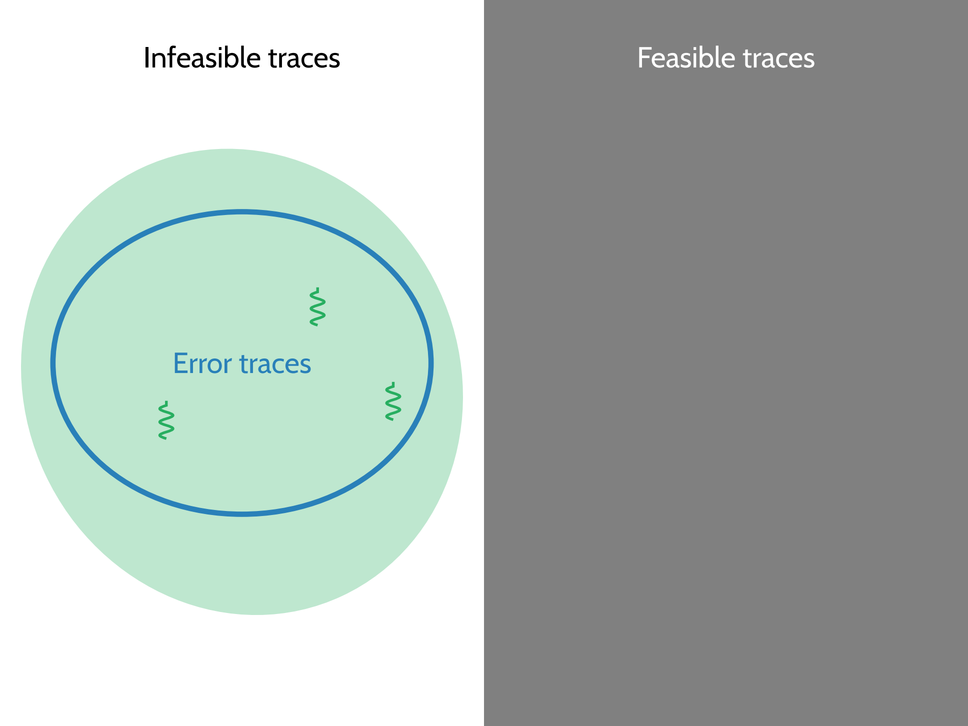

Infeasible traces
Feasible traces
Proof Space
Error traces

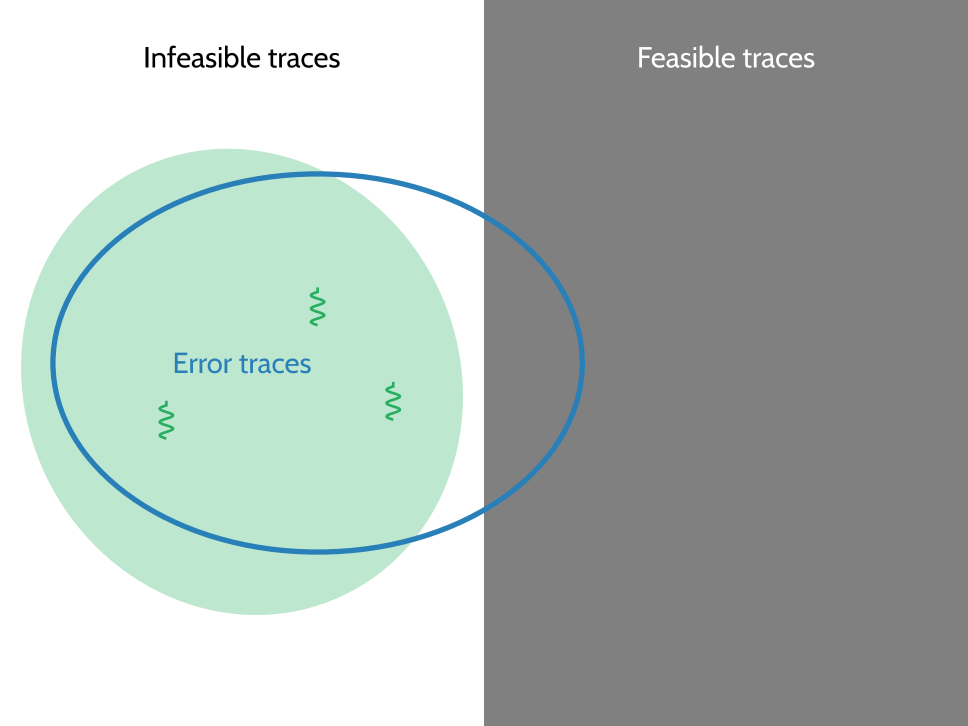


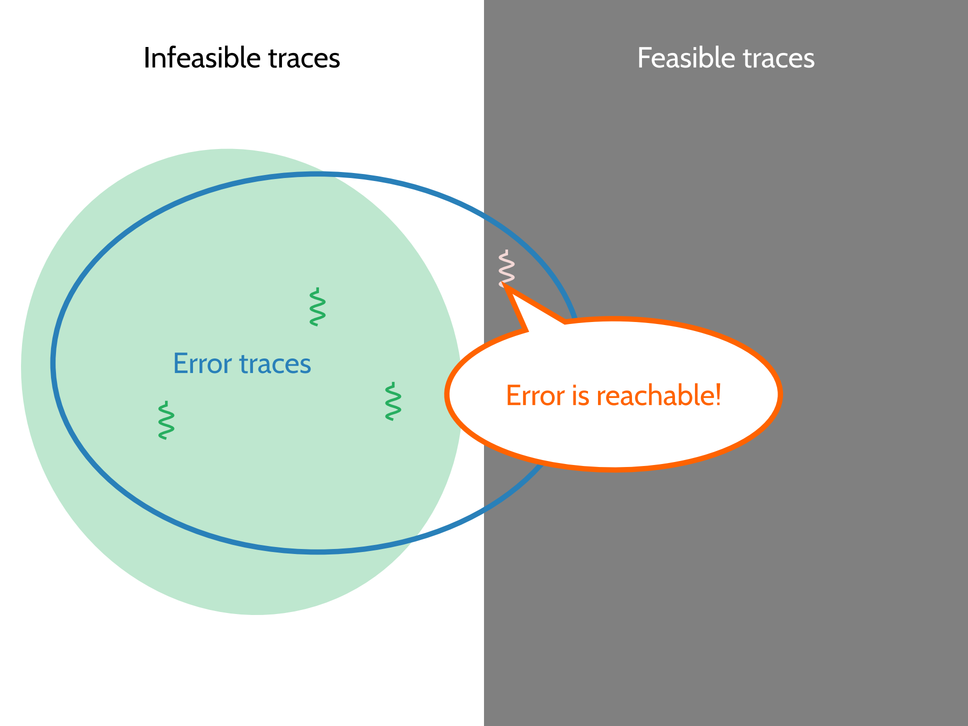
Infeasible traces
Feasible traces
Error traces

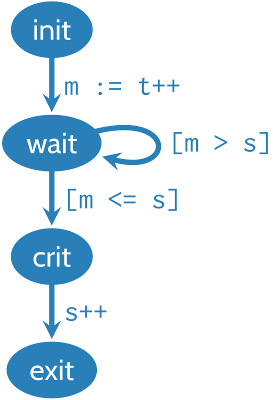

Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Error traces
Error is reachable!



$\{s \leq t\}$

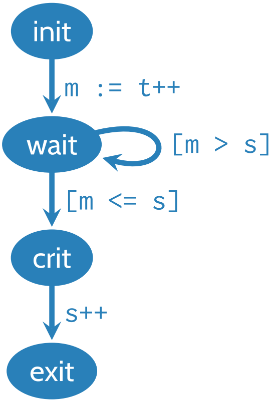
`m := t++` : 1

`m := t++` : 2

`[m <= s]` : 1

`[m <= s]` : 2

$\{\mathit{false}\}$



$$\{\mathtt{s} \leq \mathtt{t}\}$$

$$\mathtt{m\ :=\ t{+}{+}} : 1$$

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

$$\mathtt{m\ :=\ t{+}{+}} : 2$$

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

$$\mathtt{[m\ <=\ s]} : 1$$

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

$$\mathtt{[m\ <=\ s]} : 2$$

$$\{\mathit{false}\}$$

Intermediate assertions

Craig interpolation,[1]

Abstract post,[2]

Dual narrowing,[3]

...

[1] T A. Henzinger, R. Jhala, R. Majumdar, K. L. McMillan. Abstractions from proofs. POPL'04

[2] P. Cousot & R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. POPL'77.

[3] P. Cousot. Abstracting Induction by Extrapolation and Interpolation. VMCAI'15.

"Small theorems" from sequential verifiers

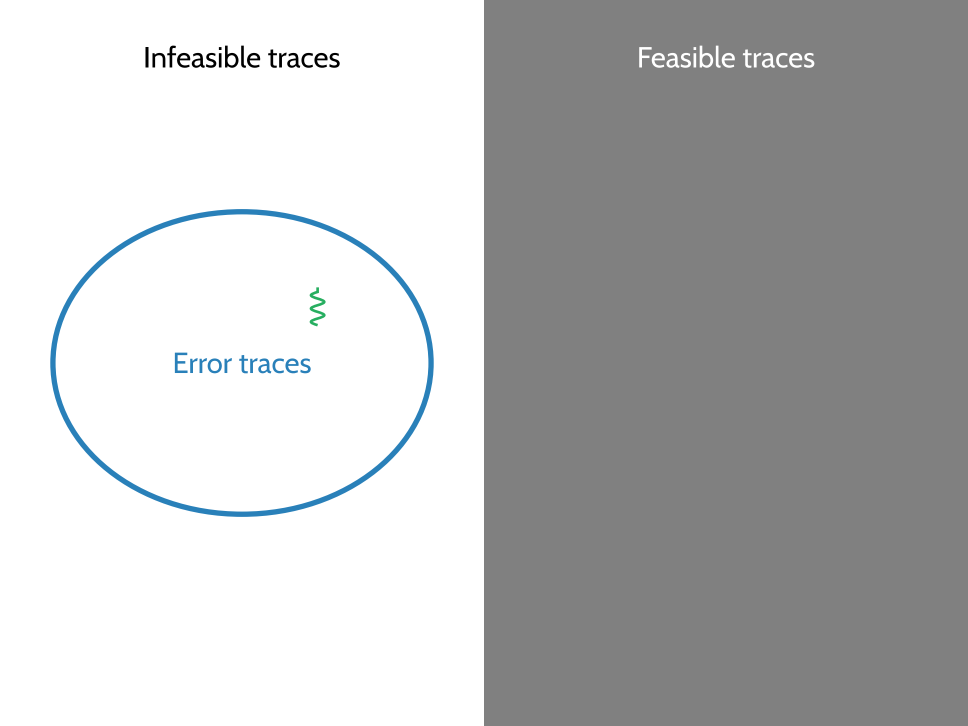
$\{s \le t\}$
m := t++ : 1
$\{s \le \mathtt{m}(1)\}$

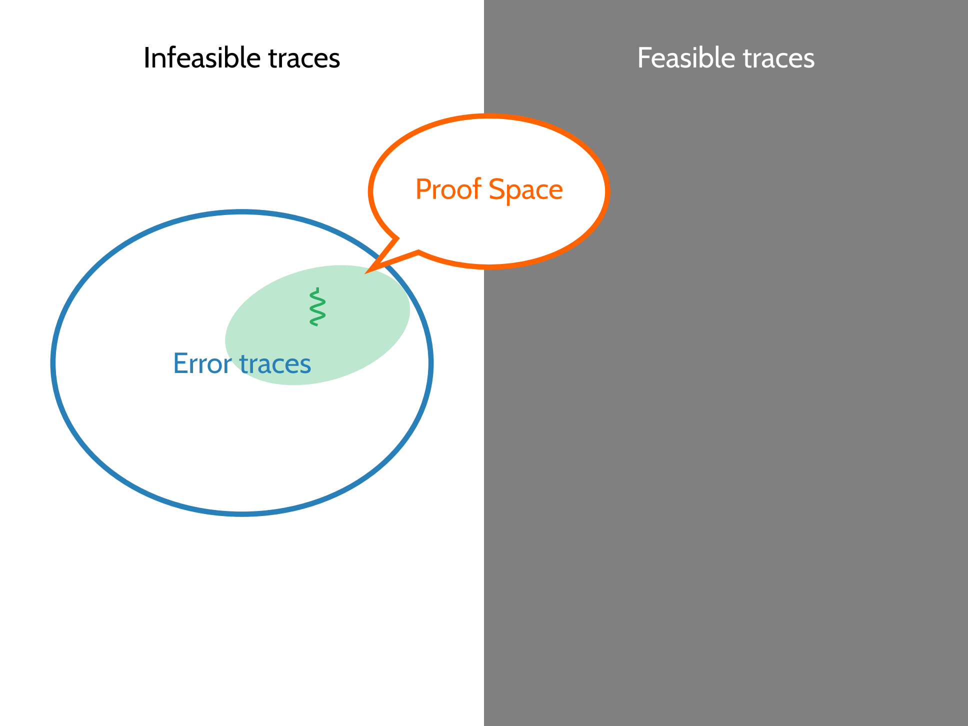
$\{\textit{true}\}$
m := t++ : 1
$\{\mathtt{m}(1) < t\}$

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

$\{s \le \mathtt{m}(1) \land \mathtt{m}(1) < \mathtt{m}(2)\}$
[m <= s] : 2
$\{\textit{false}\}$

$\{s \le \mathtt{m}(1) \land \mathtt{m}(1) < \mathtt{m}(2)\}$
s++ : 1
$\{s \le \mathtt{m}(2)\}$


Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Proof Space
Error traces

Sequencing

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

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

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

$$T^N = \underbrace{T \parallel T \parallel \cdots \parallel T}_{N \text{ times}}$$

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

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

$\{\textit{false}\}$

Symmetry

$$T^N = \underbrace{T \parallel T \parallel \cdots \parallel T}_{N \text{ times}}$$

$$\begin{array}{c} \{\mathsf{s} \leq \mathtt{m}(1) \wedge \mathtt{m}(1) < \mathtt{m}(2)\} \\ \texttt{[m <= s]} : \mathbf{2} \\ \{\mathit{false}\} \end{array} \xrightarrow[{[2 \mapsto 1]}]{[1 \mapsto 2]} \begin{array}{c} \{\mathsf{s} \leq \mathtt{m}(2) \wedge \mathtt{m}(2) < \mathtt{m}(1)\} \\ \texttt{[m <= s]} : \mathbf{1} \\ \{\mathit{false}\} \end{array}$$

Symmetry

$$T^N = \underbrace{T \parallel T \parallel \cdots \parallel T}_{N \text{ times}}$$

$$\begin{array}{c} \{\mathsf{s} \leq \mathtt{m}(1) \wedge \mathtt{m}(1) < \mathtt{m}(2)\} \\ \texttt{[m <= s]} : \mathbf{2} \\ \{\textit{false}\} \end{array} \xrightarrow[{[2 \mapsto 3]}]{[1 \mapsto 2]} \begin{array}{c} \{\mathsf{s} \leq \mathtt{m}(2) \wedge \mathtt{m}(2) < \mathtt{m}(3)\} \\ \texttt{[m <= s]} : \mathbf{3} \\ \{\textit{false}\} \end{array}$$

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

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

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

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

A *Proof space* is a set of valid Hoare triples which is closed under sequencing, symmetry, and conjunction.

A *Proof space* is a set of valid Hoare triples which is closed under sequencing, symmetry, and conjunction.

- *Finitely generated*: there is a finite "basis" which generates the space

A *Proof space* is a set of valid Hoare triples which is closed under sequencing, symmetry, and conjunction.

- *Finitely generated*: there is a finite "basis" which generates the space

Proof rule: if there exists a proof space H such that for all error traces τ

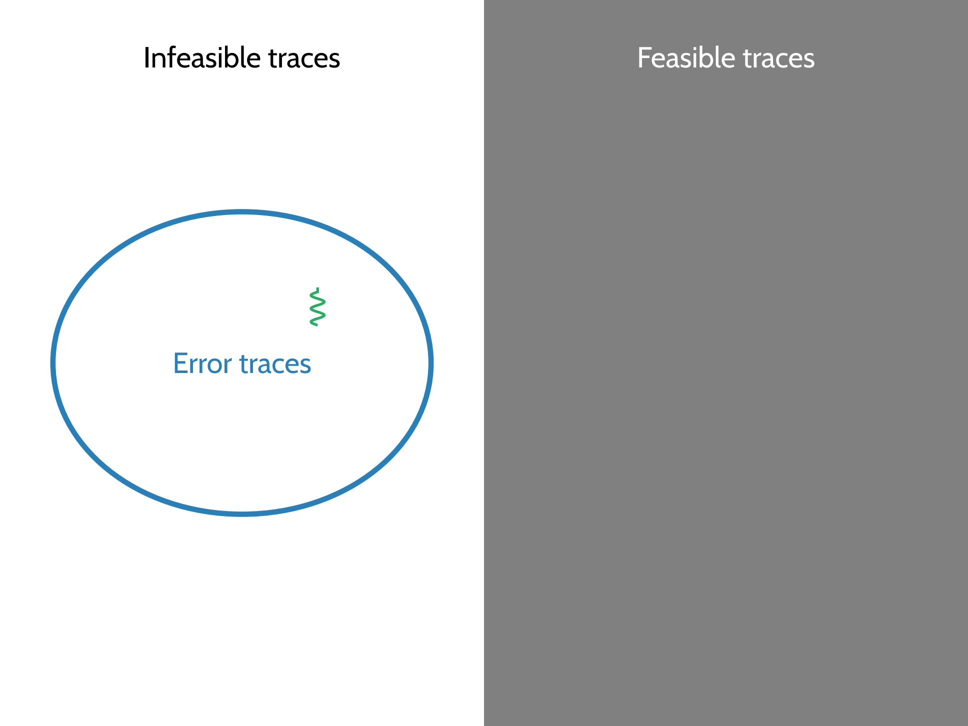
$$\{\text{pre}\}\tau\{\textit{false}\} \in H,$$

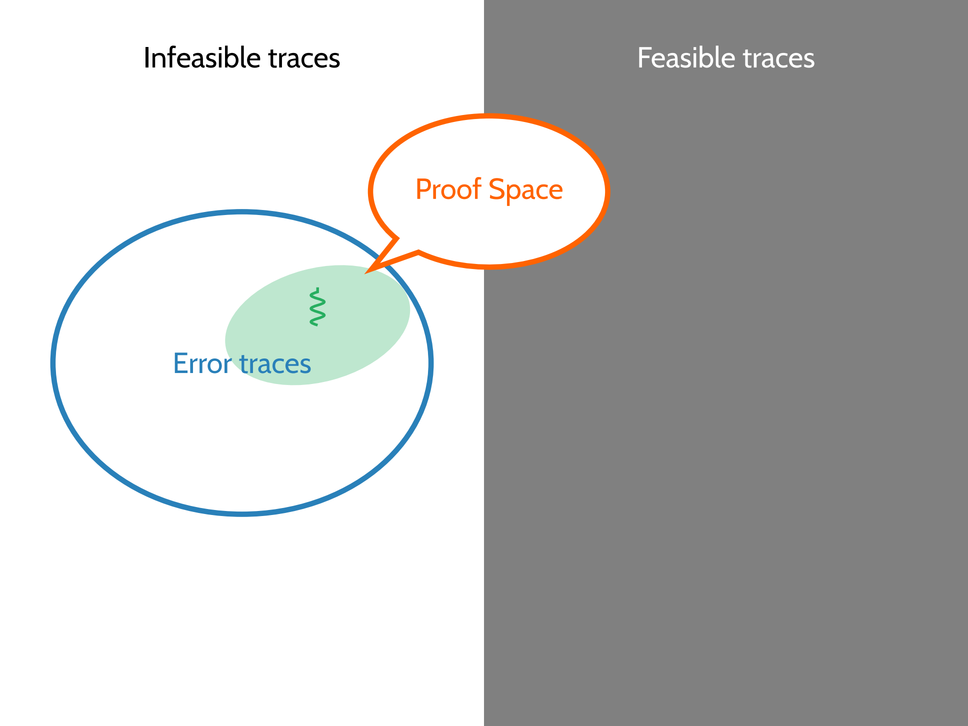
then the program is correct.

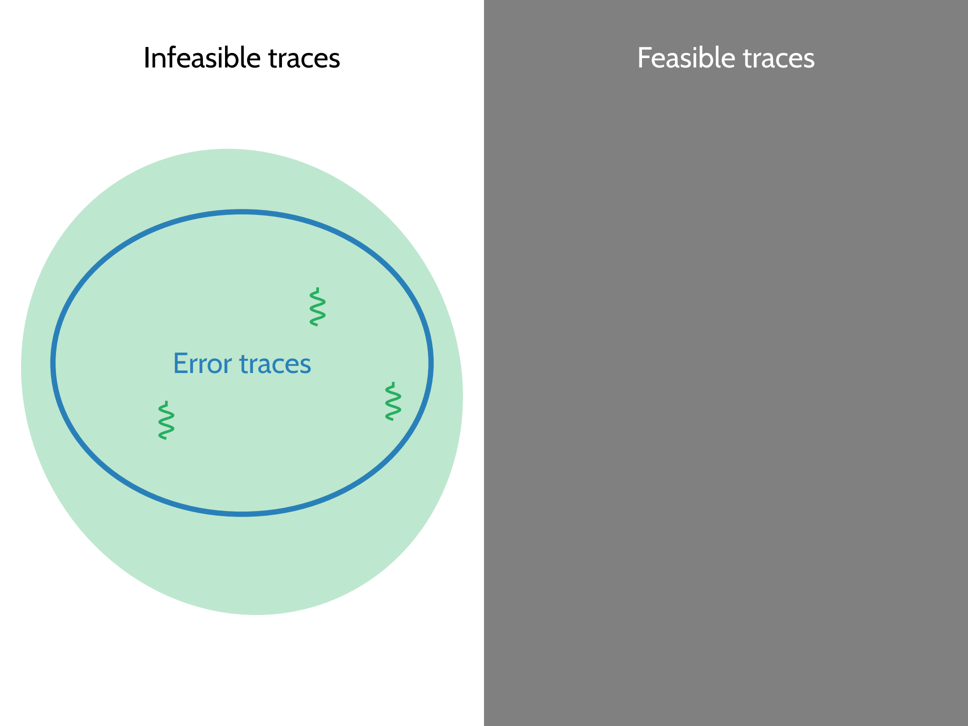
Relative completeness

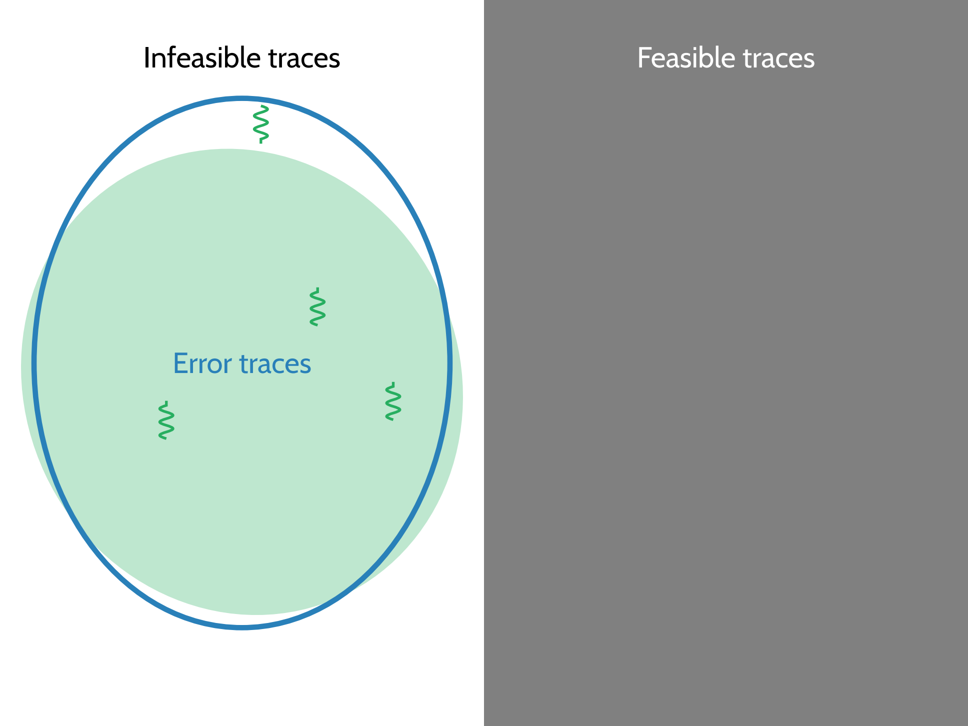
Theorem

Every inductive invariant (with control variables & universal thread quantification) corresponds to a proof space.


Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Proof Space
Error traces


Infeasible traces
Feasible traces
Error traces


Infeasible traces
Feasible traces
Error traces

Predicate Automata

Predicate automata (PA)

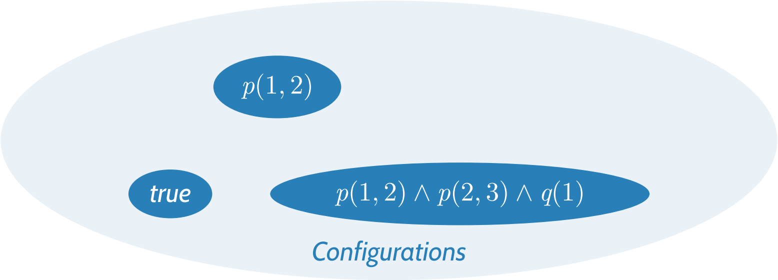
Vocabulary (Q, ar) is a finite relational first-order vocabulary

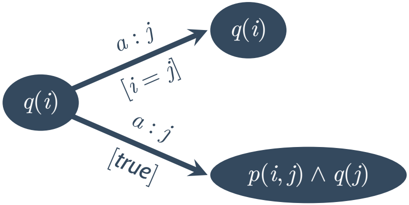
$$Q = \{p, q\}, ar(p) = 2, ar(q) = 1$$

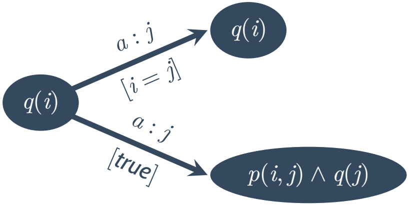
Predicate automata (PA)

Vocabulary (Q, ar) is a finite relational first-order vocabulary

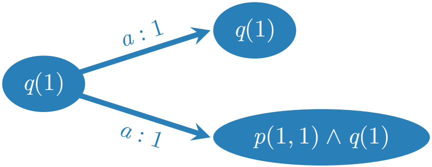
$$Q = \{p, q\}, ar(p) = 2, ar(q) = 1$$



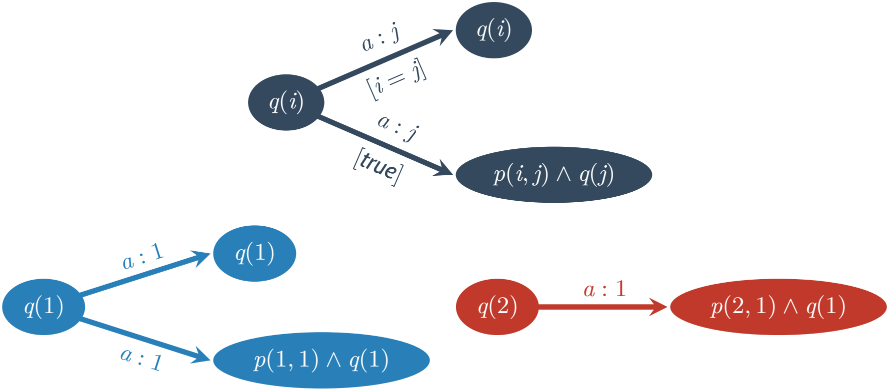

$q(i)$
$a : j$
$[i = j]$
$q(i)$
$a : j$
[true]
$p(i, j) \wedge q(j)$

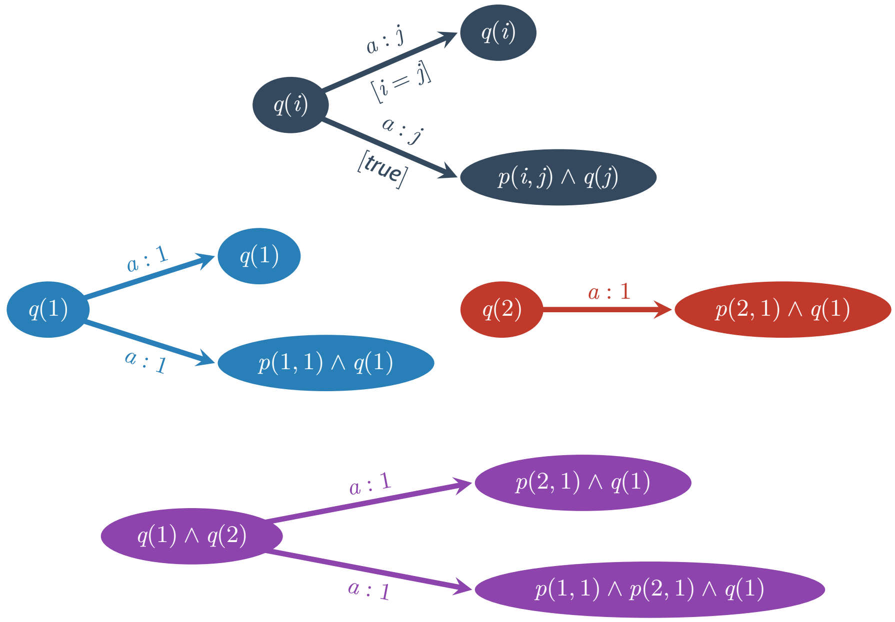


$q(i)$
$a : j$
$[i = j]$
$q(i)$
$a : j$
[true]
$p(i, j) \wedge q(j)$



$q(1)$
$a : 1$
$q(1)$
$a : 1$
$p(1, 1) \wedge q(1)$


$q(i)$
$a : j$
$[i = j]$
$q(i)$
$a : j$
[true]
$p(i, j) \wedge q(j)$
$q(1)$
$a : 1$
$q(1)$
$a : 1$
$p(1, 1) \wedge q(1)$
$q(2)$
$a : 1$
$p(2, 1) \wedge q(1)$



$q(i)$
$a : j$
$[i = j]$
$q(i)$
$a : j$
[true]
$p(i, j) \wedge q(j)$
$q(1)$
$a : 1$
$q(1)$
$a : 1$
$p(1, 1) \wedge q(1)$
$q(2)$
$a : 1$
$p(2, 1) \wedge q(1)$
$q(1) \wedge q(2)$
$a : 1$
$p(2, 1) \wedge q(1)$
$a : 1$
$p(1, 1) \wedge p(2, 1) \wedge q(1)$

Proof checking

- For any H, $\{\tau : \{\text{pre}\}\tau\{\textit{false}\} \in H\}$ is recognized by a PA $A(H)$

Proof checking

- For any H, $\{\tau : \{\text{pre}\}\tau\{\textit{false}\} \in H\}$ is recognized by a PA $A(H)$
- For any program, set of error traces is recognized by a PA *Err*

Proof checking

- For any H, $\{\tau : \{\text{pre}\}\tau\{\textit{false}\} \in H\}$ is recognized by a PA $A(H)$
- For any program, set of error traces is recognized by a PA *Err*
- PA languages are closed under intersection and complement

Proof checking

- For any H, $\{\tau : \{\text{pre}\}\tau\{\textit{false}\} \in H\}$ is recognized by a PA $A(H)$
- For any program, set of error traces is recognized by a PA *Err*
- PA languages are closed under intersection and complement

Proof space inclusion reduces to PA emptiness

$$\forall \tau \in \text{Error trace}.\{\text{pre}\}\tau\{\textit{false}\} \in H$$
$$\iff$$
$$\textit{Err} \cap \overline{A(H)} = \emptyset$$

Theorem

The emptiness problem for predicate automata is undecidable.

Theorem

The emptiness problem for predicate automata is undecidable.

Theorem

The emptiness problem for monadic predicate automata ($\forall q \in Q, ar(q) \leq 1$) is decidable.

Proof spaces: a theoretical foundation for verifying multi-threaded programs

- Prove traces, not programs
  - Sample - generalize - check loop

Proof spaces: a theoretical foundation for verifying multi-threaded programs

- Prove traces, not programs
  - Sample - generalize - check loop
- Proof generalization via a simple deductive system
  - Complete relative to inductive invariants

Proof spaces: a theoretical foundation for verifying multi-threaded programs

- Prove traces, not programs
  - Sample - generalize - check loop
- Proof generalization via a simple deductive system
  - Complete relative to inductive invariants
- Reduce "proof checking" to an automata-theoretic problem
  - Interesting decidable sub-problem