Inferring Likely Deterministic Specifications for Multithreaded Programs

Jacob Burnim
Koushik Sen
Parallel Computing Laboratory
University of California, Berkeley
Specification Inference

- Inferring **temporal** specs (protocols):
  - *Ammons, et al*, POPL 02
  - *Whaley, et al*, ISSTA 02
  - *Livshits, Zimmermann*, FSE 05
  - Perrecotta: *Yang, et al*, ICSE 06
  - JADET: *Wasylkowski, et al*, FSE 07
  - *Acharya, et al*, FSE 07
  - JAVERT: *Gabel, Su*, ICSE 08
  - many others …
Speciﬁcation Inference

• Inferring likely *invariants*:

\[ 0 \leq i < N \quad \text{sum} = \sum_{j=0}^{i-1} A[j] \land i = N + 1 \]

• DAIKON: *Ernst, et al., ICSE 00*
• DIDUCE: *Hangal, Lam, ICSE 02*
• DySy: *Csallner, et al., ICSE 08*
• Gulwani, *et al., VMCAI 09*
• many others …
Parallel Programming is Hard

- **Key Culprit:** Nondeterminism.
  - Interleaving of parallel threads

- **Determinism** key to parallel correctness.
  - Same input ==> semantically same output.
  - Parallelism is wrong if some schedules give a correct answer while others don’t.

- **Previously:** Help programmers make their parallel code deterministic.
  - Assertion framework to specify determinism.
Advantages of Deterministic Specs


- Lightweight spec of parallel correctness
  - Independent of functional specification
  - Decompose correctness efforts
- Useful for documentation
- Can effectively test deterministic specs
  - Combine with testing tools to distinguish harmful from benign data races, etc.
Advantages of Deterministic Specs


**Goal:** Automatically infer deterministic specifications by observing sample program runs.

- Useful for documentation
- Can effectively test deterministic specs
  - Combine with testing tools to distinguish harmful from benign data races, etc.
Advantages of Deterministic Specs


**Goal**: Automatically infer deterministic specifications by observing sample program runs.

*Result*: Recover our previous manual specifications for most benchmarks.
Outline

- Motivation and Overview
- **Background: Deterministic Specs**
  - Specification Inference Problem
  - Inferring Deterministic Specifications
- Experimental Evaluation
- Related Work
- Conclusions
Background: Deterministic Specs

// Parallel fractal render.
mandelbrot(params, img);

- Want to assert parallel correctness.
Background: Deterministic Specs

// Parallel fractal render.
mandelbrot(params, img);

- Want to assert parallel correctness.

x=0.7
y=0.3
...
y=5.0
Background: Deterministic Specs

// Parallel fractal render.
mandelbrot(params, img);

- Want to assert parallel correctness.

x = 0.7
y = 0.3
... 
γ = 5.0
Background: Deterministic Specs

- Specifies that any two runs on the **same input parameters** must yield the **same output image**.

```c

deterministic
assume (params == params') {
    // Parallel fractal render.
    mandelbrot(params, img);
} assert (img == img');
```
deterministic
assume (params == params') {
   // Parallel fractal render.
   mandelbrot(params, img);
} assert (img == img');

\[ s_0: \ldots \text{params} \ldots \]

\[ s_0': \ldots \text{params}' \ldots \]
Background: Deterministic Specs

deterministic
assume (params == params') {
    // Parallel fractal render.
    mandelbrot(params, img);
} assert (img == img');
deterministic
assume (params == params') {
  // Parallel fractal render.
  mandelbrot(params, img);
} assert (img == img');
Background: Deterministic Specs

deterministic
assume (params == params') {
  // Parallel fractal render.
  mandelbrot(params, img);
} assert (img == img');

\[ s_0 : \ldots \text{params} \ldots \]  \[ s_1 : \ldots \text{img} \ldots \]
\[ s_0' : \ldots \text{params}' \ldots \]  \[ s_1' : \ldots \text{img}' \ldots \]
Background: Deterministic Specs

Formally, specifies:

\[ \forall s_0 \xrightarrow{P} s_1, \ s_0' \xrightarrow{P} s_1' : \]

\[ \text{Pre}(s_0, s_0') \implies \text{Post}(s_1, s_1') \]
Background: Deterministic Specs

deterministic
assume (params == params') {
  // Parallel fractal render.
  mandelbrot(params, img);
} assert (img == img');
Background: Deterministic Specs

deterministic
assume (params == params') {
    // Parallel fractal render.
    mandelbrot(params, img);
} assert (img == img');
Background: Deterministic Specs

// Parallel fractal render.
mandelbrot (params, img);

- Much simpler than functional correctness:

\[
\forall_{0 \leq x < \text{width}} \cdot \forall_{0 \leq y < \text{height}} \cdot \\
\left( \left| f_{\text{iter}}^{\text{maxiter}} (0) \right| < 2 \land \img[x][y] = 0 \right) \\
\lor \exists_{1 \leq i < \text{maxiter}} \cdot \left| f_{\text{iter}}^i (0) \right| \geq 2 \land \forall_{1 \leq j < i} \cdot \left| f_{\text{iter}}^j (0) \right| < 2 \\
\land \img[x][y] = \text{HSB}\left(\left(i/\text{maxiter}\right)^', 1, 1\right)
\]

where \( f_{\text{iter}} (c) = c^2 + \left( x_{\text{center}} + (x_{\text{off}} + x)/\text{res} \right) \\
+ i \left( y_{\text{center}} + (y_{\text{off}} - y)/\text{res} \right) \)
set t = new RedBlackTreeSet();
...
deterministic
assume (t.equals(t')) {
    t.add(3) || t.add(5);
} assert (t.equals(t'));
Background: Deterministic Specs

double A[][[]], b[], x[];
...
deterministic
assume (A == A’ and b == b’) {
   // Solve A*x = b in parallel
   lufact_solve(A, b, x);
} assert (|x - x’| < ε);
Determinism is user-specified.

deterministic
assume (data == data') {
  // Parallel branch-and-bound
  Tree t = min_phylo_tree(data);
} assert (t.cost == t'.cost());
Background: Deterministic Specs

- Can effectively test deterministic specs.
  - Added assertions to 13 benchmarks.
  - Ran CalFuzzer to test if concurrency issues (data races, atomicity violations, etc.) could lead to violations of deterministic spec.
- Specification inference would help automate the above testing.
- Also aid program understanding.
Outline

- Motivation and Overview
- Background: Deterministic Specs
- Specification Inference Problem
- Inferring Deterministic Specifications
- Experimental Evaluation
- Related Work
- Conclusions
// Parallel branch-and-bound
(tree, cost) =
min_phylo_tree(N, data);
 Speciation Inference Problem

```plaintext
deterministic assume (???) {  
   // Parallel branch-and-bound  
   (tree, cost) =  
            min_phylo_tree(N, data);  
} assert (???)
```
Observation: Deterministic pre- and postcondition have simple structure.
• Conjunction of equality bridge predicates:

\[ N = N' \quad \text{tree.equals(tree')} \quad |cost - cost'| < \varepsilon \]
Four possible deterministic preconditions:

- `true`
- `data = data'`
- `N = N'`
- `data = data' ∧ N = N'`
Six possible deterministic postconditions:

- `true`  
  \[
  \left| \text{cost} - \text{cost}' \right| < \varepsilon \]
  \[
  \text{cost} = \text{cost}'
  \]

- `tree.equals(tree')`  
  \[
  \text{tree.equals}(\text{tree'})
  \]
  \[
  \land \left| \text{cost} - \text{cost}' \right| < \varepsilon
  \]
  \[
  \land \text{cost} = \text{cost}'
  \]
Six possible deterministic postconditions:

deterministic assume (???) {
    // Parallel branch-and-bound
    (tree, cost) =
        min_phylo_tree(N, data);
} assert (???)

Which specification should we choose?

- true
- tree.equals(tree')
- tree.equals(tree') \land |cost - cost'| < \varepsilon
- tree.equals(tree') \land cost = cost'
Principles for specification inference:
• Must be consistent with observed runs.

\[ \text{data} = \text{data}' \xrightarrow{\text{min}_\text{phylo}_\text{tree}} \text{tree}.\text{equals}(\text{tree}') \]
Specification Inference Problem

- Principles for specification inference:
  - Must be consistent with observed runs.
  - Postcondition as strong as possible.

\[
data = data' \xrightarrow{\text{min\_phylo\_tree}} |cost - cost'| < \varepsilon\]

\[
data = data' \xrightarrow{\text{min\_phylo\_tree}} \text{true}\]

Specification Inference Problem

• Principles for specification inference:
  • Must be consistent with observed runs.
  • Postcondition as strong as possible.
  • Precondition as weak as possible, for post

\[
data = data' \xrightarrow{\text{min\_phylo\_tree}} |cost - cost'| < \varepsilon
\]

\[
data = data' \wedge N = N' \xrightarrow{\text{min\_phylo\_tree}} |cost - cost'| < \varepsilon
\]
Principles for specification inference:

- Must be consistent with observed runs.
- Postcondition as strong as possible.
- Precondition as weak as possible, for post.

How do we compute such a deterministic specification?
Outline

- Motivation and Overview
- Background: Deterministic Specs
- Specification Inference Problem
- Inferring Deterministic Specifications
- Experimental Evaluation
- Related Work
- Conclusions
M variables => \( \Omega(2^M) \) specifications
- Exhaustive search infeasible.

- Two-step algorithm:
  - Compute strongest consistent postcondition.
  - Compute weakest consistent precondition for the inferred postcondition.
Inferring Deterministic Specs

\[ \text{min\_phylo\_tree} \]

\[ N = N' \]
\[ \text{data} = \text{data}' \]
\[ \text{tree.equals}(\text{tree}') \]
\[ |\text{cost} - \text{cost}'| < \varepsilon \]
\[ \text{cost} = \text{cost}' \]
Inferring Deterministic Specs

\[ N = N' \quad \text{data} = \text{data}' \]

\[ \downarrow: \quad \text{data} = \text{data}' \quad \land N = N' \]

\[ \text{true} \]

\[ \text{true} \]

\[ \text{true} \]

\[ \text{true} \]

\[ \text{true} \]

\[ \downarrow: \quad \text{tree}.equals(\text{tree}') \land \left| \text{cost} - \text{cost}' \right| < \varepsilon \]

\[ \downarrow: \quad \text{tree}.equals(\text{tree}') \land \text{cost} = \text{cost}' \]
Inferring Strongest Post

- For every pair of observed executions:
  - If they satisfy precondition \( \perp \), ensure that the postcondition holds.

\[
\begin{align*}
N &= N' \\
data &= data' \\
\perp &:\quad data = data' \\
\quad \land N = N'
\end{align*}
\]

\[
\begin{align*}
true \\
tree.equals(tree') \\
\vert cost - cost' \vert < \varepsilon \\
\perp &:\quad tree.equals(tree') \\
\quad \land \vert cost - cost' \vert < \varepsilon
\end{align*}
\]

\[
\begin{align*}
true \\
\quad \land cost = cost'
\end{align*}
\]
Inferring Strongest Post I

\[ N = 10, \ data = D1 \quad \rightarrow \quad \text{cost} = 3.7, \ tree = T1 \]

\[ N = 10, \ data = D1 \quad \rightarrow \quad \text{cost} = 3.7, \ tree = T2 \]
Inferring Strongest Post I

\[ N = 10, \text{ data} = D1 \]  \rightarrow  \[ \text{cost} = 3.7, \text{ tree} = T1 \]

\[ N = 10, \text{ data} = D1 \]  \rightarrow  \[ \text{cost} = 3.7, \text{ tree} = T2 \]

true

\[ N = N' \]  \rightarrow  \[ \text{data} = \text{data}' \]

⊥: \[ \text{data} = \text{data}' \]  \land  \[ N = N' \]

true

\[ \text{tree.equals}(\text{tree}') \]  \land  \[ |\text{cost} - \text{cost}'| < \varepsilon \]

\[ \text{tree.equals}(\text{tree}') \]  \land  \[ |\text{cost} - \text{cost}'| < \varepsilon \]  \land  \[ \text{cost} = \text{cost}' \]

⊥: \[ \text{tree.equals}(\text{tree}') \]  \land  \[ \text{cost} = \text{cost}' \]
Inferring Strongest Post I

\[ N=10, \ data=D1 \rightarrow \ cost=3.7, \ tree=T1 \]
\[ N=10, \ data=D1 \rightarrow \ cost=3.7, \ tree=T2 \]

```
true

N = N'
data = data'

⊥: data = data'
\∧ N = N'
```

```
true

tree.equals(tree')

|cost - cost'| < \(\varepsilon\)


⊥:

\begin{align*}
tree.equals(tree') \\
\land |cost - cost'| < \varepsilon
\end{align*}

\begin{align*}
tree.equals(tree') \\
\land cost = cost'
\end{align*}
```
Inferring Strongest Post I

\[ N = N' \]
\[ data = data' \]
\[ \perp : \quad data = data' \land N = N' \]

\[ true \]
\[ tree.equals(tree') \]
\[ |cost - cost'| < \epsilon \]
\[ cost = cost' \]
\[ \perp : \quad tree.equals(tree') \land cost = cost' \]
Inferring Strongest Post II

\[ N=10, \ data=D1 \rightarrow \ cost=3.7, \ tree=T1 \]
\[ N=10, \ data=D2 \rightarrow \ cost=2.4, \ tree=T3 \]

\[
\begin{align*}
\text{true} \quad &\quad \text{true} \\
N = N' \quad &\quad \text{data} = \text{data'} \\
\bot: \quad &\quad \text{data} = \text{data'} \\
&\quad \land N = N' \\
\end{align*}
\]

\[
\begin{align*}
\text{true} \quad &\quad \text{true} \\
tree\.equals(tree') \quad &\quad |cost - cost'| < \varepsilon \\
\bot: \quad &\quad tree\.equals(tree') \\
&\quad \land |cost - cost'| < \varepsilon \\
\end{align*}
\]

\[
\begin{align*}
\text{true} \quad &\quad \text{true} \\
\text{true} \quad &\quad \text{true} \\
\bot: \quad &\quad tree\.equals(tree') \\
&\quad \land cost = cost' \\
\end{align*}
\]
Inferring Strongest Post II

\[ N = 10, \quad \text{data} = D_1 \quad \text{cost} = 3.7, \quad \text{tree} = T_1 \]
\[ N = 10, \quad \text{data} = D_2 \quad \text{cost} = 2.4, \quad \text{tree} = T_3 \]

\[ \begin{align*}
\text{true} & \quad \text{true} \\
N = N' & \quad \text{data} = \text{data}' \\
\downarrow: & \quad \text{data} = \text{data}' \quad \land N = N'
\end{align*} \]

\[ \begin{align*}
\text{true} & \quad \text{true} \\
\text{tree.equals(tree')} & \quad \lvert \text{cost} - \text{cost}' \rvert < \varepsilon \\
\downarrow: & \quad \text{tree.equals(tree')} \quad \land \lvert \text{cost} - \text{cost}' \rvert < \varepsilon \\
\downarrow: & \quad \text{cost} = \text{cost}' \quad \land \text{tree.equals(tree')} \quad \land \text{data} = \text{data}' \quad \land N = N'
\end{align*} \]
Inferring Strongest Post II

\[
\begin{align*}
N &= N' \\
data &= data' \\
\bot &\quad data = data' \\
&\quad \land N = N'
\end{align*}
\]

\[
\begin{align*}
tree &= tree' \\
\mid cost - cost' \mid &< \varepsilon \\
\bot &\quad tree = tree' \\
&\quad \land cost = cost'
\end{align*}
\]
Inferring Strongest Post III

$N=10, \ data=D2 \quad \rightarrow \quad \text{cost}=2.4, \ \text{tree}=T3$

$N=10, \ data=D2 \quad \rightarrow \quad \text{cost}=2.39, \ \text{tree}=T4$

\[
\begin{align*}
\text{true} & \quad \rightarrow \quad \text{tree}.\text{equals}(\text{tree}') \land |\text{cost} - \text{cost}'| < \varepsilon \land \text{data} = \text{data}' \\
\downarrow \quad & \quad \rightarrow \quad \text{true} \\
N = N' & \quad \land \text{data} = \text{data}' \\
\downarrow \quad & \quad \rightarrow \quad \text{false} \\
\text{data} = \text{data}' & \quad \land N = N' \\
\end{align*}
\]
Inferring Strongest Post III

- $N=10$, data=$D2$ ➔ $\text{cost}=2.4$, $\text{tree}=T3$
- $N=10$, data=$D2$ ➔ $\text{cost}=2.39$, $\text{tree}=T4$

```
true

N = N'  data = data'

⊥: data = data' ∨ N = N'
```

```
true

<table>
<thead>
<tr>
<th>tree.equals(tree')</th>
<th>cost - cost'</th>
<th>&lt; ε</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>tree.equals(tree')</th>
<th>cost = cost'</th>
</tr>
</thead>
</table>

⊥: tree.equals(tree') ∧ cost = cost'}
Inferring Strongest Post III

\[ \text{true} \]

\[ N = N' \]
\[ \text{data} = \text{data}' \]
\[ \bot: \text{data} = \text{data}' \land N = N' \]

\[ \text{true} \]

\[ \text{tree.equals(tree')} \]
\[ \mid \text{cost} - \text{cost}' \mid < \varepsilon \]
\[ \bot: \text{tree.equals(tree')} \land \text{cost} = \text{cost}' \]

\[ \text{N=10, data= \text{D2}} \]
\[ \text{cost=2.4, tree= \text{T3}} \]

\[ \text{N=10, data= \text{D2}} \]
\[ \text{cost=2.39, tree= \text{T4}} \]
Proposition 1. The inferred postcondition \( \text{Post}_R \) is the unique strongest postcondition given the observed runs.

I.e., if \( \varphi_{pre} \xrightarrow{P,R} \varphi_{post} \) then \( \text{Post}_R \Rightarrow \varphi_{post} \).
Proposition 1. The inferred postcondition \( Post_R \) is the unique strongest postcondition given the observed runs. I.e., if \( \varphi_{pre} \xrightarrow{P,R} \varphi_{post} \) then \( Post_R \Rightarrow \varphi_{post} \).

Corollary 2. Let the inferred specification be \( Pre_R \Rightarrow Post_R \). Then, \( Post_R \) is the strongest postcondition of \( Pre_R \). I.e., if \( Pre_R \xrightarrow{P,R} \varphi_{post} \) then \( Post_R \Rightarrow \varphi_{post} \).
Proposition 3. The inferred postcondition $Post_R$ is at least as strong as the true unique strongest postcondition $SP_P(\bot)$.

I.e., if $\varphi_{pre} \xrightarrow{P} \varphi_{post}$ then $Post_R \Rightarrow \varphi_{post}$. 

```
true
N = N'
data = data'
\bot: data = data'
\land N = N'
tree.equals(tree')
|cost - cost'| < \epsilon
cost = cost'
\bot: tree.equals(tree')
\land cost = cost'
```
Proposition 3. The inferred postcondition $Post_R$ is at least as strong as the true unique strongest postcondition $SP_P(\bot)$.

I.e., if $\varphi_{pre} \xrightarrow{P} \varphi_{post}$ then $Post_R \Rightarrow \varphi_{post}$.

Proposition 4. More observed executions lead to a weaker inferred postcondition.

That is, if $R_1 \subseteq R_2 \subseteq R_3 \subseteq \cdots$ then

$Post_{R_1} \Rightarrow Post_{R_2} \Rightarrow \cdots \Rightarrow SP_P(\bot)$. 
Inferring Weakest Precondition

- Repeatedly weaken the precondition
  - As long as it still ensures the postcondition on every pair of observed executions.
Inferring Weakest Pre I

\[ N = N' \]
\[ \text{data} = \text{data}' \]
\[ \text{tree.equals(tree')} \]
\[ |cost - cost'| < \varepsilon \]
\[ \text{cost} = \text{cost}' \]
Inferring Weakest Pre I

- N=10, data=D1 → cost=3.7, tree=T1
- N=10, data=D2 → cost=2.4, tree=T3

```
true
```

```
\begin{align*}
N &= N' \\
data &= data' \\
\downarrow: & \quad data = data' \\
& \land N = N'
\end{align*}
```

```
true
```

```
\begin{align*}
tree.equals(tree') \\
|cost - cost'| &< \epsilon
\end{align*}
```

```
\begin{align*}
tree.equals(tree') \\
\land |cost - cost'| &< \epsilon
\end{align*}
```

```
\begin{align*}
\downarrow: & \quad tree.equals(tree') \\
& \land cost = cost'
\end{align*}
```
Inferring Weakest Pre II

\[ N = N' \]
\[ data = data' \]
\[ \bot : \quad data = data' \wedge N = N' \]

true

\[ tree.equals(tree') \]
\[ cost - cost' < \varepsilon \]

true

\[ tree.equals(tree') \]
\[ \left| cost - cost' \right| < \varepsilon \]

\[ cost = cost' \]

\[ \bot : \quad tree.equals(tree') \wedge cost = cost' \]
For all observed pairs of runs:

\[
data = data' \quad \xrightarrow{\text{min\_phylo\_tree}} \quad |cost - cost'| < \varepsilon
\]
Inferred Weakest Precondition

\[
\begin{align*}
N &= N' \\
data &= data' \\
\bot : & \quad data = data' \\
& \quad \land N = N'
\end{align*}
\]

\[
\begin{align*}
\text{true} \\
tree.equals(tree') \\
\quad \land |cost - cost'| < \epsilon \\
\bot : & \quad tree.equals(tree') \\
& \quad \land cost = cost'
\end{align*}
\]
Proposition 5. Let the inferred specification be $Pre_R \rightarrow Post_R$. Then, $Pre_R$ is a weakest precondition of $Post_R$ for the observed runs.

I.e., if $\varphi_{pre} \xrightarrow{P.R} \text{Post}_R$ and $\varphi_{pre} \Rightarrow Pre_R$,
then $\varphi_{pre} = Pre_R$.
Motivation and Overview

Background: Deterministic Specs

Specification Inference Problem

Inferring Deterministic Specifications

Experimental Evaluation

Related Work

Conclusions
Determinism Inference Experiments

- For previous benchmarks, infer specs for manually-identified deterministic blocks.
  - Benchmarks: 8 from Java Grande Forum (JGF), 4 from Parallel Java (PJ) Library.
Implementation (for Java)

- Record memory graph at open and close of deterministic block.
- Three equality predicates considered:
  - equals(), approximate equality, set equality
  - Compare any chain of fields (up to length 8): e.g., `this.tree.cost`, `Harness.matrix`
- Heuristics to reduce specification size.
  - By removing “uninteresting” conjuncts.
Heuristics needed to shrink specifications:

- Remove inputs from postcondition:
  If no run changes $v$, don’t include $v=v'$. 

\[ \text{data}=\text{data}' \Rightarrow \text{cost}=\text{cost}' \land \text{data}=\text{data}' \]
Implementation (for Java)

- Heuristics needed to shrink specifications:
  - Remove inputs from postcondition.
  - **Remove constants from pre- and post-:**
    If \( v \) is equal in every run, don’t include \( v = v' \).

\[
data = data' \land MAX\_MEM = MAX\_MEM' \\
\implies cost = cost' \land done = done'
\]
Heuristics needed to shrink specifications:

- Remove inputs from postcondition.
- Remove constants from pre- and post-.
- **Remove redundant conditions:**

```
params=params'
=> point.equals(point')
  ∧ point.x=point.x'
  ∧ point.y=point.y'
```
For previous benchmarks, infer specs for manually-identified deterministic blocks.
• Benchmarks: 8 from Java Grande Forum (JGF), 4 from Parallel Java (PJ) Library.

Compare to manual specifications.
• Are inferred specs correct?
• Capture intended deterministic behavior?
• Small enough to be useful?
Experimental Results

- Inferred specification vs. manual one:
  - Equivalent for 7/13 benchmarks.

Manual:

\[
\text{params} = \text{params}' \quad \rightarrow \quad \text{matrix} = \text{matrix}'
\]

Inferred:

\[
\text{params} = \text{params}' \quad \rightarrow \quad \text{matrix} = \text{matrix}' \quad ^\wedge \quad \text{img.equals(img')}
\]
Experimental Results

- Inferred specification vs. manual one:
  - Equivalent for 7/13 benchmarks.
  - Inferred correct while manual wrong for 2/13!
Experimental Results

- Inferred specification vs. manual one:
  - Equivalent for 7/13 benchmarks.
  - Inferred correct while manual wrong for 2/13!
  - 1/13 is correct but stronger than desired.

Manual:
\[ \text{params}=\text{params}' \quad \rightarrow \quad |\text{ek}[0]-\text{ek}[0]'| < \varepsilon \]

Inferred:
\[ \text{params}=\text{params}' \land \text{nthreads}=\text{nthreads}' \quad \rightarrow \quad |\text{ek}-\text{ek}'| < \varepsilon \]
## Experimental Results: JGF

<table>
<thead>
<tr>
<th>Bench</th>
<th>LoC</th>
<th>Size of Precondition</th>
<th>Size of Postcondition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Manual</td>
<td>Inferred</td>
</tr>
<tr>
<td>series</td>
<td>800</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>crypt</td>
<td>1.1k</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>moldyn</td>
<td>1.3k</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>raytracer</td>
<td>1.9k</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>monte</td>
<td>3.6k</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
## Experimental Results: PJ and tsp

<table>
<thead>
<tr>
<th>Bench</th>
<th>LoC</th>
<th>Size of Precondition</th>
<th>Size of Postcondition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Manual</td>
<td>Inferred</td>
</tr>
<tr>
<td>pi3</td>
<td>150</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>keysearch3</td>
<td>200</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>mandelbrot</td>
<td>250</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>phylogeny</td>
<td>4.4k</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>tsp*</td>
<td>700</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Limitations:

For 3/13 benchmarks, inferred spec is incorrect because of insufficient test suite.

Manual:

\[
\text{graph} = \text{graph}' \quad \rightarrow \quad \text{tour}.\text{cost} = \text{tour}.\text{cost}'
\]

Inferred:

\[
\text{graph} = \text{graph}' \quad \rightarrow \quad \text{tour}.\text{equals}(\text{tour}')
\]
Outline

- Motivation and Overview
- Background: Deterministic Specs
- Specification Inference Problem
- Inferring Deterministic Specs
- Experimental Evaluation
- Related Work
- Conclusions
Related Work: Determinism

- Deterministic languages and runtimes.
  - Deterministic Parallel Java (UIUC)
  - Kendo (Olszewski, et al, ASPLOS 09)
  - DMP (Devietti, et al, ASPLOS 09)

- Determinism Checking.
  - SingleTrack (Sadowski, et al, ESOP 09)
  - Race detection can be thought of as determinism checking.
Outline

- Motivation and Overview
- Background: Deterministic Specs
- Specification Inference Problem
- Inferring Deterministic Specs
- Experimental Evaluation
- Related Work
- Conclusions
Conclusions

- Deterministic specifications – lightweight spec of parallel correctness.
  - Much simpler structure than functional correctness specifications.
- Can infer high-quality deterministic specs.
  - From small number of observed runs.
  - Mostly recovered previous manual specs.
  - Found two errors in previous manual specs.
Any Questions?