Variance Analyses
from Invariance Analyses

Josh Berdine
jjb@microsoft.com
Microsoft Research, Cambridge

Joint work with Aziem Chawdhary, Byron Cook, Dino Distefano & Peter O’Hearn

SAVCBS’06: 10 Nov 2006
State of the verification toolbox

Safety properties & reachability:
- For proving that software doesn’t “crash”
- Many verification tools & techniques at hand
  - Software model checkers, e.g. SLAM, Blast, SATAbs,…
  - Abstract domains: e.g. Interval, Octagon, Polyhedra,…
  - Other static analyzers: e.g. various control-flow, shape,… analyses
- Not insignificant degree of coverage and maturity

Liveness & termination:
- For proving that software does “react”
- Fewer verification tools
- Often not as general, each strongly tailored to a form of programs
- Sometimes “inconvenient” restrictions: e.g. no nested loops, purely functional

Here: constructing termination provers from safety analyzers
Termination provers for free!

- Take an invariance analysis as a parameter
  - Computes an invariance assertion for each program location
  - An invariance assertion for $l$ holds of all reachable states at $l$

- Construct its induced variance analysis
  - Computes a variance assertion for each program location
  - A variance assertion for $l$ holds between any reachable state at $l$ and any previous state at $l$

- Yields a termination prover
  - We give a local termination predicate $LT$ such that
  - Program terminates if $LT$ holds of each program location’s variance assertion

- Need two additional operations on abstract representation
  - Seed & WellFounded
  - Not difficult to define in practice
The plan

- Introduction
- Overview induced variance analysis algorithm
- Local termination predicates
- Play-by-play for an example
- Requirements on instantiations
- Instantiation for numerical abstract domains
- Instantiation for shape analysis
- Conclusion
Parameterized variance analysis algorithm

01 VARIANCEANALYSIS(P, L, I#) {
02 IAs := INVARIANCEANALYSIS(P, I#)
03 foreach ℓ ∈ L {
04 \hspace{1em} LTPreds[ℓ] := true
05 \hspace{1em} O := ISOLATE(P, L, ℓ)
06 \hspace{2em} foreach q ∈ IAs such that pc(q) = ℓ {
07 \hspace{3em} VAs := INVARIANCEANALYSIS(O, STEP(O, \{SEED(q)\}))
08 \hspace{4em} foreach r ∈ VAs {
09 \hspace{5em} if pc(r) = ℓ ∧ ¬WELLFOUNDED(r) {
10 \hspace{6em} LTPreds[ℓ] := false
11 \hspace{5em} }
12 \hspace{4em} }
13 \hspace{2em} }
14 \hspace{1em} }
15 \hspace{0em} return LTPreds
16 }
Parameterized variance analysis algorithm

```
01 VARIANCE_ANALYSIS(P, L, I^#) {
02    IAs := INVARINANCE_ANALYSIS(P, I^#)
03    foreach ℓ ∈ L {
04        LPreds[ℓ] := true
05        O := ISOLATE(P, L, ℓ)
06                foreach q ∈ IAs such that pc(q) = ℓ {
07                      VAs := INVARINANCE_ANALYSIS(O, STEP(O, {SEED(q)}))
08                        foreach r ∈ VAs {
09                            if pc(r) = ℓ ∧ ¬WELL_FOUNDEDED(r) {
10                                LPreds[ℓ] := false
11                            }
12                        }
13                }
14            return LPreds
15        }
16    }
```
Parameterized variance analysis algorithm

01 \textbf{VarianceAnalysis}(P, L, I^+) \{ 
02 \textit{IA}s := \textbf{InvarianceAnalysis}(P, I^+) 
03 \textbf{foreach} \ell \in L \{ 
04 \textit{LTP}reds[\ell] := true 
05 \textbf{O :=} \textbf{Isolate}(P, L, \ell) 
06 \textbf{foreach} q \in \textit{IA}s \text{ such that } \text{pc}(q) = \ell \{ 
07 \textit{VA}s := \textbf{InvarianceAnalysis}(O, \textbf{Step}(O, \{\text{Seed}(q)\})) 
08 \textbf{foreach} r \in \textit{VA}s \{ 
09 \textbf{if} \text{pc}(r) = \ell \land \neg \textbf{WellFounded}(r) \{ 
10 \textit{LTP}reds[\ell] := false 
11 \}
12 \}
13 \}
14 \} 
15 \textbf{return} \textit{LTP}reds
16 \}
Local termination predicates

```java
82   while (x>a && y>b) {
83       if (nondet()) {
84           do {
85               x = x - 1;
86           } while (x>10);
87       } else {
88           y = y - 1;
89       }
90   }
```

- Line 83 is not visited infinitely often ✔
- Line 85 is not visited infinitely often ✔
- Program terminates
Local termination predicates

81 while (nondet()) {
82   while (x>a && y>b) {
83     if (nondet()) {
84       do {
85         x = x - 1;
86       } while (x>10);
87     } else {
88       y = y - 1;
89     }
90   }
91 }

→ Line 83 is visited infinitely often

→ Program diverges

but…

→ $LT(83)$: Line 83 is visited infinitely often only when the program’s execution exits the loop contained in lines 82 to 90 infinitely often
Local termination predicates

81 while (nondet()) {
82     while (x>a && y>b) {
83         if (nondet()) {
84             do {
85                 x = x - 1;
86             } while (nondet());
87         } else {
88             y = y - 1;
89         }
90     }
91 }

→ Line 85 is visited infinitely often

→ Program diverges

but still…

→ \( LT(83) \): Line 83 is visited infinitely often only when the program’s execution exits the loop contained in lines 82 to 90 infinitely often
81 while (nondet()) {
82   while (x>a && y>b) {
83     if (nondet()) {
84       do {
85         x = x - 1;
86       } while (nondet());
87     } else {
88       y = y - 1;
89     }
90   }
91 }

→ $LT(82)$: Line 82 is visited infinitely often only when the program’s execution exits the loop contained in lines 81 to 91 infinitely often  

→ $LT(83)$: Line 83 is visited infinitely often only when the program’s execution exits the loop contained in lines 82 to 90 infinitely often  

→ $LT(85)$: Line 85 is visited infinitely often only when the program’s execution exits the loop contained in lines 84 to 86 infinitely often
Consider an invariance analysis based on the Octagon domain

Can express conjunctions of inequalities of the form: $\pm x + \pm y \leq c$

Represent the program counter with equalities: $pc = c$

```c
while (nondet()) {
    while (x > a && y > b) {
        if (nondet()) {
            do {
                x = x - 1;
            } while (nondet());
        } else {
            y = y - 1;
        }
    }
}
```
Illustrative example

01 \textbf{VarianceAnalysis}(P, L, I^\#) \{ \\
02 \quad IAs := \textbf{InvarianceAnalysis}(P, I^\#) \\
03 \quad \textbf{foreach} \ell \in L \{ \\
04 \quad \quad LTPreds[\ell] := \textbf{true} \\
05 \quad \quad O := \textbf{Isolate}(P, L, \ell) \\
06 \quad \quad \textbf{foreach} q \in IAs \text{ such that } pc(q) = \ell \{ \\
07 \quad \quad \quad VAs := \textbf{InvarianceAnalysis}(O, \text{Step}(O, \{\text{Seed}(q)\})) \\
08 \quad \quad \quad \textbf{foreach} r \in VAs \{ \\
09 \quad \quad \quad \quad \textbf{if } pc(r) = \ell \land \neg \text{WellFounded}(r) \{ \\
10 \quad \quad \quad \quad \quad LTPreds[\ell] := \textbf{false} \\
11 \quad \quad \quad \quad \} \\
12 \quad \quad \} \\
13 \quad \} \\
14 \quad \textbf{return} LTPreds \\
15 \} \\

81 \quad \textbf{while} (\text{nondet}()) \{ \\
82 \quad \quad \textbf{while} (x > a \land y > b) \{ \\
83 \quad \quad \quad \textbf{if} (\text{nondet}()) \{ \\
84 \quad \quad \quad \quad \textbf{do} \{ \\
85 \quad \quad \quad \quad \quad x = x - 1; \\
86 \quad \quad \quad \quad \} \textbf{while} (\text{nondet}()); \\
87 \quad \quad \quad \} \textbf{else} \{ \\
88 \quad \quad \quad y = y - 1; \\
89 \quad \quad \} \\
90 \quad \} \\
91 \}
Illustrative example

01 \textbf{VarianceAnalysis}(P, L, I^\#) \{ \\
02 \hspace{1em} IAs := \textbf{InvarianceAnalysis}(P, I^\#) \\
03 \hspace{1em} \textbf{foreach} \ell \in L \{ \\
04 \hspace{2em} LTPreds[\ell] := \text{true} \\
05 \hspace{2em} O := \text{Isolate}(P, L, \ell) \\
06 \hspace{2em} \textbf{foreach} q \in IAs \text{ such that } \text{pc}(q) = \ell \{ \\
07 \hspace{3em} VAs := \textbf{InvarianceAnalysis}(O, \text{Step}(O, \{\text{Seed}(q)\})) \\
08 \hspace{3em} \textbf{foreach} r \in VAs \{ \\
09 \hspace{4em} \text{if } \text{pc}(r) = \ell \land \neg \text{WellFounded}(r) \{ \\
10 \hspace{5em} LTPreds[\ell] := \text{false} \\
11 \hspace{4em} \} \\
12 \hspace{3em} \} \\
13 \} \\
14 \} \\
15 \textbf{return} LTPreds \\
16 \} \\

\text{pc}=81 \land x \geq a + 1 \land y \geq b + 1
Illustrative example

01 \textsc{VarianceAnalysis}(P, L, I#) \{
02 \quad \text{IAs} := \textsc{InvarianceAnalysis}(P, I#)
03 \quad \textbf{foreach} \ l \in L \{
04 \quad \quad \text{LTPreds}[l] := \texttt{true}
05 \quad \quad O := \textsc{Isolate}(P, L, l)
06 \quad \quad \textbf{foreach} q \in \text{IAs} \text{ such that } pc(q) \quad \{
07 \quad \quad \quad \quad \text{VAs} := \textsc{InvarianceAnalysis}(P, q, I#)
08 \quad \quad \quad \quad \textbf{foreach} r \in \text{VAs} \{
09 \quad \quad \quad \quad \quad \textbf{if} \ pc(r) = l \land \neg \texttt{WellFounded} \;
10 \quad \quad \quad \quad \quad \quad \text{LTPreds}[l] := \texttt{false}
11 \quad \quad \quad \quad \}{\}
12 \quad \quad \}{\}
13 \quad \}{\}
14 \quad \textbf{return} \ LTPreds
15 \}
16 \}

\text{pc} = 81 \land x \geq a + 1 \land y \geq b + 1

\text{pc} = 83 \land x \geq a + 1 \land y \geq b + 1

\{s \mid s(pc) = 83 \land s(x) \geq s(a) + 1 \land s(y) \geq s(b) + 1\}
Illustrative example

01 VarianceAnalysis(P, L, I#) {
02 IAs := InvarianceAnalysis(P, I#)
03 foreach ℓ ∈ L {
04     LTPreds[ℓ] := true
05     O := Isolate(P, L, ℓ)
06     foreach q ∈ IAs such that pc(q) = ℓ {
07         VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
08         foreach r ∈ VAs {
09             if pc(r) = ℓ ∧ ¬WellFounded(r) {
10                 LTPreds[ℓ] := false
11             }
12         }
13     }
14 return LTPreds
15 }

while (nondet()) {
    while (x > a && y > b) {
        if (nondet()) {
            do {
                x = x - 1;
            } while (nondet());
        } else {
            y = y - 1;
        }
    } else {
    }
} ; assume(false);
Illustrative example

```c
01 VARIANCEANALYSIS(P, L, I#) {
02   IAs := INVARIANCEANALYSIS(P, I#)
03   foreach ℓ ∈ L {
04     LTPreds[ℓ] := true
05     O := ISOLATE(P, L, ℓ)
06     foreach q ∈ IAs such that pc(q) = ℓ {
07       VAs := INVARIANCEANALYSIS(O, STEP(O, {SEED(q)}))
08       foreach r ∈ VAs {
09         if pc(r) = ℓ ∧ ¬WELLFOUNDED(r) {
10           LTPreds[ℓ] := false
11         }
12       }
13   }
14   return LTPreds
15 }
```

```
while (nondet()) {
  while (x > a && y > b) {
    if (nondet()) {
      do {
        x = x - 1;
        } while (nondet());
    } else {
      y = y - 1;
    }
  }
}; assume(false);```

\[ pc=83 \land x \geq a + 1 \land y \geq b + 1 \]
Illustrative example

01 `VarianceAnalysis(P, L, I#)` {  
02     `IAs := InvarianceAnalysis(P, I#)`  
03     `foreach ℓ ∈ L` {  
04         `LTPpreds[ℓ] := true`  
05         `O := Isolate(P, L, ℓ)`  
06         `foreach q ∈ IAs such that pc(q) = ℓ` {  
07             `VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))`  
08             `foreach r ∈ VAs` {  
09                 `if pc(r) = ℓ ∧ ¬WellFormed`  
10                     `LTPpreds[ℓ] := false`  
11             }  
12         }  
13     }  
14     `return LTPpreds`  
15 }
Illustrative example

01 VARIANCE ANALYSIS(P, L, I#) {
02   IAs := INVARIANCE ANALYSIS(P, I#)
03     foreach ℓ ∈ L {
04       LTPreds[ℓ] := true
05       O := ISOLATE(P, L, ℓ)
06       foreach q ∈ IAs such that pc(q) = ℓ {
07         VAs := INVARIANCE ANALYSIS(O, STEP(O, {SEED(q)}))
08           foreach r ∈ VAs {
09             if pc(r) = ℓ ∧ ¬WELLFORMED[r] {
10               pc = 83 ∧ x ≥ a + 1 ∧ y ≥ b + 1
11                 ∧ pcₙ = pc ∧ xₙ = x ∧ yₙ = y ∧ aₙ = a ∧ bₙ = b
12           } }
13     } }
14 return LTPreds
15 }

{(s,t) | s(pc)=t(pc)=83 ∧ s(x)=t(x) ∧ s(y)=t(y) ∧ s(a)=t(a) ∧ s(b)=t(b) ∧ t(x) ≥ t(a) + 1 ∧ t(y) ≥ t(b) + 1 }

90   }; assume(false);
Illustrative example

```c
01 VarianceAnalysis(P, L, I#) {
02    IAs := InvarianceAnalysis(P, I#)
03    foreach ℓ ∈ L {
04        LTPreds[ℓ] := true
05        O := Isolate(P, L, ℓ)
06        foreach q ∈ IAs such that pc(q) = ℓ {
07            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
08            foreach r ∈ VAs {
09                if pc(r) = ℓ ∧ ¬WellR
10                LTPreds[ℓ] :=
11            }
12    pc=83 ∧ x ≥ a + 1 ∧ y ≥ b + 1 ∧ pc_s=pc ∧ x_s=x ∧ y_s=y ∧ a_s=a ∧ b_s=b
13    pc=83 ∧ pc=84 ∧ x ≥ a + 1 ∧ y ≥ b + 1
14    return LTPreds
15 }
16 }
```

```c
82     while (x>a && y>b) {
83         if (nondet()) {
84             do {
85                 x = x - 1;
86             } while (nondet());
87         } else {
88             y = y - 1;
89         }
90     }; assume(false);
91 }
```
Illustrative example

```
01 VarianceAnalysis(P, L, I#) {  
02   IAs := InvarianceAnalysis(P, I#)  
03   foreach ℓ ∈ L {  
04     LTPreds[ℓ] := true  
05     O := Isolate(P, L, ℓ)  
06     foreach q ∈ IAs such that pc(q) = ℓ {  
07       VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))  
08       foreach r ∈ VAs {  
09         if pc(r) = ℓ ∧ ¬WellFounded(r) {  
10           LTPreds[ℓ] := false  
11       }  
12       }  
13     }  
14   pc_s=83 ∧ pc=84 ∧ x ≥ a + 1 ∧ y ≥ b + 1  
15   ∧ x_s=x ∧ y_s=y ∧ a_s=a ∧ b_s=b  
15   return LTPreds  
16 }  
```
Illustrative example

01 \text{VarianceAnalysis}(P, L, i) \{ 
02 \text{IA}s := \text{InvarianceAnalysis}(P, L, i) 
03 \text{foreach} \ell \in L \{ 
04 \text{LTPreds}[^{\ell}] := + 
05 O := \text{Isolate}(P, L, \ell) 
06 \text{foreach} q \in \text{IA}s \text{ such that } \text{pc}(q) = \ell \{ 
07 VAs := \text{InvarianceAnalysis}(O, \text{Step}(O, \{\text{Seed}(q)\})) 
08 \text{foreach} r \in VAs \{ 
09 \text{if} \text{pc}(r) = \ell \land \neg \text{WellFounded}(r) \{ 
10 \text{LTPreds}[^{\ell}] := \text{false} 
11 \} 
\} 
\}
15 \text{return} LTPreds 
16 \}

\quad \{ \text{pc}_{s}=83 \land \text{pc}=83 \land x \geq a + 1 \land y \geq b + 1 \\
\quad \land x_{s}=x \land y_{s}=y \land a_{s}=a \land b_{s}=b 
\}

\quad \{ \text{pc}_{s}=83 \land \text{pc}=83 \land x \geq a + 1 \land y \geq b + 1 \\
\quad \land x_{s}=x \land y_{s}=y \land a_{s}=a \land b_{s}=b 
\}

\quad \{ \text{pc}_{s}=83 \land \text{pc}=83 \land x \geq a - 1 \land y \geq b + 1 \\
\quad \land x_{s}=x + 1 \land y_{s}=y + 1 \land a_{s}=a \land b_{s}=b 
\}

01 \text{while} (\text{nondet}()) \{ 
02 \text{while} (x > a \land y > b) \{ 
03 \text{if} (\text{nondet}()) \{ 
04 \text{do} \{ 
05 x = x - 1; 
06 \} \text{while} (\text{nondet}()); 
07 \} \text{else} \{ 
08 y = y - 1; 
09 \} 
10 \}; \text{assume}(\text{false}); 
11 \}
Illustrative example

```
01 VARIANCE ANALYSIS(P, L, k) {
02     IAs := INVARIANCE ANALYSIS(P, L, k);
03     foreach ℓ ∈ L {
04         LTPreds[ℓ] := true;
05         O := ISOLATION(P, L, ℓ);
06     }
07     foreach q ∈ IAs such that pc(q) = ℓ {
08         VAs := INVARIANCE ANALYSIS(O, STEP(O, {SEED(q)}));
09         foreach r ∈ VAs {
10             if pc(r) = ℓ ∧ ¬WELLFOUNDED(r) {
11                 LTPreds[ℓ] := false
12             }
13         }
14     }
15     return LTPreds
16 }
```

"A superset of the possible transitions from states at 83 to states also at line 83 reachable in 1 or more steps of the program's execution"

```
81 while (nondet()) {
82     while (x>a && y>b) {
83         if (nondet()) {
84             do {
85                 x = x - 1;
86             } while (nondet());
87         } else {
88             y = y - 1;
89         }
90     } assume(false); };
```
A superset of the possible transitions from states at 83 to states also at line 83 reachable in 1 or more steps of the program’s execution

\[ \subseteq \{ \text{pcs}=83 \land \text{pc}=83 \land x \geq a + 1 \land y \geq b + 1 \\
\land x_s \geq x + 1 \land y_s \geq y \land a_s=a \land b_s=b \}
\]

If \( LTPreds[l] = \text{true} \), then \( VAs \) is a finite disjunction of well-founded relations that over-approximates \( R^+ \). Then isolated program terminates by Podelski & Rybalchenko [LICS04]
Remarks

- Speed: the induced termination provers are fast:
  - 0.07s for Octagon-based prover on this example, vs 8.3s for Terminator

- Automatic:
  - Termination arguments are automatically found and checked

- Disjunctive termination arguments:
  - Disjunctive decomposition under the control of the invariance analysis
  - Allows using invariance analyzers based on simpler domains
    - Traditional ranking function for blue loop is:
      $$ f(s) = s(x) + s(y) $$
      and the program’s transition relation
      (whose coverage must be proven) is:
      $$ \{(s,t) | s(x) + s(y) \geq t(x) + t(y) - 1 \land t(x) + t(y) \geq 0\} $$
      Note the 4-variable inequality.

```plaintext
81  while (nondet()) {
82    while (x>a && y>b) {
83      if (nondet()) {
84        do {
85          x = x - 1;
86        } while (x>10);
87      } else {
88        y = y - 1;
89      }
90    }
91 }
```
Remarks

→ Dynamic seeding: improved precision
  ▪ Seeding may be done after some disjunctive decomposition
  ▪ Auxiliary information kept by the invariance analysis can be seeded

→ No rank function synthesis:
  ▪ Well-foundedness checks only need boolean result, a full rank-function synthesizer is unnecessary

→ Some usable information is computed whether or not overall termination is established
  ▪ The well-founded disjuncts that are found provide refinement-based tools like Terminator with a much better starting point

→ Robust wrt nested loops, etc. by use of standard analysis methods
  ▪ Fits in comfortably with cutpoint decomposition techniques

→ Over-approximation of program’s transition relation holds by construction, in Terminator checking this is the performance bottleneck
Instantiating the algorithm: Seed & WellFounded

- Seed encodes a binary relation on states into a predicate on states
- *Ghost state* is the additional information in a state used to represent a relation (the seed variables)
- Seeding must introduce ghost state, approximating copying the state, in a fashion such that:
  - The concrete semantics is independent of any ghost state
  - The abstract semantics (InvarianceAnalysis) must ignore the ghost state and not introduce spurious facts about it

- WellFounded must soundly check well-foundedness of the relations seeded states represent

and of course:

- Step and InvarianceAnalysis must be sound over-approximations of the program’s concrete semantics
Induced termination provers for numerical domains

→ Take a conventional invariance analysis based on the Ocatgon or Polyhedra abstract domains

→ Fit a post-analysis phase that recovers some disjunctive information

→ Define:

\[
\begin{align*}
\text{SEED}(F') & \triangleq F \land \bigwedge_{v \in \text{PVar}} \{v = \rho(v)\} \\
\text{WELLFOUNDED}(F') & \triangleq \text{WFCHECK}(\rho(\text{PVar}), \text{PVar}, F')
\end{align*}
\]

- \(\rho\) is a bijection between program and seed variables
- WfCheck can be e.g. RankFinder or PolyRank

→ That’s it!
Induced termination provers for numerical domains

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>0.11</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>P</td>
<td>1.40</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>PR</td>
<td>0.02</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>T/O</td>
<td>-</td>
</tr>
<tr>
<td>T</td>
<td>6.31</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>T/O</td>
<td>-</td>
</tr>
</tbody>
</table>

(a) Results from experiments with termination tools on arithmetic examples from the Octagon Library distribution.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>0.30</td>
<td>†</td>
<td>0.05</td>
<td>†</td>
<td>0.11</td>
<td>†</td>
<td>0.50</td>
<td>†</td>
<td>0.10</td>
<td>†</td>
<td>0.17</td>
<td>†</td>
</tr>
<tr>
<td>P</td>
<td>1.42</td>
<td>✔</td>
<td>0.82</td>
<td>✔</td>
<td>1.06</td>
<td>†</td>
<td>2.29</td>
<td>†</td>
<td>2.61</td>
<td>†</td>
<td>1.28</td>
<td>†</td>
</tr>
<tr>
<td>PR</td>
<td>0.21</td>
<td>✔</td>
<td>0.13</td>
<td>✔</td>
<td>0.44</td>
<td>✔</td>
<td>1.62</td>
<td>✔</td>
<td>3.88</td>
<td>✔</td>
<td>0.11</td>
<td>✔</td>
</tr>
<tr>
<td>T</td>
<td>435.23</td>
<td>✔</td>
<td>61.15</td>
<td>✔</td>
<td>T/O</td>
<td>-</td>
<td>T/O</td>
<td>-</td>
<td>75.33</td>
<td>✔</td>
<td>T/O</td>
<td>-</td>
</tr>
</tbody>
</table>

(b) Results from experiments with termination tools on arithmetic examples from the PolyRank distribution.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>1.42</td>
<td>✔</td>
<td>1.67</td>
<td>⊘</td>
<td>0.47</td>
<td>⊘</td>
<td>0.18</td>
<td>✔</td>
<td>0.06</td>
<td>✔</td>
</tr>
<tr>
<td>P</td>
<td>4.66</td>
<td>✔</td>
<td>6.35</td>
<td>⊘</td>
<td>1.48</td>
<td>⊘</td>
<td>1.10</td>
<td>✔</td>
<td>1.30</td>
<td>✔</td>
</tr>
<tr>
<td>PR</td>
<td>T/O</td>
<td>-</td>
<td>T/O</td>
<td>-</td>
<td>T/O</td>
<td>-</td>
<td>T/O</td>
<td>-</td>
<td>0.10</td>
<td>✔</td>
</tr>
<tr>
<td>T</td>
<td>10.22</td>
<td>✔</td>
<td>31.51</td>
<td>⊘</td>
<td>20.65</td>
<td>⊘</td>
<td>4.05</td>
<td>✔</td>
<td>12.63</td>
<td>✔</td>
</tr>
</tbody>
</table>

(c) Results from experiments with termination tools on small arithmetic examples taken from Windows device drivers. Note that the examples are small as they must currently be hand-translated for the three tools that do not accept C syntax.
Induced termination prover for shape analysis

- Take Sonar, the separation-logic based shape analysis that tracks sizes of abstracted portions of the heap

- No post-analysis, the Sonar analysis is already fully disjunctive

- Define:

\[
\begin{align*}
\text{SEED}(\Pi \land \Sigma) & \triangleq (\Pi \land \Sigma \land \bigwedge_{v \in \text{DV}(\Pi \land \Sigma)} \{v = \rho(v)\}) \\
\text{SEED}(\top) & \triangleq \top \\
\text{WELLFOUNDED}(\Pi \land \Sigma) & \triangleq \text{WFCHECK}(\rho(\text{DVar}), \text{DVar}, \Pi) \\
\text{WELLFOUNDED}(\top) & \triangleq \text{false}
\end{align*}
\]

- $\rho$ is a bijection between list length and seeded length variables
- WfCheck can be e.g. RankFinder or PolyRank

- Surprisingly similar to instantiation for numerical domains, despite the underlying analyses being radically different
### Induced termination prover for shape analysis

<table>
<thead>
<tr>
<th>Loop</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (s)</td>
<td>0.0</td>
<td>0.0</td>
<td>8.0</td>
<td>0.3</td>
<td>1.7</td>
<td>13</td>
<td>296</td>
<td>0.1</td>
<td>5.4</td>
<td>0.0</td>
<td>8.2</td>
<td>821</td>
<td>0.0</td>
<td>1.6</td>
<td>152</td>
<td>0.0</td>
<td>2.6</td>
<td>3.5</td>
<td>58</td>
<td>32</td>
<td>261</td>
</tr>
<tr>
<td>Result</td>
<td>✓</td>
<td>⊗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>⊗</td>
<td>⊗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>⊗</td>
<td>✓</td>
<td>⊗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>WF checks</td>
<td>1</td>
<td>4</td>
<td>16</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>15</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>39</td>
<td>1</td>
<td>3</td>
<td>16</td>
<td>1</td>
<td>28</td>
<td>9</td>
<td>85</td>
<td>20</td>
<td>37</td>
</tr>
</tbody>
</table>

→ Results on examples Terminator flags as buggy

→ 1 false bug reported: loop 8, essentially reversing a pan-handle list
Conclusions

- Variance analyses can be constructed from invariance analyses

- Resulting termination provers are fast: at least competitive with the state-of-the art

- Even (quickly) failed proofs can help other provers

- Usual analysis techniques for varying the precision versus performance balance can now be done for termination

- Questions?

details in a paper to appear in POPL