



# 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



- → 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*  $\mathcal{LT}$  such that
  - Program terminates if  $\mathcal{LT}$  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

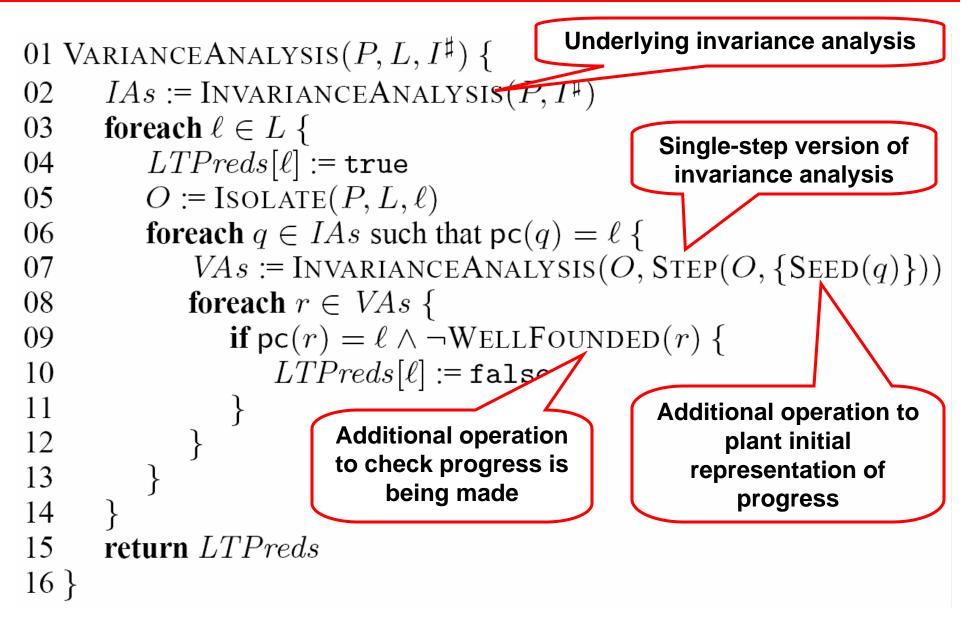


- Introduction
- Overview induced variance analysis algorithm
- Local termination predicates
- → Play-by-play for an example
- → Requirements on instantiations
- Instantiation for numerical abstract domains
- → Instantiation for shape analysis
- Conclusion

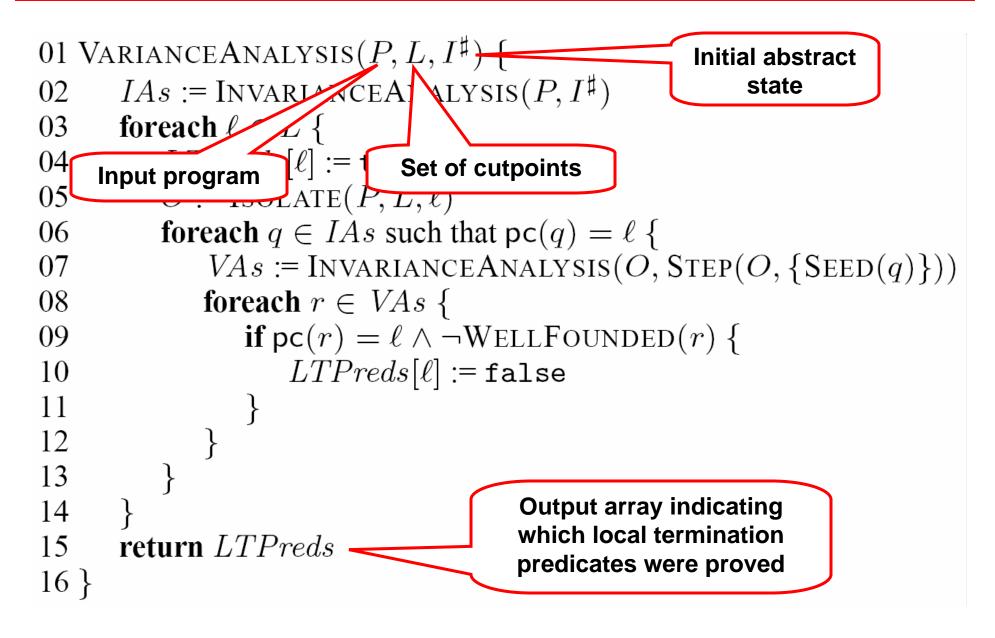
```
01 VARIANCEANALYSIS(P, L, I^{\sharp}) {
      IAs := INVARIANCEANALYSIS(P, I^{\sharp})
02
03
      foreach \ell \in L {
          LTPreds[\ell] := true
04
          O := \text{ISOLATE}(P, L, \ell)
05
06
          foreach q \in IAs such that pc(q) = \ell {
              VAs := INVARIANCEANALYSIS(O, STEP(O, {SEED(q)}))
07
             foreach r \in VAs {
08
09
                if pc(r) = \ell \land \neg WELLFOUNDED(r) {
                    LTPreds[\ell] := false
10
11
12
13
14
15
      return LTPreds
16 }
```

Microsoft\*







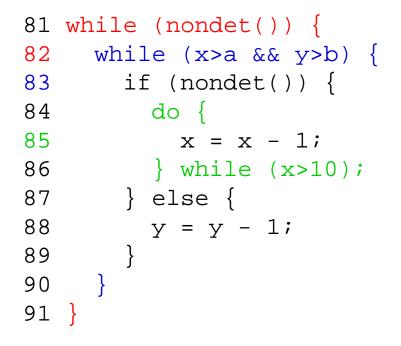




```
82 while (x>a && y>b) {
83
      if (nondet()) {
84
       do {
85
         x = x - 1;
       } while (x>10);
86
87 } else {
88
       y = y - 1;
89
      }
90
    }
```

- → Line 83 is not visited infinitely often
- → Line 85 is not visited infinitely often
- → Program terminates



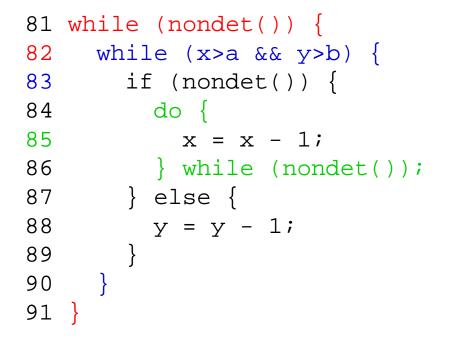


- → Line 83 is visited infinitely often
- Program diverges

but...

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





- → Line 85 is visited infinitely often
- Program diverges

but still...

→  $\mathcal{LT}(83)$ : 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()) {
    while (x>a \&\& y>b) {
82
83
      if (nondet()) {
84
        do {
85
          x = x - 1;
86
        } while (nondet());
87 } else {
88
        y = y - 1;
89
90
    }
91 }
```

- →  $\mathcal{LT}(82)$ : 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
- →  $\mathcal{LT}(85)$ : 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
- $\rightarrow$  Can express conjunctions of inequalities of the form:  $\pm x + \pm y \leq c$
- $\rightarrow$  Represent the program counter with equalities: pc = c

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

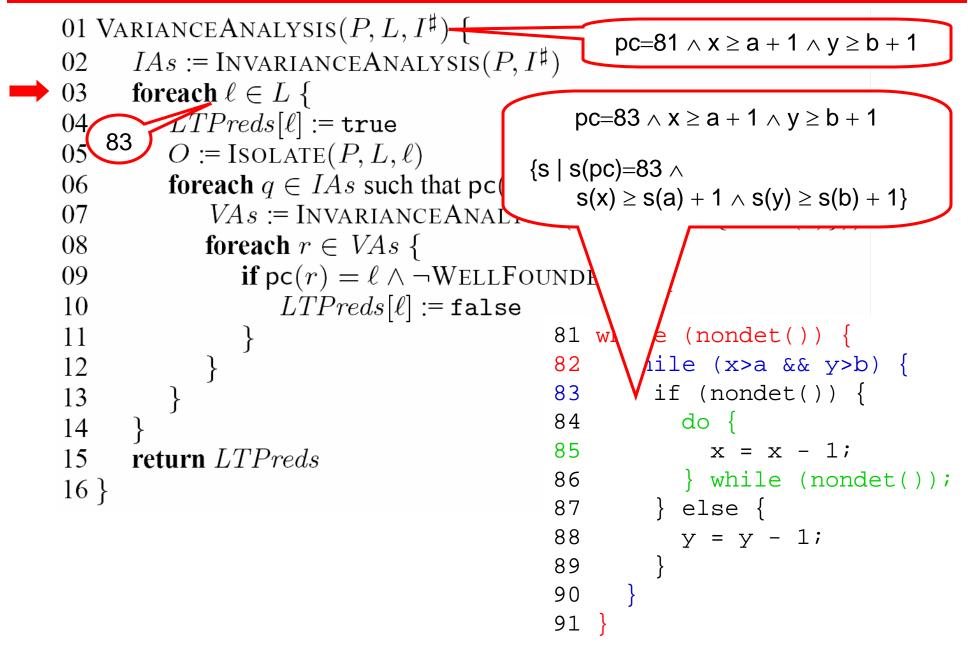


```
\rightarrow 01 VARIANCEANALYSIS(P, L, I^{\sharp}) {
         IAs := INVARIANCEANALYSIS(P, I^{\sharp})
   02
   03
         foreach \ell \in L {
   04
             LTPreds[\ell] := \texttt{true}
            O := \text{ISOLATE}(P, L, \ell)
   05
            foreach q \in IAs such that pc(q) = \ell {
   06
                VAs := INVARIANCEANALYSIS(O, STEP(O, {SEED(q)}))
   07
   08
                foreach r \in VAs {
                   if pc(r) = \ell \land \neg WELLFOUNDED(r) {
   09
                      LTPreds[\ell] := \texttt{false}
   10
                                               81 while (nondet()) {
   11
                   }
                                               82
                                                     while (x>a && y>b) {
   12
                                                        if (nondet()) {
                                               83
   13
                                                          do {
                                               84
   14
                                               85
                                                             x = x - 1;
         return LTPreds
   15
                                                          } while (nondet());
                                               86
   16 }
                                                       } else {
                                               87
                                               88
                                                          y = y - 1;
                                               89
                                               90
                                               91 }
```

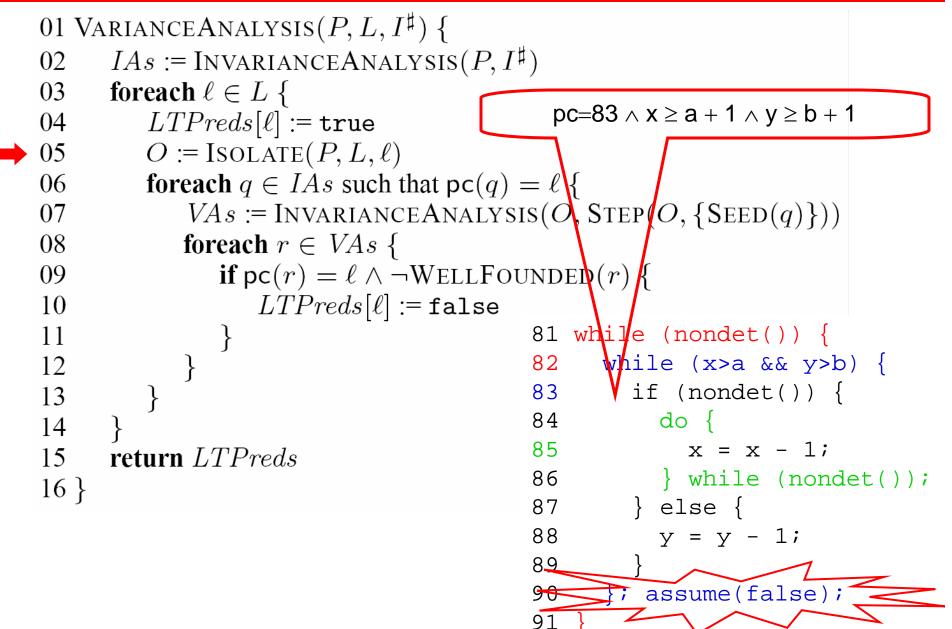


```
01 VARIANCEANALYSIS(P, L, I^{\sharp})
                                                   pc=81 \land x \ge a+1 \land y \ge b+1
       IAs := INVARIANCEANALYSIS(P, I^{\sharp})
02
       foreach \ell \in L {
 03
 04
           LTPreds[\ell] := \texttt{true}
           O := \text{ISOLATE}(P, L, \ell)
 05
           foreach q \in IAs such that pc(q) = \ell {
 06
               VAs := INVARIANCEANALYSIS(O, STEP(O, {SEED(q)}))
 07
 08
              foreach r \in VAs {
                 if pc(r) = \ell \land \neg WELLFOUNDED(r) {
 09
                     LTPreds[\ell] := \texttt{false}
 10
                                              81 while (nondet()) {
 11
                  }
                                              82
                                                    while (x>a && y>b) {
 12
                                                       if (nondet()) {
                                              83
 13
                                              84
                                                         do {
 14
                                              85
                                                            x = x - 1;
       return LTPreds
 15
                                                          } while (nondet());
                                              86
 16 }
                                                       } else {
                                              87
                                              88
                                                         y = y - 1;
                                              89
                                              90
                                              91
                                                 }
```





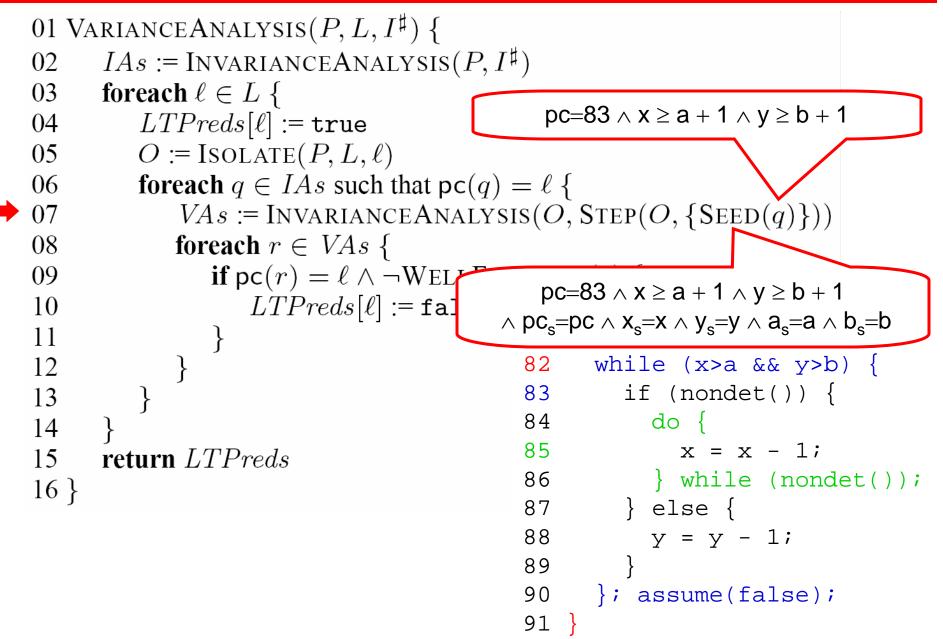




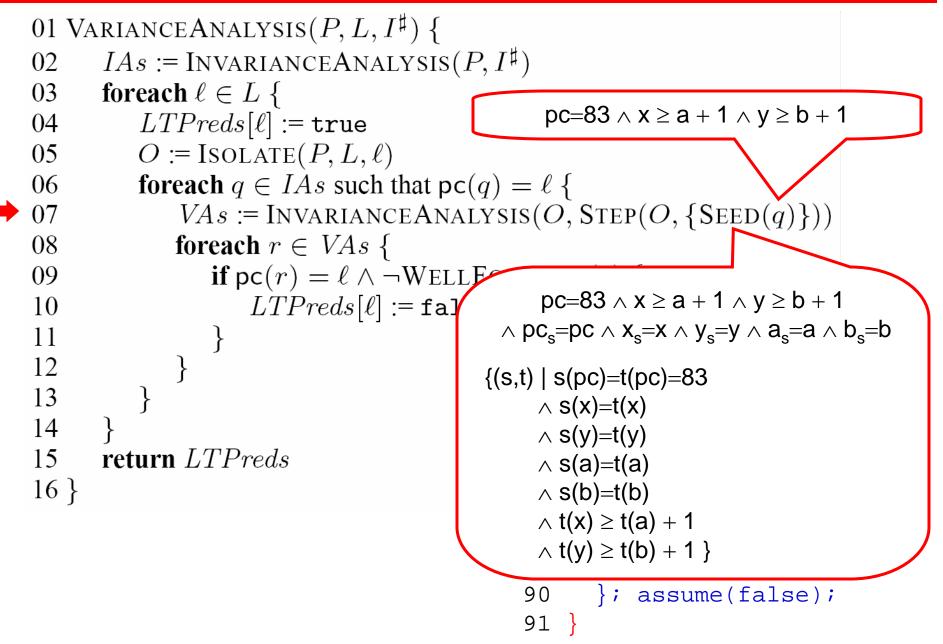


```
01 VARIANCEANALYSIS(P, L, I^{\sharp}) {
      IAs := INVARIANCEANALYSIS(P, I^{\ddagger})
02
03
      foreach \ell \in L {
                                               pc=83 \land x \ge a + 1 \land y \ge b + 1
04
         LTPreds[\ell] := \texttt{true}
         O := \text{ISOLATE}(P, L, \ell)
05
         foreach q \in IAs such that pc(q) = \ell {
06
             VAs := INVARIANCEANALYSIS(O, STEP(O, {SEED(q)}))
07
             foreach r \in VAs {
08
                if pc(r) = \ell \land \neg WELLFOUNDED(r) {
09
                   LTPreds[\ell] := \texttt{false}
10
                                             81 while (nondet()) {
11
                }
                                                   while (x>a && y>b) {
                                             82
12
                                             83
                                                      if (nondet()) {
13
                                                         do {
                                             84
14
                                             85
                                                           x = x - 1;
      return LTPreds
15
                                                         } while (nondet());
                                             86
16 }
                                                      } else {
                                             87
                                             88
                                                         y = y - 1;
                                             89
                                             90
                                                    }; assume(false);
                                             91
```

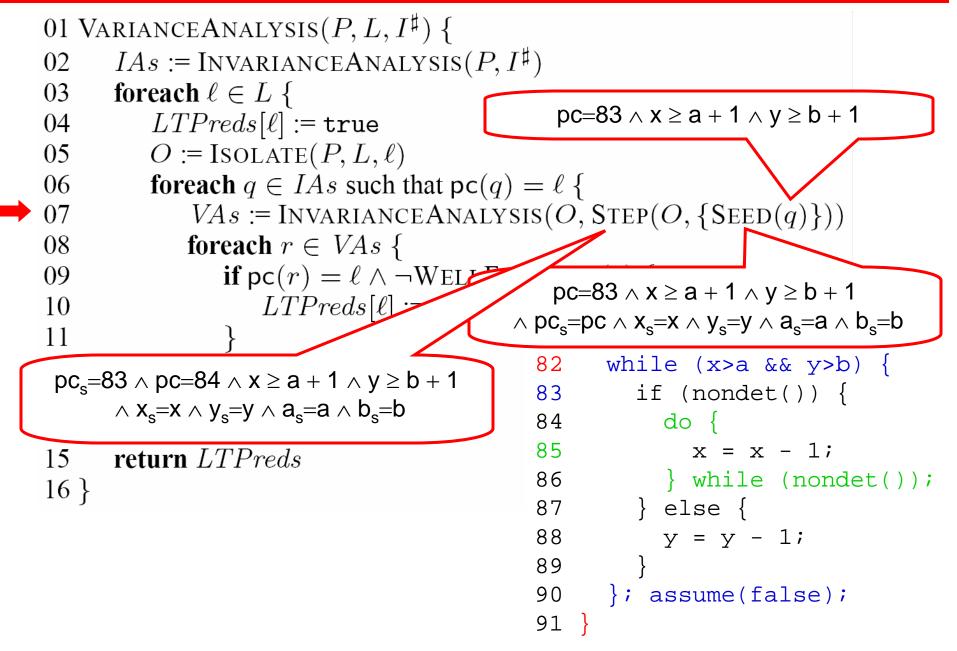




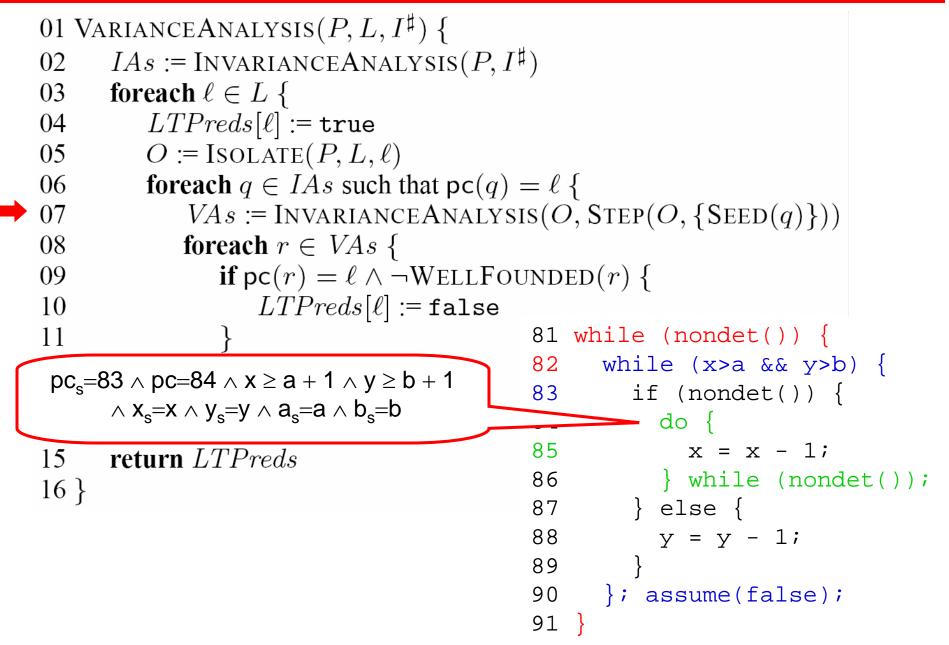






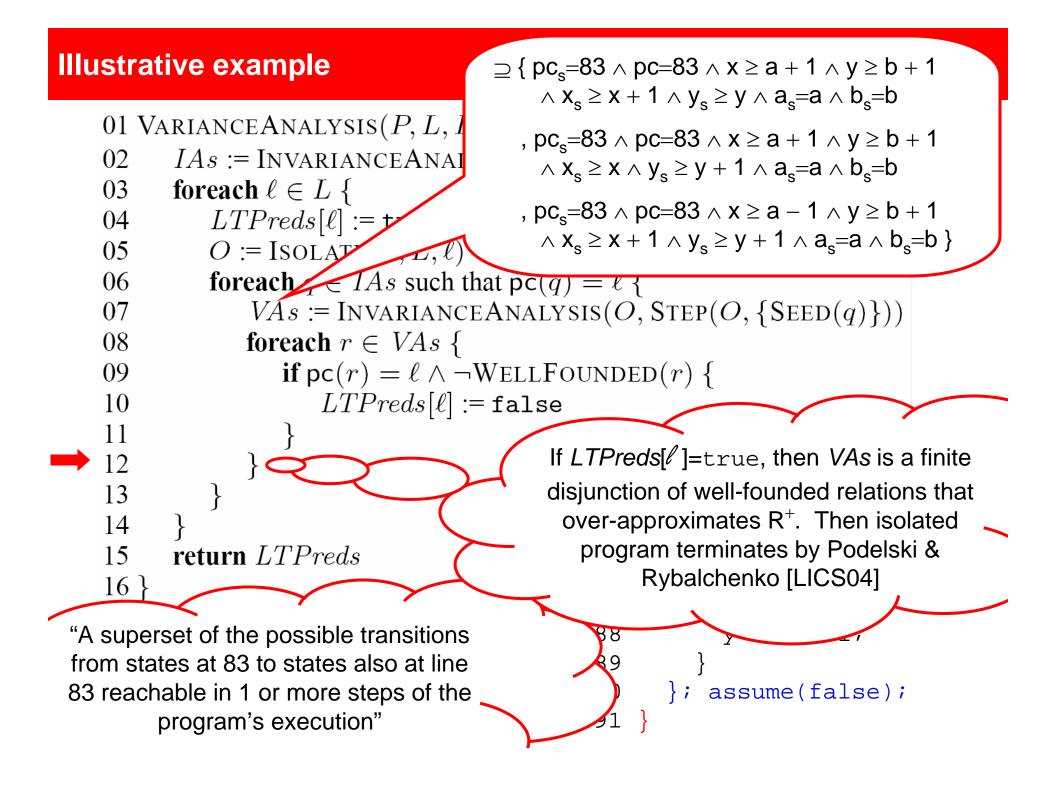






Illustrative example	$ \supseteq \{ pc_s = 83 \land pc = 83 \land x \ge a + 1 \land y \ge b + 1 \\ \land x_s \ge x + 1 \land y_s \ge y \land a_s = a \land b_s = b $
08 foreach $r \in VAs$ {	, $pc_s=83 \land pc=83 \land x \ge a + 1 \land y \ge b + 1$ $\land x_s \ge x \land y_s \ge y + 1 \land a_s=a \land b_s=b$ , $pc_s=83 \land pc=83 \land x \ge a - 1 \land y \ge b + 1$ $\land x_s \ge x + 1 \land y_s \ge y + 1 \land a_s=a \land b_s=b$ } that $pc(q) = \ell$ { $CEANALYSIS(O, STEP(O, {SEED(q)}))$ = WELLFOUNDED(r) { := false 81 while (nondet()) {
$pc_s = 83 \land pc = 84 \land x \ge a + 1 \land y \ge a + x \land x_s = x \land y_s = y \land a_s = a \land b_s = b$	
<pre>15 return LTPreds 16 }</pre>	<pre>85</pre>

Illustrative example	$ \supseteq \{ pc_s = 83 \land pc = 83 \land x \ge a + 1 \land y \ge b + 1 \\ \land x_s \ge x + 1 \land y_s \ge y \land a_s = a \land b_s = b $										
01 VARIANCEANALYSIS $(P, L, I)$ 02 $IAs :=$ INVARIANCEANAL	$, \mu c_s = 0.5 \land \mu c = 0.5 \land x \ge a + 1 \land y \ge b + 1$										
03 foreach $\ell \in L$ {	$ \land \land_{s} \leq \land \land y_{s} \leq y + 1 \land a_{s} - a \land b_{s} - b $										
04 $LTPreds[\ell] := t$	, $pc_s = 83 \land pc = 83 \land x \ge a - 1 \land y \ge b + 1$										
05 $O := ISOLAT$ $(L, \ell)$	$\land x_s \ge x + 1 \land y_s \ge y + 1 \land a_s = a \land b_s = b $										
06 <b>foreach</b> $\in TAs$ such											
-	$CEANALYSIS(O, STEP(O, \{SEED(q)\}))$										
$08 \qquad \qquad \text{foreach } r \in VAs \ \{$											
•	$\neg WELLFOUNDED(r) $										
10 $LTPreds[\ell]$											
11 }	<pre>81 while (nondet()) {</pre>										
12 }	82 while (x>a && y>b) {										
13 }	83 if (nondet()) {										
14 }	84 do {										
15 return <i>LTPreds</i>	85 $x = x - 1;$										
16 }	<pre>86 } while (nondet());</pre>										
	87 } else {										
"A superset of the possible transitions											
from states at 83 to states also at line											
83 reachable in 1 or more steps of the	e }; assume(false);										
program's execution"											





- → 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:

 $\label{eq:star} \{(s,t) \mid s(x) + s(y) \geq t(x) + t(y) - 1 \ \land \ t(x) + t(y) \geq 0 \}$  Note the 4-variable inequality.

```
81 while (nondet()) {
82
     while (x>a && y>b) {
83
       if (nondet()) {
84
         do
             ł
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)

Microsoft<sup>\*</sup>

- 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

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

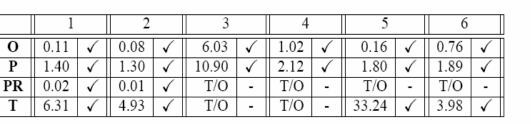
Microsoft<sup>\*</sup>

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

→ Define:

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

→ That's it!



Microsoft\*

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

	1	1 2 3		4 6				7		8		9		10		11		12				
0	0.30	t	0.05	Ť	0.11	†	0.50	t	0.10	Ť	0.17	†	0.16	†	0.12	†	0.35	†	0.86	t	0.12	†
Р	1.42	$\checkmark$	0.82	$\checkmark$	1.06	t	2.29	t	2.61	t	1.28	t	0.24	†	1.36	$\checkmark$	1.69	Ť	1.56	Ť	1.05	t
PR	0.21	$\checkmark$	0.13	$\checkmark$	0.44	$\checkmark$	1.62	$\checkmark$	3.88	$\checkmark$	0.11	$\checkmark$	2.02	$\checkmark$	1.33	$\checkmark$	13.34	$\checkmark$	174.55	$\checkmark$	0.15	$\checkmark$
Т	435.23	$\checkmark$	61.15	$\checkmark$	T/O	-	T/O	-	75.33	$\checkmark$	T/O	-	T/O	-	T/O	-	T/O	-	T/O	-	10.31	†

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

	1		2		2		3		4		5		6		7		8	9		10	
0	1.42	$\checkmark$	1.67	$\oslash$	0.47	$\oslash$	0.18	$\checkmark$	0.06	$\checkmark$	0.53	$\checkmark$	0.50	$\checkmark$	0.32	$\checkmark$	0.14	$\oslash$	0.17	$\checkmark$	
Р	4.66	$\checkmark$	6.35	$\oslash$	1.48	$\oslash$	1.10	$\checkmark$	1.30	$\checkmark$	1.60	$\checkmark$	2.65	$\checkmark$	1.89	$\checkmark$	2.42	$\oslash$	1.27	$\checkmark$	
PR	T/O	-	T/O	-	T/O	-	T/O	-	0.10	$\checkmark$	T/O	-	T/O	-	T/O	-	T/O	-	0.31	$\checkmark$	
Т	10.22	$\checkmark$	31.51	$\oslash$	20.65	$\oslash$	4.05	$\checkmark$	12.63	$\checkmark$	67.11	$\checkmark$	298.45	$\checkmark$	444.78	$\checkmark$	T/O	-	55.28	$\checkmark$	

(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



Microsoft<sup>\*</sup>

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

→ Define:

- $\rho$  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



Loop	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Time (s)	0.0	0.0	8.0	0.3	1.7	13	296	0.1	5.4	0.0	8.2	821	0.0	1.6	152	0.0	2.6	3.5	58	32	261
Result	$\checkmark$	$\oslash$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\oslash$	$\bigcirc$	$\checkmark$	$\oslash$	$\checkmark$	$\oslash$	$\checkmark$	$\checkmark$						
WF checks	1	4	16	3	5	9	15	2	4	1	6	39	1	3	16	1	28	9	85	20	37

- → 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 stateof-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