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* $\mathcal{LT}$ 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

Parameterized variance analysis algorithm

$\textsc{VarianceAnalysis}(P, L, I^\sharp)$ {

$\quad IAs := \textsc{InvarianceAnalysis}(P, I^\sharp)$

$\quad$ **foreach** $\ell \in L$ {

$\quad\quad LTPreds[\ell] := \texttt{true}$

$\quad\quad O := \textsc{Isolate}(P, L, \ell)$

$\quad\quad$ **foreach** $q \in IAs$ such that $\mathsf{pc}(q) = \ell$ {

$\quad\quad\quad VAs := \textsc{InvarianceAnalysis}(O, \textsc{Step}(O, \{\textsc{Seed}(q)\}))$

$\quad\quad\quad$ **foreach** $r \in VAs$ {

$\quad\quad\quad\quad$ **if** $\mathsf{pc}(r) = \ell \wedge \neg\textsc{WellFounded}(r)$ {

$\quad\quad\quad\quad\quad LTPreds[\ell] := \texttt{false}$

$\quad\quad\quad\quad$ }

$\quad\quad\quad$ }

$\quad\quad$ }

$\quad$ }

$\quad$ **return** $LTPreds$

}

Parameterized variance analysis algorithm

$\textsc{VarianceAnalysis}(P, L, I^\sharp)$ {
  $IAs := \textsc{InvarianceAnalysis}(P, I^\sharp)$
  foreach $\ell \in L$ {
    $LTPreds[\ell] := \texttt{true}$
    $O := \textsc{Isolate}(P, L, \ell)$
    foreach $q \in IAs$ such that $\mathsf{pc}(q) = \ell$ {
      $VAs := \textsc{InvarianceAnalysis}(O, \textsc{Step}(O, \{\textsc{Seed}(q)\}))$
      foreach $r \in VAs$ {
        if $\mathsf{pc}(r) = \ell \wedge \neg\textsc{WellFounded}(r)$ {
          $LTPreds[\ell] := \texttt{false}$
        }
      }
    }
  }
  return $LTPreds$
}

Underlying invariance analysis

Single-step version of invariance analysis

Additional operation to check progress is being made

Additional operation to plant initial representation of progress

Parameterized variance analysis algorithm

```
VarianceAnalysis(P, L, I♯) {
    IAs := InvarianceAnalysis(P, I♯)
    foreach ℓ ∈ L {
        LTPreds[ℓ] := true
        O := Isolate(P, L, ℓ)
        foreach q ∈ IAs such that pc(q) = ℓ {
            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
            foreach r ∈ VAs {
                if pc(r) = ℓ ∧ ¬WellFounded(r) {
                    LTPreds[ℓ] := false
                }
            }
        }
    }
    return LTPreds
}
```

Initial abstract state

Input program

Set of cutpoints

Output array indicating which local termination predicates were proved

Local termination predicates

```
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…

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

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…

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

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 }
```

- $\mathcal{LT}(82)$: Line 82 is visited infinitely often only when the program's execution exits the loop contained in lines 81 to 91 infinitely often ✗
- $\mathcal{LT}(83)$: Line 83 is visited infinitely often only when the program's execution exits the loop contained in lines 82 to 90 infinitely often ✓
- $\mathcal{LT}(85)$: Line 85 is visited infinitely often only when the program's execution exits the loop contained in lines 84 to 86 infinitely often ✗

Illustrative example

- 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: $pc = c$

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

Illustrative example

Microsoft Research Cambridge

```
VarianceAnalysis(P, L, I♯) {
    IAs := InvarianceAnalysis(P, I♯)
    foreach ℓ ∈ L {
        LTPreds[ℓ] := true
        O := Isolate(P, L, ℓ)
        foreach q ∈ IAs such that pc(q) = ℓ {
            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
            foreach r ∈ VAs {
                if pc(r) = ℓ ∧ ¬WellFounded(r) {
                    LTPreds[ℓ] := false
                }
            }
        }
    }
    return LTPreds
}
```

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

Illustrative example

```
VarianceAnalysis(P, L, I♯) {
    IAs := InvarianceAnalysis(P, I♯)
    foreach ℓ ∈ L {
        LTPreds[ℓ] := true
        O := Isolate(P, L, ℓ)
        foreach q ∈ IAs such that pc(q) = ℓ {
            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
            foreach r ∈ VAs {
                if pc(r) = ℓ ∧ ¬WellFounded(r) {
                    LTPreds[ℓ] := false
                }
            }
        }
    }
    return LTPreds
}
```

pc=81 $\wedge$ x $\geq$ a + 1 $\wedge$ y $\geq$ b + 1

```
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 }
```

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(
07             VAs := InvarianceAnal
08             foreach r ∈ VAs {
09                 if pc(r) = ℓ ∧ ¬WellFounde
10                     LTPreds[ℓ] := false
11                 }
12             }
13         }
14     }
15     return LTPreds
16 }
```

83

pc=81 ∧ x ≥ a + 1 ∧ y ≥ b + 1

pc=83 ∧ x ≥ a + 1 ∧ y ≥ b + 1

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

```
81 wh   e (nondet()) {
82     hile (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 }
```

Illustrative example

```
VarianceAnalysis(P, L, I♯) {
    IAs := InvarianceAnalysis(P, I♯)
    foreach ℓ ∈ L {
        LTPreds[ℓ] := true
        O := Isolate(P, L, ℓ)
        foreach q ∈ IAs such that pc(q) = ℓ {
            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
            foreach r ∈ VAs {
                if pc(r) = ℓ ∧ ¬WellFounded(r) {
                    LTPreds[ℓ] := false
                }
            }
        }
    }
    return LTPreds
}
```

$pc=83 \wedge x \geq a+1 \wedge y \geq b+1$

```
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);
91 }
```

Illustrative example

```
VarianceAnalysis(P, L, I♯) {
    IAs := InvarianceAnalysis(P, I♯)
    foreach ℓ ∈ L {
        LTPreds[ℓ] := true
        O := Isolate(P, L, ℓ)
        foreach q ∈ IAs such that pc(q) = ℓ {
            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
            foreach r ∈ VAs {
                if pc(r) = ℓ ∧ ¬WellFounded(r) {
                    LTPreds[ℓ] := false
                }
            }
        }
    }
    return LTPreds
}
```

pc=83 $\wedge$ x $\geq$ a + 1 $\wedge$ y $\geq$ b + 1

```
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);
91 }
```

Illustrative example

```
VarianceAnalysis(P, L, I♯) {
    IAs := InvarianceAnalysis(P, I♯)
    foreach ℓ ∈ L {
        LTPreds[ℓ] := true
        O := Isolate(P, L, ℓ)
        foreach q ∈ IAs such that pc(q) = ℓ {
            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
            foreach r ∈ VAs {
                if pc(r) = ℓ ∧ ¬Well
                    LTPreds[ℓ] := fal
                }
            }
        }
    }
    return LTPreds
}
```

$pc=83 \wedge x \geq a+1 \wedge y \geq b+1$

$pc=83 \wedge x \geq a+1 \wedge y \geq b+1$
$\wedge\ pc_s=pc \wedge x_s=x \wedge y_s=y \wedge a_s=a \wedge b_s=b$

```
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

```
VarianceAnalysis(P, L, I♯) {
    IAs := InvarianceAnalysis(P, I♯)
    foreach ℓ ∈ L {
        LTPreds[ℓ] := true
        O := Isolate(P, L, ℓ)
        foreach q ∈ IAs such that pc(q) = ℓ {
            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
            foreach r ∈ VAs {
                if pc(r) = ℓ ∧ ¬WellF
                    LTPreds[ℓ] := fal
                }
            }
        }
    }
    return LTPreds
}
```

$pc=83 \wedge x \geq a+1 \wedge y \geq b+1$

$pc=83 \wedge x \geq a+1 \wedge y \geq b+1$
$\wedge\ pc_s=pc \wedge x_s=x \wedge y_s=y \wedge a_s=a \wedge b_s=b$

$\{(s,t) \mid s(pc)=t(pc)=83$
$\wedge\ s(x)=t(x)$
$\wedge\ s(y)=t(y)$
$\wedge\ s(a)=t(a)$
$\wedge\ s(b)=t(b)$
$\wedge\ t(x) \geq t(a)+1$
$\wedge\ t(y) \geq t(b)+1\ \}$

```
    }; assume(false);
}
```

Illustrative example

$\textsc{VarianceAnalysis}(P, L, I^\sharp)$ {
    $IAs := \textsc{InvarianceAnalysis}(P, I^\sharp)$
    foreach $\ell \in L$ {
        $LTPreds[\ell]$:= `true`
        $O := \textsc{Isolate}(P, L, \ell)$
        foreach $q \in IAs$ such that $\mathsf{pc}(q) = \ell$ {
            $VAs := \textsc{InvarianceAnalysis}(O, \textsc{Step}(O, \{\textsc{Seed}(q)\}))$
            foreach $r \in VAs$ {
                if $\mathsf{pc}(r) = \ell \wedge \neg\textsc{Wel}$
                    $LTPreds[\ell]$:=
                }

$\mathsf{pc}_s{=}83 \wedge \mathsf{pc}{=}84 \wedge x \geq a + 1 \wedge y \geq b + 1 \wedge x_s{=}x \wedge y_s{=}y \wedge a_s{=}a \wedge b_s{=}b$

    return $LTPreds$
}

$\mathsf{pc}{=}83 \wedge x \geq a + 1 \wedge y \geq b + 1$

$\mathsf{pc}{=}83 \wedge x \geq a + 1 \wedge y \geq b + 1 \wedge \mathsf{pc}_s{=}\mathsf{pc} \wedge x_s{=}x \wedge y_s{=}y \wedge a_s{=}a \wedge b_s{=}b$

```
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                 }
15     return LTPreds
16 }
```

$pc_s=83 \wedge pc=84 \wedge x \geq a+1 \wedge y \geq b+1$
$\wedge\ x_s=x \wedge y_s=y \wedge a_s=a \wedge b_s=b$

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

Illustrative example

$\supseteq \{\, pc_s{=}83 \wedge pc{=}83 \wedge x \geq a+1 \wedge y \geq b+1$
$\wedge\, x_s \geq x+1 \wedge y_s \geq y \wedge a_s{=}a \wedge b_s{=}b$

$,\, pc_s{=}83 \wedge pc{=}83 \wedge x \geq a+1 \wedge y \geq b+1$
$\wedge\, x_s \geq x \wedge y_s \geq y+1 \wedge a_s{=}a \wedge b_s{=}b$

$,\, pc_s{=}83 \wedge pc{=}83 \wedge x \geq a-1 \wedge y \geq b+1$
$\wedge\, x_s \geq x+1 \wedge y_s \geq y+1 \wedge a_s{=}a \wedge b_s{=}b \,\}$

```
VarianceAnalysis(P, L, I
    IAs := InvarianceAnal
    foreach ℓ ∈ L {
        LTPreds[ℓ] := t
        O := Isolat      , L, ℓ)
        foreach q ∈ IAs such that pc(q) = ℓ {
            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
            foreach r ∈ VAs {
                if pc(r) = ℓ ∧ ¬WellFounded(r) {
                    LTPreds[ℓ] := false
                }
    return LTPreds
}
```

$pc_s{=}83 \wedge pc{=}84 \wedge x \geq a+1 \wedge y \geq b+1$
$\wedge\, x_s{=}x \wedge y_s{=}y \wedge a_s{=}a \wedge b_s{=}b$

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

Illustrative example

$\supseteq$ { pc_s=83 $\wedge$ pc=83 $\wedge$ x $\geq$ a + 1 $\wedge$ y $\geq$ b + 1 $\wedge$ $x_s \geq$ x + 1 $\wedge$ $y_s \geq$ y $\wedge$ a_s=a $\wedge$ b_s=b

, pc_s=83 $\wedge$ pc=83 $\wedge$ x $\geq$ a + 1 $\wedge$ y $\geq$ b + 1 $\wedge$ $x_s \geq$ x $\wedge$ $y_s \geq$ y + 1 $\wedge$ a_s=a $\wedge$ b_s=b

, pc_s=83 $\wedge$ pc=83 $\wedge$ x $\geq$ a – 1 $\wedge$ y $\geq$ b + 1 $\wedge$ $x_s \geq$ x + 1 $\wedge$ $y_s \geq$ y + 1 $\wedge$ a_s=a $\wedge$ b_s=b }

```
VarianceAnalysis(P, L, I
    IAs := InvarianceAnal
    foreach ℓ ∈ L {
        LTPreds[ℓ] := t
        O := Isolat        , L, ℓ)
        foreach q ∈ IAs such that pc(q) = ℓ {
            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
            foreach r ∈ VAs {
                if pc(r) = ℓ ∧ ¬WellFounded(r) {
                    LTPreds[ℓ] := false
                }
            }
        }
    }
    return LTPreds
}
```

```
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);
91 }
```

“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”

Illustrative example

$\supseteq \{\, pc_s{=}83 \wedge pc{=}83 \wedge x \geq a+1 \wedge y \geq b+1 \wedge x_s \geq x+1 \wedge y_s \geq y \wedge a_s{=}a \wedge b_s{=}b$

$,\ pc_s{=}83 \wedge pc{=}83 \wedge x \geq a+1 \wedge y \geq b+1 \wedge x_s \geq x \wedge y_s \geq y+1 \wedge a_s{=}a \wedge b_s{=}b$

$,\ pc_s{=}83 \wedge pc{=}83 \wedge x \geq a-1 \wedge y \geq b+1 \wedge x_s \geq x+1 \wedge y_s \geq y+1 \wedge a_s{=}a \wedge b_s{=}b \,\}$

```
VarianceAnalysis(P, L, I
    IAs := InvarianceAnal
    foreach ℓ ∈ L {
        LTPreds[ℓ] := t
        O := Isolat        , L, ℓ)
        foreach   ∈ IAs such that pc(q) = ℓ {
            VAs := InvarianceAnalysis(O, Step(O, {Seed(q)}))
            foreach r ∈ VAs {
                if pc(r) = ℓ ∧ ¬WellFounded(r) {
                    LTPreds[ℓ] := false
                }
            }
        }
    }
    return LTPreds
}
```

If *LTPreds*[ℓ]=`true`, then *VAs* is a finite disjunction of well-founded relations that over-approximates R^+. Then isolated program terminates by Podelski & Rybalchenko [LICS04]

"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"

```
88          y
89      }
    }; assume(false);
91 }
```

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:

$$\{(s,t) \mid s(x)+s(y) \geq t(x)+t(y)-1 \wedge t(x)+t(y) \geq 0\}$$

Note the 4-variable inequality.

```
while (nondet()) {
  while (x>a && y>b) {
    if (nondet()) {
      do {
        x = x - 1;
      } while (x>10);
    } else {
      y = y - 1;
    }
  }
}
```

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:

$$\begin{array}{lcl} \textsc{Seed}(F) & \triangleq & F \wedge \bigwedge_{v \in \mathrm{PVar}} \{v = \rho(v)\} \\ \textsc{WellFounded}(F) & \triangleq & \textsc{WfCheck}(\rho(\mathrm{PVar}), \mathrm{PVar}, F) \end{array}$$

  - ρ 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

	1		2		3		4		5		6	
O	0.11	✓	0.08	✓	6.03	✓	1.02	✓	0.16	✓	0.76	✓
P	1.40	✓	1.30	✓	10.90	✓	2.12	✓	1.80	✓	1.89	✓
PR	0.02	✓	0.01	✓	T/O	-	T/O	-	T/O	-	T/O	-
T	6.31	✓	4.93	✓	T/O	-	T/O	-	33.24	✓	3.98	✓

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

	1		2		3		4		6		7		8		9		10		11		12	
O	0.30	†	0.05	†	0.11	†	0.50	†	0.10	†	0.17	†	0.16	†	0.12	†	0.35	†	0.86	†	0.12	†
P	1.42	✓	0.82	✓	1.06	†	2.29	†	2.61	†	1.28	†	0.24	†	1.36	✓	1.69	†	1.56	†	1.05	†
PR	0.21	✓	0.13	✓	0.44	✓	1.62	✓	3.88	✓	0.11	✓	2.02	✓	1.33	✓	13.34	✓	174.55	✓	0.15	✓
T	435.23	✓	61.15	✓	T/O	-	T/O	-	75.33	✓	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		3		4		5		6		7		8		9		10	
O	1.42	✓	1.67	⊘	0.47	⊘	0.18	✓	0.06	✓	0.53	✓	0.50	✓	0.32	✓	0.14	⊘	0.17	✓
P	4.66	✓	6.35	⊘	1.48	⊘	1.10	✓	1.30	✓	1.60	✓	2.65	✓	1.89	✓	2.42	⊘	1.27	✓
PR	T/O	-	T/O	-	T/O	-	T/O	-	0.10	✓	T/O	-	T/O	-	T/O	-	T/O	-	0.31	✓
T	10.22	✓	31.51	⊘	20.65	⊘	4.05	✓	12.63	✓	67.11	✓	298.45	✓	444.78	✓	T/O	-	55.28	✓

(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:

$$
\begin{array}{lcl}
\textsc{Seed}(\Pi \wedge \Sigma) & \triangleq & (\Pi \wedge \Sigma \wedge \bigwedge_{v \in \mathrm{fDV}(\Pi \wedge \Sigma)} \{v = \rho(v)\}) \\
\textsc{Seed}(\top) & \triangleq & \top
\end{array}
$$

$$
\begin{array}{lcl}
\textsc{WellFounded}(\Pi \wedge \Sigma) & \triangleq & \textsc{WfCheck}(\rho(\mathrm{DVar}), \mathrm{DVar}, \Pi) \\
\textsc{WellFounded}(\top) & \triangleq & \mathsf{false}
\end{array}
$$

  - ρ 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

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	✓	⊘	✓	✓	✓	✓	✓	⊘	⊘	✓	✓	✓	✓	✓	✓	✓	⊘	✓	⊘	✓	✓
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 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