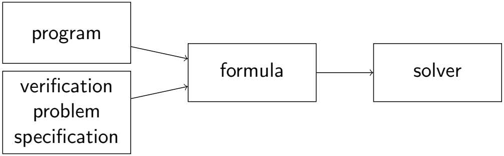
Challenges in Decomposing Encodings of Verification Problems

Peter Schrammel

HCVS 2016
Eindhoven, The Netherlands

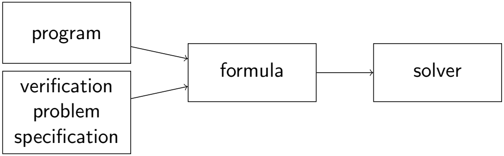
Motivation

Modern software verification tools:



Motivation

Modern software verification tools:

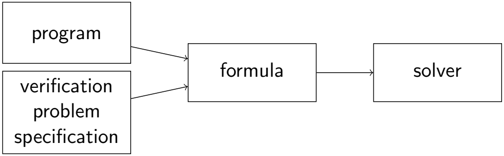


Large programs:

- Problem: Formula too large for existing backend solvers

Motivation

Modern software verification tools:

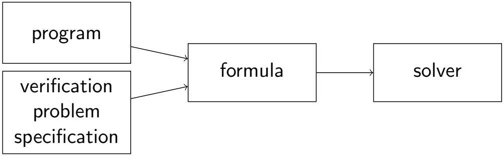


Large programs:

- Problem: Formula too large for existing backend solvers
- Solution: Make formula smaller

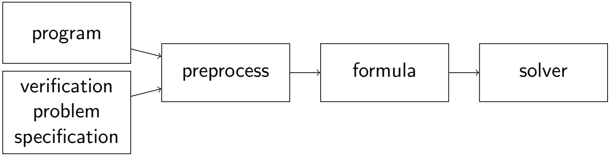
Motivation

Modern software verification tools:



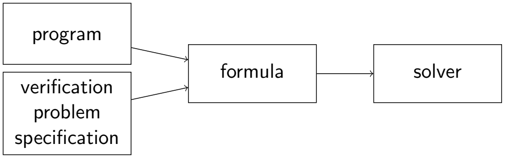
Large programs:

- Problem: Formula too large for existing backend solvers
- Solution: Make formula smaller



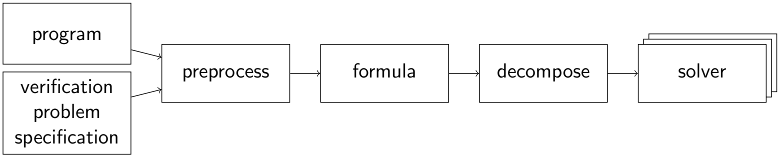
Motivation

Modern software verification tools:



Large programs:

- Problem: Formula too large for existing backend solvers
- Solution: Make formula smaller



Case Studies: Termination Analyses (ASE'15)

Universal termination:

- Result: terminating / potentially non-term. / non-terminating
- Decision problem

Conditional termination:

- Result: sufficient precondition for termination
- Inference problem

Case Studies: Termination Analyses (ASE'15)

Universal termination:

- Result: terminating / potentially non-term. / non-terminating
- Decision problem

Conditional termination:

- Result: sufficient precondition for termination
- Inference problem

Example: Universal Termination Analysis

Encoding of the modular universal termination problem:

$$
\begin{array}{l}
\exists_2 Summary_{f_1}, \ldots, Summary_{f_n} : \bigwedge_{f \in F} \\
\quad \exists_2 Inv_f, RR_f : \forall \mathbf{x}^{in}{}_f, \mathbf{x}_f, \mathbf{x}'{}_f, \mathbf{x}^{out}{}_f : \\
\qquad Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \Longrightarrow Inv_f(\mathbf{x}_f) \\
\quad \wedge \quad Inv_f(\mathbf{x}_f) \wedge Trans_f(\mathbf{x}_f, \mathbf{x}'{}_f) \wedge \bigwedge_{h_i \in H_f} Summary_h(\mathbf{x}^{p\_in}{}_{h_i}, \mathbf{x}^{p\_out}{}_{h_i}) \\
\qquad\qquad \Longrightarrow Inv_f(\mathbf{x}'{}_f) \wedge RR_f(\mathbf{x}_f, \mathbf{x}'{}_f) \\
\quad \wedge \quad Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \wedge Inv_f(\mathbf{x}'{}_f) \wedge Out_f(\mathbf{x}'{}_f, \mathbf{x}^{out}{}_f) \\
\qquad\qquad \Longrightarrow Summary_f(\mathbf{x}^{in}{}_f, \mathbf{x}^{out}{}_f)
\end{array}
$$

Decomposition:

- Procedural, top-down, context-sensitive

Example: Universal Termination Analysis

Benchmarks:

- Product line benchmarks from SV-COMP (597 benchmarks, 1.6 MLOC)
- Non-trivial procedural structure (on average 67 procedures, 5.5 loops)

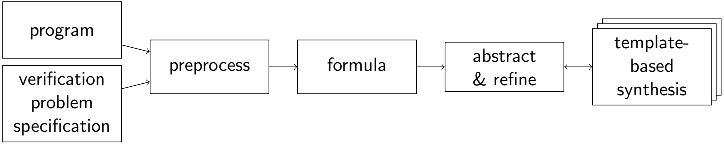
Results:

	expected	**2LS IPTA**	2LS MTA	TAN	Ultimate
terminating	264	**249**	26	18	50
non-terminating	333	**320**	333	3	324
potentially non-terminating	—	**14**	1	425	0
timed out (0.5h)	—	**14**	237	150	43
errors	—	0	0	1	180
total run time (h)	—	**58.7**	119.6	92.8	23.9

2LS for Program Analysis

http://www.cprover.org/2LS

- Verification and static analysis on logical formulae
- Approximates solution to 2OL by reduction to FOL



- Bit-precise analysis
  (including floating-point arithmetic)

SV-COMP'16 

- Template-based synthesis
  (using strategy iteration for optimisation)

Analysis features:

- Interprocedural static analysis, termination analysis
- Incremental BMC, k-induction, kIkI (SAS'15)

Logical Specification of Verification Problems

Safety verification:

$$\begin{aligned} \exists_2 Inv.\quad & \forall \mathbf{x}^{in}, \mathbf{x}, \mathbf{x}'. \\ & \big(Init(\mathbf{x}^{in}, \mathbf{x}) \Longrightarrow Inv(\mathbf{x})\big) \wedge \\ & (Inv(\mathbf{x}) \wedge Trans(\mathbf{x}, \mathbf{x}') \Longrightarrow Inv(\mathbf{x}')) \wedge \\ & (Inv(\mathbf{x}) \Longrightarrow \neg Err(\mathbf{x})) \end{aligned}$$

(Blass and Gurevich '87, Grebenshchikov et al '12, David et al '15, ...)

Logical Specification of Verification Problems

Safety verification:

$$\begin{aligned}\exists_2 Inv.\quad & \forall \mathbf{x}^{in}, \mathbf{x}, \mathbf{x}'. \\ & \big(Init(\mathbf{x}^{in}, \mathbf{x}) \Longrightarrow Inv(\mathbf{x})\big) \wedge \\ & \big(Inv(\mathbf{x}) \wedge Trans(\mathbf{x}, \mathbf{x}') \Longrightarrow Inv(\mathbf{x}')\big) \wedge \\ & \big(Inv(\mathbf{x}) \Longrightarrow \neg Err(\mathbf{x})\big)\end{aligned}$$

Invariant inference:

$$\begin{aligned}\min\ Inv.\quad & \forall \mathbf{x}, \mathbf{x}'. \\ & \big(Init(\mathbf{x}) \Longrightarrow Inv(\mathbf{x})\big) \wedge \\ & \big(Inv(\mathbf{x}) \wedge Trans(\mathbf{x}, \mathbf{x}') \Longrightarrow Inv(\mathbf{x}')\big)\end{aligned}$$

(Blass and Gurevich '87, Grebenshchikov et al '12, David et al '15, ...)

Logical Specification of Verification Problems

Safety verification:

$$\exists_2 Inv.\quad \forall \mathbf{x}^{in}, \mathbf{x}, \mathbf{x}'.\quad \big(Init(\mathbf{x}^{in}, \mathbf{x}) \Longrightarrow Inv(\mathbf{x})\big) \wedge (Inv(\mathbf{x}) \wedge Trans(\mathbf{x}, \mathbf{x}') \Longrightarrow Inv(\mathbf{x}')) \wedge (Inv(\mathbf{x}) \Longrightarrow \neg Err(\mathbf{x}))$$

Invariant inference:

$$\min Inv.\quad \forall \mathbf{x}, \mathbf{x}'.\quad (Init(\mathbf{x}) \Longrightarrow Inv(\mathbf{x})) \wedge (Inv(\mathbf{x}) \wedge Trans(\mathbf{x}, \mathbf{x}') \Longrightarrow Inv(\mathbf{x}'))$$

Termination verification:

$$\exists_2 RR, Inv.\quad \forall \mathbf{x}, \mathbf{x}'.\quad (Init(\mathbf{x}) \Longrightarrow Inv(\mathbf{x})) \wedge \big(Inv(\mathbf{x}) \wedge Trans(\mathbf{x}, \mathbf{x}') \Longrightarrow Inv(\mathbf{x}') \wedge RR(\mathbf{x}, \mathbf{x}')\big)$$

...

(Blass and Gurevich '87, Grebenshchikov et al '12, David et al '15, ...)

Template-Based Synthesis

Reduction to first-order logic via templates, e.g. safety verification:

$$\begin{aligned}\exists_2 Inv.\forall \mathbf{x}, \mathbf{x}' 1.\quad & (Init(\mathbf{x}) \Longrightarrow Inv(\mathbf{x})) \wedge \\ & (Inv(\mathbf{x}) \wedge Trans(\mathbf{x}, \mathbf{x}') \Longrightarrow Inv(\mathbf{x}')) \wedge \\ & (Inv(\mathbf{x}) \Longrightarrow \neg Err(\mathbf{x}))\end{aligned}$$

Template-Based Synthesis

Reduction to first-order logic via templates, e.g. safety verification:

$$\begin{aligned}\exists \mathbf{d}.\quad \forall \mathbf{x}, \mathbf{x}'.\quad & (\mathit{Init}(\mathbf{x}) \Longrightarrow \mathcal{T}(\mathbf{x}, \mathbf{d})) \wedge \\ & (\mathcal{T}(\mathbf{x}, \mathbf{d}) \wedge \mathit{Trans}(\mathbf{x}, \mathbf{x}') \Longrightarrow \mathcal{T}(\mathbf{x}', \mathbf{d})) \wedge \\ & (\mathcal{T}(\mathbf{x}, \mathbf{d}) \Longrightarrow \neg \mathit{Err}(\mathbf{x}))\end{aligned}$$

where **d** are template parameters.

(Graf & Saïdi CAV'97, . . . , Reps et al, . . . Brauer et al, . . . , Srivastava et al, . . .)

Template-Based Synthesis

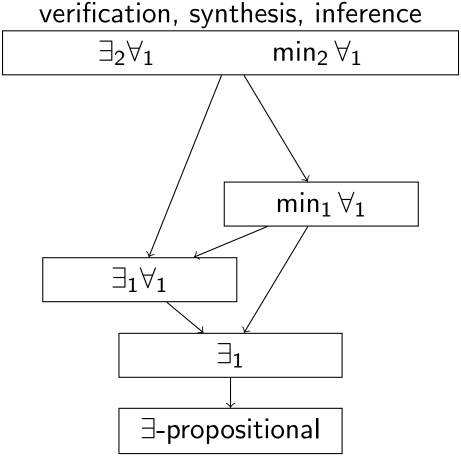
Reduction to first-order logic via templates, e.g. invariant inference:

$$\min \mathbf{d}. \quad \forall \mathbf{x}, \mathbf{x}'. \quad \begin{array}{l} (\mathit{Init}(\mathbf{x}) \Longrightarrow \mathcal{T}(\mathbf{x}, \mathbf{d})) \wedge \\ (\mathcal{T}(\mathbf{x}, \mathbf{d}) \wedge \mathit{Trans}(\mathbf{x}, \mathbf{x}') \Longrightarrow \mathcal{T}(\mathbf{x}', \mathbf{d})) \end{array}$$

where **d** are template parameters.

(Graf & Saïdi CAV'97, . . . , Reps et al, . . . Brauer et al, . . . , Srivastava et al, . . .)

Solver Hierarchy



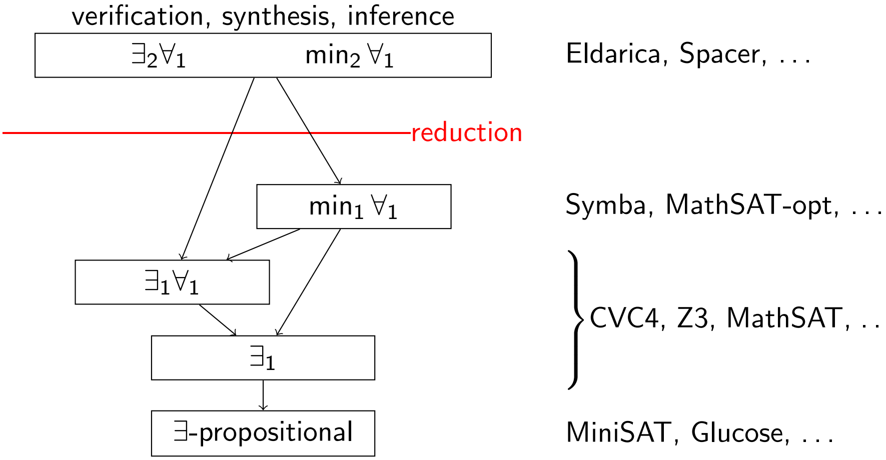
Eldarica, Spacer, . . .

Symba, MathSAT-opt, . . .

CVC4, Z3, MathSAT, . .

MiniSAT, Glucose, . . .

Solver Hierarchy



Program Encoding

Non-recursive programs with multiple procedures

Procedure f: $\left(\quad Init(\mathbf{x}^{in}, \mathbf{x}) \quad , \quad Trans(\mathbf{x}, \mathbf{x}') \quad , \quad Out(\mathbf{x}, \mathbf{x}^{out}) \quad \right)$

Program Encoding

Non-recursive programs with multiple procedures

Procedure f: ($Init(\mathbf{x}^{in}, \mathbf{x})$, $Trans(\mathbf{x}, \mathbf{x}')$, $Out(\mathbf{x}, \mathbf{x}^{out})$)

```
unsigned f(unsigned z) {
  unsigned w = 0;
  if(z>0)
    w = h(z);

  return w;
}
```

$$
\begin{aligned}
& w_0 = 0 \\
\wedge\ & g_4 = z > 0 \\
\wedge\ & h_0(z, r_{h_0}) \wedge w_1 = r_{h_0} \\
\wedge\ & w_2^{\phi} = g_4 ? w_1 : w_0 \\
\wedge\ & r_h = x_1^{\phi}
\end{aligned}
$$

```
unsigned h(unsigned y) {
  unsigned x;
  for (x=0;

       x<10;
       x+=y);
  return x;
}
```

$$
\begin{aligned}
& g_0 = true \\
\wedge\ & x_0 = 0 \\
\wedge\ & g_1 = g_0 \wedge x_1^{\phi} = (ls_3 ? x_3^{lb} : x_0) \\
\wedge\ & g_2 = (x_1^{\phi} < 10 \wedge g_1) \\
\wedge\ & x_2 = x_1^{\phi} + y \\
\wedge\ & r_h = x_1^{\phi}
\end{aligned}
$$

Program Encoding

Non-recursive programs with multiple procedures

Procedure f: $\left(\quad Init(\mathbf{x}^{in}, \mathbf{x}) \quad , \quad Trans(\mathbf{x}, \mathbf{x}') \quad , \quad Out(\mathbf{x}, \mathbf{x}^{out}) \quad \right)$

```
unsigned f(unsigned z) {
  unsigned w = 0;
  if(z>0)
    w = h(z);

  return w;
}
```

$$
\begin{array}{ll}
 & w_0 = 0 \\
\wedge & g_4 = z > 0 \\
\wedge & h_0(z, r_{h_0}) \wedge w_1 = r_{h_0} \\
\wedge & w_2^{\phi} = g_4 ? w_1 : w_0 \\
\wedge & r_h = x_1^{\phi}
\end{array}
$$

```
unsigned h(unsigned y) {
  unsigned x;
  for (x=0;

      x<10;
      x+=y);
  return x;
}
```

$$
\begin{array}{ll}
 & g_0 = true \\
\wedge & x_0 = 0 \\
\wedge & g_1 = g_0 \wedge x_1^{\phi} = (ls_3 ? x_3^{lb} : x_0) \\
\wedge & g_2 = (x_1^{\phi} < 10 \wedge g_1) \\
\wedge & x_2 = x_1^{\phi} + y \\
\wedge & r_h = x_1^{\phi}
\end{array}
$$

Example: Modular Universal Termination Problem

$$
\begin{array}{l}
\exists_2 Summary_{f_1}, \ldots, Summary_{f_n} : \bigwedge_{f \in F} \\
\quad \exists_2 Inv_f, RR_f : \forall \mathbf{x}^{in}{}_f, \mathbf{x}_f, \mathbf{x}'{}_f, \mathbf{x}^{out}{}_f : \\
\qquad Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \Longrightarrow Inv_f(\mathbf{x}_f) \\
\quad \wedge \quad Inv_f(\mathbf{x}_f) \wedge Trans_f(\mathbf{x}_f, \mathbf{x}'{}_f) \wedge \bigwedge_{h_i \in H_f} Summary_h(\mathbf{x}^{p\_in}{}_{h_i}, \mathbf{x}^{p\_out}{}_{h_i}) \\
\qquad\qquad \Longrightarrow Inv_f(\mathbf{x}'{}_f) \wedge RR_f(\mathbf{x}_f, \mathbf{x}'{}_f) \\
\quad \wedge \quad Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \wedge Inv_f(\mathbf{x}'{}_f) \wedge Out_f(\mathbf{x}'{}_f, \mathbf{x}^{out}{}_f) \\
\qquad\qquad \Longrightarrow Summary_f(\mathbf{x}^{in}{}_f, \mathbf{x}^{out}{}_f)
\end{array}
$$

Decomposition into a sequence of subproblems

- Classical approach: Follow the call graph top-down

Decomposition

Soundness of the decomposition by

- Soundness of the individual subproblems
- Soundness of the combination of the subproblem results
- Induction over the recursive decomposition algorithm

Decomposition

Soundness of the decomposition by

- Soundness of the individual subproblems
  (e.g. calling contexts, summaries, ...)
- Soundness of the combination of the subproblem results
  (e.g. joins, fixed points, upwards propagation, ...)
- Induction over the recursive decomposition algorithm
  (e.g. call graph traversal)

Decomposition

Soundness of the decomposition by

- Soundness of the individual subproblems
  (e.g. calling contexts, summaries, ...)
- Soundness of the combination of the subproblem results
  (e.g. joins, fixed points, upwards propagation, ...)
- Induction over the recursive decomposition algorithm
  (e.g. call graph traversal)

Problem

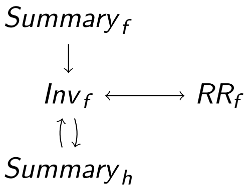
- Cyclically dependent predicates

Example: Universal Termination

$$
\begin{aligned}
&\exists_2 Summary_{f_1}, \ldots, Summary_{f_n} : \bigwedge_{f \in F} \\
&\quad \exists_2 Inv_f, RR_f : \forall \mathbf{x}^{in}{}_f, \mathbf{x}_f, \mathbf{x}'{}_f, \mathbf{x}^{out}{}_f : \\
&\qquad Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \Longrightarrow Inv_f(\mathbf{x}_f) \\
&\quad \wedge \quad Inv_f(\mathbf{x}_f) \wedge Trans_f(\mathbf{x}_f, \mathbf{x}'{}_f) \wedge \bigwedge_{h_i \in H_f} Summary_h(\mathbf{x}^{p\_in}{}_{h_i}, \mathbf{x}^{p\_out}{}_{h_i}) \\
&\qquad\qquad \Longrightarrow Inv_f(\mathbf{x}'{}_f) \wedge RR_f(\mathbf{x}_f, \mathbf{x}'{}_f) \\
&\quad \wedge \quad Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \wedge Inv_f(\mathbf{x}'{}_f) \wedge Out_f(\mathbf{x}'{}_f, \mathbf{x}^{out}{}_f) \\
&\qquad\qquad \Longrightarrow Summary_f(\mathbf{x}^{in}{}_f, \mathbf{x}^{out}{}_f)
\end{aligned}
$$

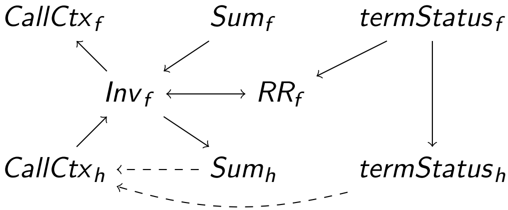
Example: Universal Termination

Cyclically dependent predicates:



Example: Universal Termination

Cyclically dependent predicates:



Example: Universal Termination

$$
\begin{array}{l}
\exists_2 Summary_{f_1}, \ldots, Summary_{f_n} : \bigwedge_{f\in F} \\
\quad \exists_2 Inv_f, RR_f : \forall \mathbf{x}^{in}{}_f, \mathbf{x}_f, \mathbf{x}'{}_f, \mathbf{x}^{out}{}_f : \\
\qquad Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \Longrightarrow Inv_f(\mathbf{x}_f) \\
\quad \wedge \quad Inv_f(\mathbf{x}_f) \wedge Trans_f(\mathbf{x}_f, \mathbf{x}'{}_f) \wedge \bigwedge_{h_i\in H_f} Summary_h(\mathbf{x}^{p\_in}{}_{h_i}, \mathbf{x}^{p\_out}{}_{h_i}) \\
\qquad\qquad \Longrightarrow Inv_f(\mathbf{x}'{}_f) \wedge RR_f(\mathbf{x}_f, \mathbf{x}'{}_f) \\
\quad \wedge \quad Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \wedge Inv_f(\mathbf{x}'{}_f) \wedge Out_f(\mathbf{x}'{}_f, \mathbf{x}^{out}{}_f) \\
\qquad\qquad \Longrightarrow Summary_f(\mathbf{x}^{in}{}_f, \mathbf{x}^{out}{}_f)
\end{array}
$$

Example: Universal Termination

Summary (callee's perspective):

$\exists_2 Summary_{f_1}, \ldots, Summary_{f_n} : \bigwedge_{f \in F}$

For each f : $\exists_2 Sum_f$:

$$
\begin{array}{ll}
& \exists_2 Inv_f, RR_f : \forall \mathbf{x}^{in}{}_f, \mathbf{x}_f, \mathbf{x}'_f, \mathbf{x}^{out}{}_f : \\
& \qquad\qquad Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \Longrightarrow Inv_f(\mathbf{x}_f) \\
\wedge & \quad Inv_f(\mathbf{x}_f) \wedge Trans_f(\mathbf{x}_f, \mathbf{x}'_f) \wedge \bigwedge_{h_i \in H_f} Sum_h(\mathbf{x}^{p\_in}{}_{h_i}, \mathbf{x}^{p\_out}{}_{h_i}) \\
& \qquad \Longrightarrow Inv_f(\mathbf{x}'_f) \wedge RR_f(\mathbf{x}_f, \mathbf{x}'_f) \\
\wedge & \qquad\qquad Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \wedge Inv_f(\mathbf{x}'_f) \wedge Out_f(\mathbf{x}'_f, \mathbf{x}^{out}{}_f) \\
& \qquad \Longrightarrow Sum_f(\mathbf{x}^{in}{}_f, \mathbf{x}^{out}{}_f)
\end{array}
$$

Example: Universal Termination

Summary (callee's perspective):

$\exists_2 Summary_{f_1}, \ldots, Summary_{f_n} : \bigwedge_{f \in F}$

For each f : Given $CallCtx_f$: $\exists_2 Sum_f$:

$$\begin{aligned}
&\exists_2 Inv_f, RR_f : \forall \mathbf{x}^{in}{}_f, \mathbf{x}_f, \mathbf{x}'{}_f, \mathbf{x}^{out}{}_f : \\
&\qquad CallCtx_f(\mathbf{x}^{in}{}_f, \mathbf{x}^{out}{}_f) \wedge Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \Longrightarrow Inv_f(\mathbf{x}_f) \\
&\wedge \quad Inv_f(\mathbf{x}_f) \wedge Trans_f(\mathbf{x}_f, \mathbf{x}'{}_f) \wedge \bigwedge_{h_i \in H_f} Sum_h(\mathbf{x}^{p_in}{}_{h_i}, \mathbf{x}^{p_out}{}_{h_i}) \\
&\qquad\qquad \Longrightarrow Inv_f(\mathbf{x}'{}_f) \wedge RR_f(\mathbf{x}_f, \mathbf{x}'{}_f) \\
&\wedge \quad CallCtx_f(\mathbf{x}^{in}{}_f, \mathbf{x}^{out}{}_f) \wedge Init_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \wedge Inv_f(\mathbf{x}'{}_f) \wedge Out_f(\mathbf{x}'{}_f, \mathbf{x}^{out}{}_f) \\
&\qquad\qquad \Longrightarrow Sum_f(\mathbf{x}^{in}{}_f, \mathbf{x}^{out}{}_f)
\end{aligned}$$

Example: Universal Termination

Calling context (caller's perspective):

$$\text{For each } f : \text{Given } \mathit{CallCtx}_f : \text{For each } h_j \in H_f : \exists_2 \mathit{CallCtx}_{h_j} :$$

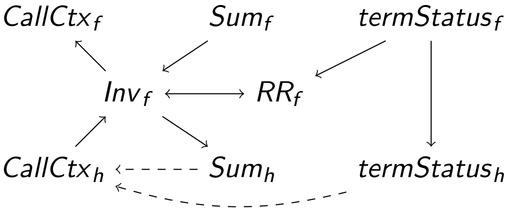
$$\exists_2 \mathit{Inv}_f : \forall \mathbf{x}^{in}{}_f, \mathbf{x}_f, \mathbf{x}'{}_f, \mathbf{x}^{out}{}_f :$$

$$\mathit{CallCtx}_f(\mathbf{x}^{in}{}_f, \mathbf{x}^{out}{}_f) \wedge \mathit{Init}_f(\mathbf{x}^{in}{}_f, \mathbf{x}_f) \Longrightarrow \mathit{Inv}_f(\mathbf{x}_f)$$

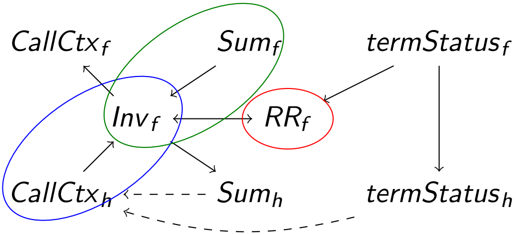
$$\wedge \quad \mathit{Inv}_f(\mathbf{x}_f) \wedge \mathit{Trans}_f(\mathbf{x}_f, \mathbf{x}'{}_f) \wedge \bigwedge_{h_i \in H_f} \mathit{Sum}_h(\mathbf{x}^{p\_in}{}_{h_i}, \mathbf{x}^{p\_out}{}_{h_i})$$

$$\Longrightarrow \mathit{Inv}_f(\mathbf{x}'{}_f) \wedge \mathit{CallCtx}_{h_j}(\mathbf{x}^{p\_in}{}_{h_j}, \mathbf{x}^{p\_out}{}_{h_j})$$

Example: Universal Termination



Example: Universal Termination



1. Calling contexts of procedure calls h in f
2. Recurse
3. Invariants and summary of procedure f
4. Termination argument for procedure f
5. Determine termination status of f

Example 2: Conditional Termination

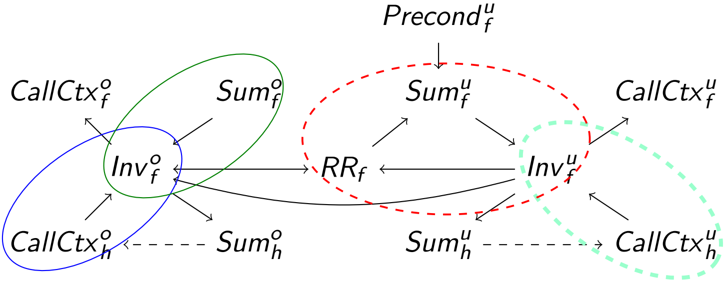
Universal termination:

- Result: terminating / potentially non-term. / non-terminating
- Decision problem

Conditional termination:

- Result: sufficient precondition for termination
- Inference problem

Sufficient Preconditions for Termination



over-approximations under-approximations

Lessons Learned

2LS for program analysis `http://www.cprover.org/2LS`

- Modular analyses and verification algorithms as decompositions of large formulae
- Predicate inference (abstract interpretation, synthesis, optimisation)
- Should expose solver interface

Observations:

- Decomposition increases scalability (smaller formulae, parallelise)
- Decomposition introduces abstraction
- Decomposition does not eliminate fixed points
- Subproblems of decision problems may be inference problems
- Syntax is a bad criterion for decomposition.

Lessons Learned

Criteria for decomposition?

- Syntax
  $\rightarrow$ eliminate syntactic bias (control where to cut)
- Abstract domains
  (e.g. invariants vs ranking functions, lfp vs gfp)
- Predicate interdependencies
  $\rightarrow$ avoid cutting cycles (still need fixed point, lfp vs gfp)
- Precision / solving capacity-driven (inference problems)
  As much as possible, as little as necessary
- Property-driven (decision problems)
  Decomposition = abstraction (start rough and refine)

Lessons Learned

Challenges

- Dynamic decomposition
  decomposing upfront is difficult
- Better predicate inference algorithms
  (product domains, under-approximations, lfp+gfp, . . .)
- Decomposition of cycles?
  $\rightarrow$ precise handling of recursive functions
- Precision / solving capacity-driven (inference problems)
  $\rightarrow$ best-effort refinement
- Property-driven (decision problems)
  $\rightarrow$ modular refinement algorithms