Combining Equational Theories

Combining Logical Interpreters

Ashish Tiwari

Tiwari@csl.sri.com

Computer Science Laboratory

SRI International

Menlo Park CA 94025

`http://www.csl.sri.com/~tiwari`

Outline

- Combining Decision Procedures
  - Nelson-Oppen Combination Result
  - Constructive Version of the Nelson-Oppen Combination
  - Application: Abstract Congruence Closure + AC Congruence Closure + Polynomial Rings
- Combining Abstract Interpreters
  - Logical Abstract Interpretation
  - Combining Logical Interpreters
  - Hardness of Combination

Combining Decision Procedures

Satisfiability Problem : Decide if $\mathbf{T} \models \exists \vec{x} : \phi$, where

$\mathbf{T}$: theory

ϕ : finite conjunction of literals

$\vec{x}$: all the variables in ϕ

Satisfiability Problem in Combination of Theories : $\mathbf{T} \quad := \quad \mathbf{T}_1 \cup \mathbf{T}_2$

Nelson-Oppen Combination Result

$\mathbf{T}_1$, $\mathbf{T}_2$	:	consistent, stably-infinite theories over disjoint signatures
$T_1(n), T_2(n)$	:	time complexity of $\mathbf{T}_i$-satisfiability problem

Then,

1. $\mathbf{T} := \mathbf{T}_1 \cup \mathbf{T}_2$ is consistent and stably infinite.
2. Time complexity of $\mathbf{T}$-satisfiability is $O(2^{n^2} * (T_1(n) + T_2(n)))$.
3. If $\mathbf{T}_1$ and $\mathbf{T}_2$ are convex, then so is $\mathbf{T}$ and
   $\mathbf{T}$-satisfiability is in $O(n^3 * (T_1(n) + T_2(n)))$ time.

Nelson-Oppen Result: Correctness

The combination procedure:

Initial State : ϕ is over $\Sigma_1 \cup \Sigma_2$.

Purification : Transform ϕ to $\phi_1 \wedge \phi_2$ s.t. ϕ_i is over Σ_i.

Interaction : Guess a partition of $\mathcal{V}(\phi_1) \cap \mathcal{V}(\phi_2)$. Express it as ψ.
E.g. partition $\{x_1\}, \{x_2, x_3\}$ is $x_2 = x_3 \wedge x_1 \neq x_2 \wedge x_1 \neq x_3$.

Component Procedures : Decide if $\phi_i \wedge \psi$ is $\mathbf{T}_i$-satisfiable.

Return : If both answer yes, return yes. No, otherwise.

Time Complexity: $O(2^{n^2} * (T_1(n) + T_2(n)))$

NO Deterministic Procedure for Convex Theories

If $\mathbf{T}_1, \mathbf{T}_2$ are convex, we can deduce equalities to be shared.

Purification : As before.

Interaction : Deduce an equality $x{=}y$:

$$\mathbf{T}_1 \vdash (\phi_1 \Rightarrow x{=}y)$$

Update $\phi_2 \;:=\; \phi_2 \wedge x{=}y$. And vice-versa. Repeat until no further changes to get ϕ_{i_∞}.

Component Procedures : Decide if ϕ_{i_∞} is satisfiable.

Time Complexity: $O(n^3 * (T_1(n) + T_2(n)))$

Outline

- Combining Decision Procedures
  - Nelson-Oppen Combination Result
  - Constructive Version of the Nelson-Oppen Combination
  - Application: Abstract Congruence Closure + AC Congruence Closure + Polynomial Rings
- Combining Abstract Interpreters
  - Logical Abstract Interpretation
  - Combining Logical Interpreters
  - Hardness of Combination

NO: Equational Theory Version

1. Equational theories are always consistent.
2. If $E \cup \{\exists x, y. x \neq y\}$ is consistent, then this theory is also stably-infinite.
3. Equational theories are convex.
4. Often decision procedures based on standard Knuth-Bendix completion can be used to deduce equalities.
5. Therefore, satisfiability procedures can be combined with only a polynomial factor overhead.

Satisfiability $\leftrightarrow$ Uniform Word Problem $\leftrightarrow$ Completion

Nelson-Oppen Combination: Constructive Variant

Assumptions:

- R_1, R_2: two convergent presentations (for two disjoint equational theories)
- $\succ$: Ordering over $\Sigma_i \cup K$ s.t. constants in K are minimal
- $Com_i(E, \succ)$: Completion modulo R_i of ground E (over $\Sigma_i \cup K$)

Then, there is a procedure $Com(E, \succ)$ that performs completion modulo $R_1 \cup R_2$ of ground E over $\Sigma_1 \cup \Sigma_2$

Constructive NO Procedure

- Purify E: $E \mapsto E_1 \cup E_2$, E_i over $\Sigma_i \cup K$
- Componentwise completion:

$$\begin{aligned} E_1^{(1)} &:= Com_1(E_1, \succ) \\ E_2^{(1)} &:= Com_2(E_2, \succ) \end{aligned}$$

- Share equalities between constants and repeat above step

$$\begin{aligned} E_1^{(j+1)} &:= Com_1(E_1^{(j)} \cup E_2^{(j)}|_K, \succ) \\ E_2^{(j+1)} &:= Com_2(E_2^{(j)} \cup E_1^{(j)}|_K, \succ) \end{aligned}$$

- The final system $(R_1 \cup R_2) \cup (E_1^{\infty} \cup E_2^{\infty})$ is convergent

Ordering guarantees that equalities among constants are smallest

Hence, they are always generated by the individual completion procedures

Outline

- Combining Decision Procedures
  - Nelson-Oppen Combination Result
  - Constructive Version of the Nelson-Oppen Combination
  - Application: Abstract Congruence Closure + AC Congruence Closure + Polynomial Rings
- Combining Abstract Interpreters
  - Logical Abstract Interpretation
  - Combining Logical Interpreters
  - Hardness of Combination

Application 1: Abstract Congruence Closure

Σ : Signature consisting of function symbols and constants

R : empty theory

E : ground equation over Σ

We view the problem of completing E as a combination problem

Σ = $\bigcup_i \Sigma_i$, disjoint union of singleton Σ_i

R = $\bigcup_i R_i$, empty background theory

We only need $Com_i(E, \succ)$.

Abstract Congruence Closure: Example

$$a = fab, f(fab)b = b$$

$a \to c_1$	$b \to c_2$	$fc_1c_2 \to c_3, fc_3c_2 \to c_4$	$c_1 = c_3, c_4 = c_2$
$a \to c_1$	$b \to c_2$	$fc_1c_2 \to c_3, fc_3c_2 \to c_4$	$c_3 \to c_1, c_4 \to c_2$
$a \to c_1$	$b \to c_2$	$fc_1c_2 \to c_3, fc_1c_2 \to c_4$	$c_3 \to c_1, c_4 \to c_2$
$a \to c_1$	$b \to c_2$	$fc_1c_2 \to c_3$	$c_4 = c_3, c_3 \to c_1, c_4 \to c_2$
$a \to c_1$	$b \to c_2$	$fc_1c_2 \to c_3$	$c_2 = c_1, c_3 \to c_1, c_4 \to c_2$
$a \to c_1$	$b \to c_2$	$fc_1c_2 \to c_3$	$c_2 \to c_1, c_3 \to c_1, c_4 \to c_2$
$a \to c_1$	$b \to c_2$	$fc_1c_1 \to c_3$	$c_2 \to c_1, c_3 \to c_1, c_4 \to c_2$
$a \to c_1$	$b \to c_1$	$fc_1c_1 \to c_3$	$c_2 \to c_1, c_3 \to c_1, c_4 \to c_1$

Time Complexity: $O(n \log(n))$

Application 2: AC Congruence Closure

Σ : Signature consisting of function symbols and constants including some associative-commutative symbols

R : AC rules for the AC symbols

E : ground equation over Σ

We view the problem of completing E as a combination problem

Σ = $\bigcup_i \Sigma_i$, disjoint union of singleton Σ_i

R = $\bigcup_i R_i$, where R_i has AC rules *if* Σ_i contains an AC symbol and it is the empty background theory *otherwise*

We only $Com_i(E, \succ)$ for *one* AC symbol f.

Completion for one AC symbol

Example: Use notation $c_1^2 c_2$ for $f(c_1, f(c_1, c_2))$

$$
\begin{array}{ccc}
c_1^2 c_2 = c_2, & c_1 c_2^2 = c_2 & \\
\hline
c_1^2 c_2 \rightarrow c_2, & c_1 c_2^2 \rightarrow c_2 & \\
\hline
c_1^2 c_2 \rightarrow c_2, & c_1 c_2^2 \rightarrow c_2, & c_2^2 = c_1 c_2 \\
\hline
c_1^2 c_2 \rightarrow c_2, & c_1 c_2^2 \rightarrow c_2, & c_1 c_2 = c_2^2 \\
\hline
c_1^2 c_2 \rightarrow c_2, & c_1 c_2^2 \rightarrow c_2, & c_1 c_2 \rightarrow c_2^2 \\
\hline
c_1 c_2^2 = c_2, & c_2^3 = c_2, & c_1 c_2 \rightarrow c_2^2 \\
\hline
c_2^3 = c_2, & c_2^3 = c_2, & c_1 c_2 \rightarrow c_2^2 \\
\hline
c_2^3 \rightarrow c_2, & & c_1 c_2 \rightarrow c_2^2
\end{array}
$$

Use of total-degree lex ordering important for combination

Application 3: UF + AC(U) + Polynomial Rings

$$\Sigma = \Sigma_F \cup \Sigma_{AC} \cup \Sigma_{ACU} \cup \Sigma_{GB}$$

R = Convergent presentation for polynomial rings + AC rules

E = Ground equations over Σ

We again view the problem of completing E (modulo R) as a combination problem

We get $Com(E, \succ)$ by combining:

- abstract congruence closure for equations on Σ_F
- $AC(U)$ congruence closure for equations on each $f \in \Sigma_{AC(U)}$
- Gröbner basis algorithm for equations over Σ_{GB}

Since each theory is convex and stably-infinite, we get a polynomial time combination over the individual theories.

Gröbner Basis Example

$$\frac{\frac{\frac{\frac{\frac{\frac{\frac{\frac{\frac{xy - x = 0,\ x^2 - x - 1 = 0}{xy \to x,\ x^2 \to x + 1}}{xy \to x,\ x^2 \to x + 1,\ x * x = y * (x + 1)}}{xy \to x,\ x^2 \to x + 1,\ x^2 \to xy + y}}{xy \to x,\ x^2 \to x + 1,\ x + 1 = xy + y}}{xy \to x,\ x^2 \to x + 1,\ xy \to x - y + 1}}{xy \to x,\ x^2 \to x + 1,\ x = x - y + 1}}{xy \to x,\ x^2 \to x + 1,\ y \to 1}}{x = x,\ x^2 \to x + 1,\ y \to 1}}{x^2 \to x + 1,\ y \to 1}$$

Outline

- Combining Decision Procedures
  - Nelson-Oppen Combination Result
  - Constructive Version of the Nelson-Oppen Combination
  - Abstract Congruence Closure + AC Congruence Closure + Polynomial Rings
- Combining Abstract Interpreters
  - Logical Abstract Interpretation
  - Combining Logical Interpreters
  - Hardness of Combination

Abstract Interpretation

Interpreting program over abstract values from domain D with partial order $\sqsubseteq$

```
  [ true ]
  x := 0; y := 0; z := n;
  [ x = 0 ]
  while (*) {
      [ x = 0 ∨ x ≥ 0 = (x ≥ 0) ]
      if (*) {
          x++; z--; [ x ≥ 0 ]
      } else {
          y++; z--; [ x = 0 ]
      }
      [ x ≥ 0 ]
  }
```

Domain D:

$$\mathit{false}, \quad x \le 0, \quad x = 0, \quad \mathit{true}, \quad x \ge 0$$

Partial Order $\sqsubseteq$:

$$\begin{array}{rcl} x = 0 & \sqsubseteq & x \ge 0 \\ x = 0 & \sqsubseteq & x \le 0 \\ \mathit{false} & \sqsubseteq & d \\ d & \sqsubseteq & \mathit{true} \end{array}$$

Logical Abstract Interpretation

D: logical formulas E theory $\mathbf{T}$

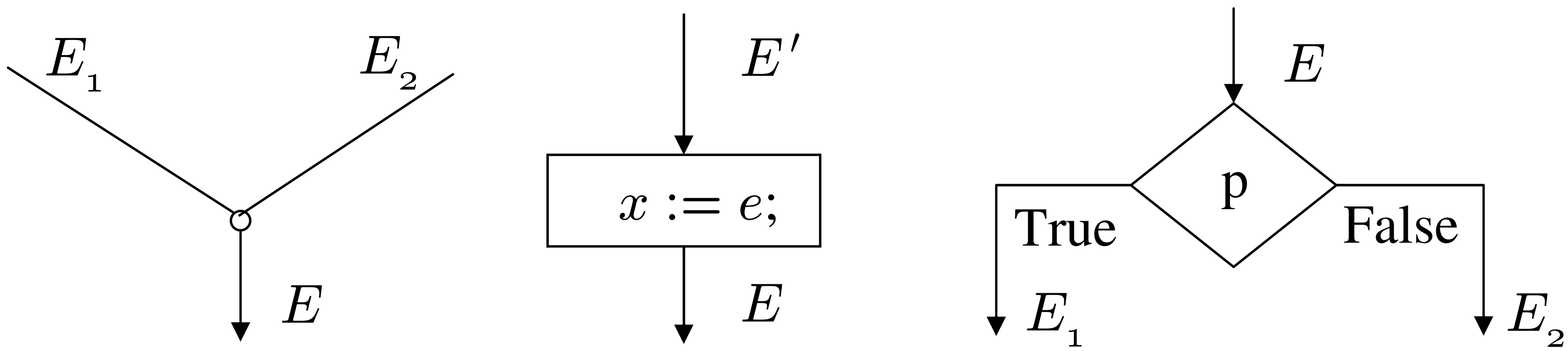
$\sqsubseteq$: logical implication, $E \sqsubseteq E'$ iff $E \Rightarrow_{\mathbf{T}} E'$

Examples of logical interpretation:

- D consists of finite conjunctions of atomic facts over $\mathbf{T}$.
    - Linear Arithmetic
    - Uninterpreted Functions
    - Combination of Linear Arithmetic and Uninterpreted Functions
- D consists of universally quantified formulas over $\mathbf{T}$.

Transfer Functions for Logical AI

Transfer functions compute abstract facts at each program point from facts at preceding program points



(a) Join Node (b) Assignment Node (c) Conditional Node

Join	:	$E := E_1 \lceil \vee \rceil E_2$
Assignment	:	$E := \lceil \exists \rceil x' : (E'[x'/x] \wedge x = e)$
Conditional	:	$E_1 := E \wedge p$, $E_2 := E \wedge \neg p$

Logical Domain

D	:	Class of formulas using $\mathbf{T}$-symbols and the program variables
$E \sqsubseteq E'$	:	$E \Rightarrow_{\mathbf{T}} E'$
$E_1 \lceil\vee\rceil E_2$	:	least upper bound of E_1 and E_2 in D
		least $E \in D$ s.t. $E_1 \sqsubseteq E$, $E_2 \sqsubseteq E$
		least $E \in D$ s.t. $E_1 \vee E_2 \Rightarrow_{\mathbf{T}} E$
$\lceil\exists\rceil x : E'$	:	least upper bound of $\{E \mid E' \sqsubseteq E, x \notin \mathcal{V}(E)\}$
		least $E \in D$ s.t. $\exists x : E' \Rightarrow_{\mathbf{T}} E$

Logical Abstract Interpreter

Initialize : Assign an abstract fact to each program point

- $true$ to the program entry point
- $false$ to all other program points

Propagate : Use transfer functions to update facts at each program point

Terminate : Upon reaching a fixpoint

Require operators for over-approximating disjunction and existential quantifier elimination, and for deciding logical implication

Linear Arithmetic Logical Domain

The linear arithmetic logical domain:

D	:	conjunction of linear equations over program variables
$E \sqsubseteq E'$	:	$E \Rightarrow_{LA} E'$
$E_1 \lceil\vee\rceil E_2$	:	least $E \in D$ s.t. $E_1 \vee E_2 \Rightarrow_{\mathbf{T}} E$
$\lceil\exists\rceil x : E'$	:	least $E \in D$ s.t. $\exists x : E' \Rightarrow_{\mathbf{T}} E$

Examples:

$$\begin{aligned}
(x = 0 \wedge y = 1 \wedge z = n-1) \lceil\vee\rceil x = 1 \wedge y = 0 \wedge z = n-1 &= x + y = 1 \wedge \\
& z = n-1 \\
(x + y = 1 \wedge z = n-1) \lceil\vee\rceil x = 0 \wedge y = 0 \wedge z = n &= x + y + z = n \\
\lceil\exists\rceil x', z' : (x' + y + z' = n \wedge x = x' + 1 \wedge z = z' - 1) &= x + y + z = n
\end{aligned}$$

Linear Arithmetic Abstract Interpreter

`[ true ]`
`x := 0; y := 0; z := n;`
`[` $x = 0 \wedge y = 0 \wedge z = n$ `]`
`while (*) {`
    `[` $(x = 0 \wedge y = 0 \wedge z = n) \vee (x + y = 1 \wedge z = n - 1)$ `]`
    `if (*) {`
        `x := x+1;`
        `z := z-1; [` $(x = 1 \wedge y = 0 \wedge z = n - 1)$ `]`
    `} else {`
        `y := y+1;`
        `z := z-1; [` $(x = 0 \wedge y = 1 \wedge z = n - 1)$ `]`
    `}`
    `[` $(x + y = 1 \wedge z = n - 1)$ `]`
`}`

Uninterpreted Functions Logical Domain

The uninterpreted functions logical domain:

D	:	conjunction of equations over UF and program variables
$E \sqsubseteq E'$	:	$E \Rightarrow_{UF} E'$
$E_1 \lceil\vee\rceil E_2$	:	least $E \in D$ s.t. $E_1 \vee E_2 \Rightarrow_{UF} E$
$\lceil\exists\rceil x : E'$	:	least $E \in D$ s.t. $\exists x : E' \Rightarrow_{UF} E$

Examples:

$$(u = c \wedge v = c) \lceil\vee\rceil u = F(c) \wedge v = F(c) \quad = \quad u = v$$

$$\lceil\exists\rceil u', v' : (u' = v' \wedge u = F(u') \wedge v = F(v')) \quad = \quad u = v$$

Uninterpreted Functions Abstract Interpreter

`[ true ]`
`u := c; v := c;`
[$u = c \wedge v = c$]
`while (*) {`
    [$(u = c \wedge v = c) \lceil\vee\rceil (u = F(c) \wedge v = F(c))$]
    `u := F(u);`
    `v := F(v);`
    [$(u = F(c) \wedge v = F(c))$]
`}`
[$u = v$]

Outline

- Combining Decision Procedures
  - Nelson-Oppen Combination Result
  - Constructive Version of the Nelson-Oppen Combination
  - Abstract Congruence Closure + AC Congruence Closure + Polynomial Rings
- Combining Abstract Interpreters
  - Logical Abstract Interpretation
  - Combining Logical Interpreters
  - Hardness of Combination

Combining Abstract Interpreters: Motivation

```
x := 0; y := 0;
u := 0; v := 0;
while (*) {
  x := u + 1;
  y := 1 + v;
  u := F(x);
  v := F(y);
}
assert( x = y )
```

$\Sigma = \Sigma_{LA} \cup \Sigma_{UF}$

$\mathbf{T} = \mathbf{T}_{LA} + \mathbf{T}_{UF}$

```
x:= c; y := c;
u := c; v := c;
while (*) {
 x := G(u, 1);
 y := G(1, v);
 u := F(x);
 v := F(y);
}
assert( x = y )
```

$\Sigma = \Sigma_{UF}$

$\mathbf{T} = \mathbf{T}_{UF}$

```
x:= 0; y := 0;
u := 0; v := 0;
while (*) {
 x := u + 1;
 y := 1 + v;
 u := *;
 v := *;
}
assert( x = y )
```

$\Sigma = \Sigma_{LA}$

$\mathbf{T} = \mathbf{T}_{LA}$

Combining Logical Lattices

Combining abstract interpreters is not easy [Cousot76]

Given logical lattices L_1 and L_2:

- Direct product: $\langle L_1 \times L_2, \ \Rightarrow_{\mathbf{T}_1} \times \Rightarrow_{\mathbf{T}_2}\rangle$
- Reduced product: $\langle L_1 \times L_2, \ \Rightarrow_{\mathbf{T}_1 \cup \mathbf{T}_2}\rangle$
- Logical+ product: $\langle$Infinite* conjunctions of $AF(\Sigma_1 \cup \Sigma_2, \mathcal{V}), \Rightarrow_{\mathbf{T}_1 \cup \mathbf{T}_2}\rangle$
- Logical product:
  $\langle$Conjunctions of $AF(\Sigma_1 \cup \Sigma_2, \mathcal{V}), \Rightarrow_{\mathbf{T}_1 \cup \mathbf{T}_2}$ with some restriction$\rangle$

Different Kinds of Combinations

Kind	Lattice elements	Lattice Preorder	Can verify
Logical+	Inf conj of atm facts in $T_1 \cup T_2$	$\Rightarrow_{T_1 \cup T_2}$	1,2, 3 , 4
Logical	conj of atm facts in $T_1 \cup T_2$	$\Rightarrow^{\preceq}_{T_1 \cup T_2}$	1,2, 3
Reduced	$L_1 \times L_2$	$\Rightarrow_{T_1 \cup T_2}$	1,2
Direct	$L_1 \times L_2$	$\Rightarrow_{T_1} \times \Rightarrow_{T_2}$	1

```
if (*)
   x := 1; y := F(1); z := G(2);
else
   x := 4; y := F(8-x); z := G(5);
```

Assertions: $x \geq 1$, $y = F(x)$, $z = G(x+1)$,
$H(x) + H(5-x) = H(1) + H(4)$

Issues in Combining Logical Lattices

Why not use the logical+ product?

The logical+ product is undesirable for two reasons:

1. Join may contain infinite atoms

$$\begin{aligned} & (x = 0 \wedge y = 1) \sqcup (x = 1 \wedge y = 0) \\ = \quad & x + y = 1 \wedge C[x] + C[y] = C[0] + C[1] \end{aligned}$$

2. Combination can be hard (later)

Logical Product

Given two logical lattices, we define the logical product as:

elements : conjunction ϕ of atomic formulas in $\mathbf{T}_1 \cup \mathbf{T}_2$

$E \lfloor \Rightarrow \rfloor E'$: $E \Rightarrow_{\mathbf{T}_1 \cup \mathbf{T}_2} E'$ and $\mathit{AlienTerms}(E') \subseteq \mathit{Terms}(E)$

$$\begin{array}{lcl} \mathit{AlienTerms}(E) & = & \text{subterms in } E \text{ that belong to different theory} \\ \mathit{Terms}(E) & = & \text{all subterms in } E\text{, plus all terms equivalent} \\ & & \text{to these subterms (in } \mathbf{T}_1 \cup \mathbf{T}_2 \cup E\text{)} \end{array}$$

Eg. $\{x = F(a+1), y = a\} \lceil \vee \rceil \{x = F(b+1), y = b\} = \{x = F(y+1)\}$ $\because$

$$\begin{array}{rcl} x = F(a+1) \wedge y = a & \Rightarrow & x = F(y+1) \\ x = F(b+1) \wedge y = b & \Rightarrow & x = F(y+1) \\ x = F(\underline{a+1}) \wedge y = a & \Rightarrow & y+1 = \underline{a+1} \\ x = F(\underline{b+1}) \wedge y = b & \Rightarrow & y+1 = \underline{b+1} \end{array}$$

Logical Product

- Includes only those atomic facts in the least upper bound of E and E' whose alien terms occur semantically in both elements E and E'
- Is more powerful than direct product and reduced product
- Allows us to combine the abstract interpreters modularly in some cases

We will discuss how to combine the abstract interpretation operations

Combining the Preorder Test

Required for testing convergence of fixpoint

$E \lfloor\Rightarrow\rfloor E'$ iff

1. $\mathbf{T}_1 \cup \mathbf{T}_2 \models E \Rightarrow E'$
2. $\mathit{AlienTerms}(E') \subseteq \mathit{Terms}(E)$

So, the crucial problem is (1)

Nelson-Oppen Combination Procedure can solve (1)

Works for convex, stably-infinite, disjoint theories

Combining Join Operator

Given procedures to compute:

$E_l \lceil \vee \rceil_{\mathbf{T}_1} E_r$

$E_l \lceil \vee \rceil_{\mathbf{T}_2} E_r$

We wish to compute $E_l \lceil \vee \rceil_{\mathbf{T}_1 \cup \mathbf{T}_2} E_r$

Example.

$$\{z + 1 = a, y = f(a)\} \lceil \vee \rceil_{UF+LA} \{z = b - 1, y = f(b)\} \quad = \quad \{y = f(1 + z)\}$$

Combining Join Operators

$L_{12} \lceil \vee \rceil_{\mathbf{T}_{12}} R_{12} =$

1. $\langle L_1, L_2 \rangle$:= Purify and Saturate L_{12};
   $\langle R_1, R_2 \rangle$:= Purify and Saturate R_{12};
2. $D_L := \bigwedge \{ v_i = \langle v_i, v_j \rangle \mid v_i \in \mathcal{V}(L_1, L_2), v_j \in \mathcal{V}(R_1, R_2) \}$;
   $D_R := \bigwedge \{ v_j = \langle v_i, v_j \rangle \mid v_i \in \mathcal{V}(L_1, L_2), v_j \in \mathcal{V}(R_1, R_2) \}$;
3. $L'_1 := L_1 \wedge D_L$; $R'_1 := R_1 \wedge D_R$;
   $L'_2 := L_2 \wedge D_L$; $R'_2 := R_2 \wedge D_R$;
4. $A_1 := L'_1 \lceil \vee \rceil_{\mathbf{T}_1} R'_1$;
   $A_2 := L'_2 \lceil \vee \rceil_{\mathbf{T}_2} R'_2$;
5. $V := \mathcal{V}(A_1, A_2) -$ Program Variables;
   $A_{12} := \lceil \exists \rceil_{\mathbf{T}_{12}} (A_1 \wedge A_2, V)$;
6. Return A_{12};

Combining Join Operators

$$\boxed{z = a - 1, y = f(a)} \qquad \boxed{z = b - 1, y = f(b)}$$

Purify+NOSat	$z = a - 1$	$y = f(a)$	$z = b - 1$	$y = f(b)$
LR-Exchange	$a = \langle a, b \rangle$	$a = \langle a, b \rangle$	$b = \langle a, b \rangle$	$b = \langle a, b \rangle$
Base Joins	$\boxed{\lceil \vee \rceil_{LA}}$		$\boxed{\lceil \vee \rceil_{UF}}$	
	$\langle a, b \rangle = 1 + z$		$y = f(\langle a, b \rangle)$	
Quant Elim	$\boxed{\lceil \exists \rceil_{UF+LA}}$			
Return	$\boxed{y = f(1 + z)}$			

Combining Existential Quantifier Elimination

Required to compute transfer function for assignments

$E = \lceil \exists \rceil_{\mathbf{T}} V : E'$ if E is the least element s.t.

- $E' \lfloor \Rightarrow \rfloor_{\mathbf{T}} E$
- $\mathcal{V}(E) \cap V = \emptyset$

Examples:

- $\lceil \exists \rceil_{LA}(\{x < a, a < y\}, \{a\}) = \{x \leq y\}$
- $\lceil \exists \rceil_{UF}(\{x = f(a), y = f(f(a))\}, \{a\}) = \{y = f(x)\}$
- $\lceil \exists \rceil_{LA+UF}(\{a < b < y, z = c+1, a = ffb, c = fb\}, \{a, b, c\}) = \{f(z-1) \leq y\}$

How to construct $\lceil \exists \rceil_{LA+UF}$ using $\lceil \exists \rceil_{LA}$ and $\lceil \exists \rceil_{UF}$?

Combining $\lceil\exists\rceil$ Operators

$\lceil\exists\rceil_{\mathbf{T}_{12}} V : E_{12}$

1. $\langle E_1, E_2 \rangle$:= Purify and Saturate E_{12};
2. $\langle D, Defs \rangle := \mathit{DefSaturate}(E_1, E_2, V \cup \mathit{TempVars})$;
3. $V' := V \cup \mathit{TempVars} - D$;
   $E_1' := \lceil\exists\rceil_{T_1} V' : E_1$;
   $E_2' := \lceil\exists\rceil_{T_2} V' : E_2$;
4. $E := (E_1' \wedge E_2')[Defs(y)/y]$;
5. Return E;

$\mathit{DefSaturate}(E_1, E_2, U)$ returns the set of all variables D that have definitions $Defs$ in terms of variables not in U as implied by $E_1 \wedge E_2$

Combining $\lceil\exists\rceil$ Operators

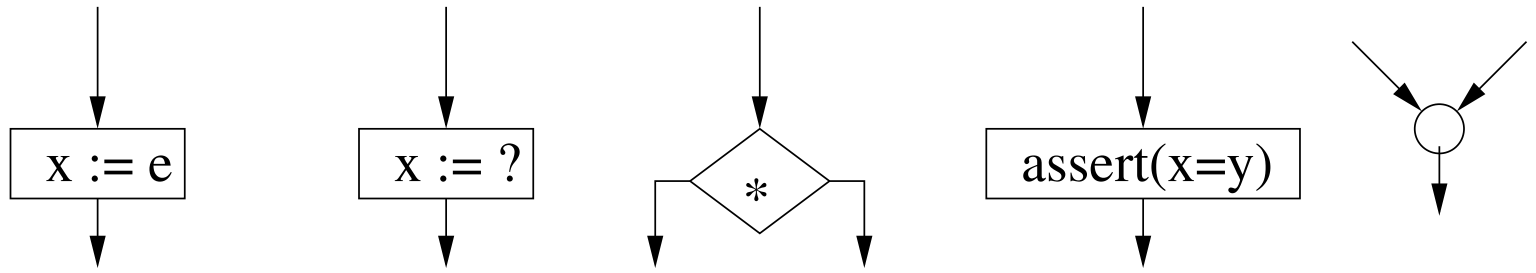
Problem	$\boxed{a < b < y, z = c + 1, a = ffb, c = fb}$		$\boxed{\{a, b, c\}}$
Purify+NOSat	$\boxed{a < b < y, z = c + 1}$		$\boxed{a = ffb, c = fb}$
QSat		$\longrightarrow$	$c \mapsto z - 1$
QSat	$a \mapsto fc$	$\longleftarrow$	
Base $\lceil\exists\rceil$	$\boxed{\lceil\exists\rceil_{LA} b}$		$\boxed{\lceil\exists\rceil_{UF} b}$
	$\boxed{a < y, z = c + 1}$		$\boxed{a = fc}$
Substitute	$c \mapsto z - 1, a \mapsto fc$		
Return	$\boxed{f(z-1) < y}$		

Outline

- Combining Decision Procedures
  - Nelson-Oppen Combination Result
  - Constructive Version of the Nelson-Oppen Combination
  - Abstract Congruence Closure + AC Congruence Closure + Polynomial Rings
- Combining Abstract Interpreters
  - Logical Abstract Interpretation
  - Combining Logical Interpreters
  - Hardness of Combination

Assertion Checking Problem

Verify that an assertion is an invariant in a program:



Choices for underlying theory:

Linear Arithmetic	:	$e := y \mid c \mid e_1 \pm e_2 \mid ce$
Uninterpreted	:	$e := y \mid F(e_1, e_2)$
Combination	:	$e := y \mid c \mid e_1 \pm e_2 \mid ce \mid F(e_1, e_2)$

Complexity of Assertion Checking Problem

With interpreted conditionals, it is undecidable

Theory	Complexity	Method
Linear Arithmetic	PTime	Logical AI over LA [Karr 76]
Uninterpreted	PTime	Logical AI over UF [GN 04]
Combination	coNP hard	

Logical AI over Logical+ product can not be combined efficiently.

coNP-hardness: Checking Disjunctive Assertions

ψ: boolean 3-SAT instance with m clauses and k variables

$x_i := 0$, for $i = 1, 2, \ldots, m$
for $i = 1$ to k do
    if (*) then
        $x_j := 1$, $\forall j$: variable i occurs positively in clause j
    else
        $x_j := 1$, $\forall j$: variable i occurs negatively in clause j
$sum := x_1 + \cdots + x_m$
assert($sum = 0 \vee \cdots \vee sum = m - 1$)

Assertion is valid IFF ψ is unsatisfiable

coNP-hardness of Assertion Checking

Key Idea: Disjunctive assertion can be encoded in the combination.

$$x = a \vee x = b \quad \Leftrightarrow \quad F(a) + F(b) = F(x) + F(a + b - x)$$

Using this recursively, we can write an assertion (atomic formula) which holds iff $x = 0 \vee x = 1 \vee \cdots \vee x = m - 1$ holds.

For e.g., encoding for $x = 0 \vee x = 1 \vee x = 2$ is obtained by encoding $Fx = F2 \vee Fx = F0 + F1 - F(1 - x)$:

$$F(F0 + F1 - F(1 - x)) + FF2 = FFx + F(F0 + F1 + F2 - F(1 - x) - Fx)$$

Conclusions

- We can combine completion procedures using a constructive variant of NO result
- Logical lattices are good candidates for thinking about and building abstract interpreters.
  - Logical product is more powerful than direct or reduced product
  - Operations on logical lattices can be modularly combined to yield operations for logical products
  - Using ideas from the classical Nelson-Oppen combination method
- The assertion checking problem on logical+ product of two logical lattices can be hard, even when it is in PTime for the component logical lattices