INVARIANT RELATIONS: EVERYTHING YOU ALWAYS WANTED TO KNOW ABOUT LOOPS

Ali Mili, NJIT
NII, Tokyo, Japan
December 27, 2011
Plan

- Summary of Last Week
- Correctness and Refinement
- Program Semantics
- Invariant Relations and Termination
- Invariant Relations and Correctness
- Invariant Relations and Predicate Transformers
- Conclusion: Summary and Prospects
- References
Summary of Last Week

- While loop \( w \) on space \( S \) of the form:
  
  \[
  \text{while } t \text{ do } b. 
  \]

- Invariant Relation: Reflexive transitive superset of \( (T \cap B) \), where
  - \( T \) is the vector that represents \( t \): \( \{(s,s') | t(s)\} \).
  - \( B \) is the relation that represents \( b \):
    
    \( \{(s,s') | \text{if } b \text{ executes on state } s \text{ it terminates in } s'\} \).

- How do we generate invariant relations:
  - If variable \( x \) is incremented: \( \{(s,s') | x \leq x'\} \).
  - If variable \( x \) is decremented: \( \{(s,s') | x \geq x'\} \).
  - If variable \( x \) is preserved: \( \{(s,s') | x=x'\} \).
  - If function \( V \) is invariant: \( \{(s,s') | V(s) = V(s')\} \).
SUMMARY OF LAST WEEK

Interest of Invariant Relations:

- An Invariant Relation \( R \) can be converted into an approximation of the loop function:
  \[ R \cap \hat{T} \supseteq W. \]

- An Invariant Relation \( R \) can be converted into an invariant assertion for a given precondition:
  \[ A = \hat{R}P. \]

All invariant assertions stem from invariant relations: product of an invariant relation by the precondition.

- A Symmetric Invariant Relation \( R \) can be converted to an invariant function by projection (each element of \( S \) mapped onto its equivalence class):
  \[ V = \prod_R. \]
SUMMARY OF LAST WEEK

- Automation of Invariant Relation Based Loop Analysis:
  - *Source Code (cpp) to Internal Notation (cca)*:
    language independence + intersection (or union of intersections).
  - *Internal Notation (cca) to Invariant Relations (mat)*:
    Pattern matching against pre-stored recognizers, reflecting programming knowledge, domain knowledge.
    - Eventually, if cca is union of intersections, perform a merge.
  - *Post-Invariant Relation Analysis (mat to nb)*:
    Depends on what aspect of the loop we want to analyze.
SUMMARY OF LAST WEEK

- Post-Invariant Relation Analysis:
  - Approximating Loop Function ($R \cap \hat{T} \supseteq \hat{w}$): 
    \[
    \text{reduce}[R(s,s') \land \neg t(s')](s').
    \]
    demo last week.
  - Computing Invariant Assertion ($A = \hat{R}P$):
    \[
    \text{simplify}[\exists s': R(s',s) \land p(s')].
    \]
    demo.
PLAN

- Summary of Last Week
- Correctness and Refinement
- Program Semantics
- Invariant Relations and Termination
- Invariant Relations and Correctness
- Invariant Relations and Predicate Transformers
- Conclusion: Summary and Prospects
- References
Correctness and Refinement

- \( S = \{a, b, c, d, e, f, g, h, i, j\} \).
- \( R = \{(a,a), (a,b), (a,c), (b,b), (b,c), (b,d), (c,c), (c,d), (c,e)\} \).
- Candidate programs \( p1, p2, p3, p4, p5, p6, p7 \), such that:
  - \( P1 = \{(a,a), (b,b), (c,c)\} \).
  - \( P2 = \{(a,a), (b,b)\} \).
  - \( P3 = \{(a,a), (b,b), (d,d), (e,e)\} \).
  - \( P4 = \{(a,a), (b,b), (c,c), (d,d), (e,e), (f,f), (g,g)\} \).
  - \( P5 = \{(a,a), (b,c), (c,e), (d,f), (e,h), (f,j)\} \).
  - \( P6 = \{(a,f), (b,c), (c,d)\} \).
  - \( P7 = \{(a,f), (b,c), (c,d), (d,e), (e,f), (f,g), (g,h), (h,i)\} \).

Which of these programs are correct with respect to \( R \)?
Correctness and Refinement

- $S = \{a, b, c, d, e, f, g, h, i, j\}$.
- $R = \{(a,a), (a,b), (a,c), (b,b), (b,c), (b,d), (c,c), (c,d), (c,e)\}$.
- Candidate programs $p1, p2, p3, p4, p5, p6, p7$, such that:
  - $P1 = \{(a,a),(b,b),(c,c)\}$.
  - $P2 = \{(a,a),(b,b)\}$.
  - $P3 = \{(a,a),(b,b),(d,d),(e,e)\}$.
  - $P4 = \{(a,a),(b,b),(c,c),(d,d),(e,e),(f,f),(g,g)\}$.
  - $P5 = \{(a,a),(b,c),(c,e),(d,f),(e,h),(f,j)\}$.
  - $P6 = \{(a,f),(b,c),(c,d)\}$.
  - $P7 = \{(a,f),(b,c),(c,d),(d,e),(e,f),(f,g),(g,h),(h,i)\}$.

Which of these programs are correct with respect to $R$?
CORRECTNESS AND REFINEMENT

- Space $S$,
- Specification $R$ on $S$,
- Program $p$ on $S$, function $P$.

**Definition.** Program $p$ is said to be correct with respect to specification $R$ if and only if:

$$\forall s \in S: s \in \text{dom}(R) \implies s \in \text{dom}(P) \land (s, P(s)) \in R.$$ 

**Proposition.** Program $p$ is correct with respect to $R$ if and only if:

$$\text{dom}(R \cap P) = \text{dom}(R).$$
CORRECTNESS AND REFINEMENT

- Space $S$,
- Specification $R$ on $S$,
- Program $p$ on $S$, function $P$.

**Definition.** Program $p$ is said to be **correct with respect to specification $R$** if and only if:

$$\forall s \in S: \ s \in \text{dom}(R) \implies s \in \text{dom}(P) \land (s, P(s)) \in R.$$ 

**Proposition.** Program $p$ is correct with respect to $R$ if and only if:

$$\text{dom}(R \cap P) = \text{dom}(R).$$
**Correctness and Refinement**

- \( S = \{a, b, c, d, e, f, g, h, i, j\} \).
- \( R = \{(a,a), (a,b), (a,c), (b,b), (b,c), (b,d), (c,c), (c,d), (c,e)\} \).
- Candidate programs \( p_1, p_2, p_3, p_4, p_5, p_6, p_7 \), such that:
  - \( P_1 = \{(a,a),(b,b),(c,c)\} \).
    - \( R \cap P_1 = \{(a,a), (b,b), (c,c)\} \). \( \text{dom}(R \cap P_1) = \{a,b,c\} \)
  - \( P_2 = \{(a,a),(b,b)\} \).
  - \( P_3 = \{(a,a),(b,b),(d,d),(e,e)\} \).
  - \( P_4 = \{(a,a),(b,b),(c,c),(d,d),(e,e),(f,f),(g,g)\} \).
  - \( P_5 = \{(a,a),(b,c),(c,e),(d,f),(e,h),(f,j)\} \).
  - \( P_6 = \{(a,f),(b,c),(c,d)\} \).
    - \( R \cap P_6 = \{(b,c), (c,d)\} \). \( \text{dom}(R \cap P_1) = \{b,c\} \)
  - \( P_7 = \{(a,f),(b,c),(c,d),(d,e),(e,f),(f,g),(g,h),(h,i)\} \).
Correctness and Refinement

Refinement: A specification $R$ is said to refine a specification $R'$ if and only if any program correct with respect to $R$ is correct with respect to $R'$.

Intuitively, you make an input/output specification more refined by
- Involving more inputs,
- Allowing fewer outputs per input.
CORRECTNESS AND REFINEMENT

$R$ refines $R'$
CORRECTNESS AND REFINEMENT

\[ R \text{ refines } R'. \]
CORRECTNESS AND REFINEMENT

The most refined relations: Total deterministic relations.
**Correctness and Refinement**

Proposition. Specification $R$ refines specification $R'$ if and only if:

$$RL \cap R'L \cap (R \cup R') = R'.$$

Proposition. Program $p$ is correct with respect to specification $R$ if and only if $P$ refines $R$.

The same refinement ordering:
- Between specifications: $R$ is harder to satisfy than $R'$.
- Between a specification and a program: $P$ is correct with respect to $R$. 
PLAN

- Summary of Last Week
- Correctness and Refinement
- Program Semantics
- Invariant Relations and Termination
- Invariant Relations and Correctness
- Invariant Relations and Predicate Transformers
- Conclusion: Summary and Prospects
- References
PROGRAM SEMANTICS

Three ways to define the semantics of a program (programming language).

- Operational,
- Denotational,
- Axiomatic.
PROGRAM SEMANTICS

Three ways to define the semantics of a program (programming language).

- Operational,
  - Associate an operation to each program/ statement.

- Denotational,
  - Associate a function (denotation) to each program/ statement.

- Axiomatic,
  - Associate an axiom that characterizes (the behavior) of each program.
Program Semantics

Three ways to define the semantics of a program (programming language).

- **Operational,**
  - Associate an operation to each program/statement.
    - A compiler.

- **Denotational,**
  - Associate a function (denotation) to each program/statement.
    - Program function.

- **Axiomatic,**
  - Associate an axiom that characterizes (the behavior) of each program.
    - Weakest precondition, strongest postcondition.
Program Semantics

Hoare Formula:

\{p\} g \{q\}

If program \(g\) starts execution in a state that satisfies \(p\), and if it terminates, then upon termination, the state of the program verifies \(q\).

\{/p/\} g \{/q/\}

If program \(g\) starts execution in a state that satisfies \(p\), then it terminates, and upon termination, the state of the program verifies \(q\).

Partial correctness, total correctness.
PROGRAM SEMANTICS

Focus on partial correctness:

\{p\} g \{q\}

This formula does not characterize g, because p can be arbitrarily strong, and q arbitrarily weak.

Tighter characterization: weakest precondition, strongest postcondition.
PROGRAM SEMANTICS

Weakest Precondition:

\{p\} g \{q\}

Given a postcondition \(q\), what is the weakest precondition \(p\) that we must ensure prior to \(g\) to be sure \(q\) holds after \(g\).

Denoted by \(wp(g,q)\).

Any formula that logically implies \(wp(g,q)\) is called a \textit{precondition} for \(g\) with respect to \(q\).
Program Semantics
PROGRAM SEMANTICS

Strongest Postcondition:

\{p\} \ g \ \{q\}

Given a precondition \( p \), what is the strongest postcondition \( q \) that we can infer after execution of \( g \) if \( p \) is known to hold prior to \( g \).

Denoted by \( sp(g,p) \).

Any logical consequence of \( sp(g,p) \) is called a postcondition of \( g \) with respect to \( p \).
PROGRAM SEMANTICS
PLAN

• Summary of Last Week
• Correctness and Refinement
• Program Semantics
• Invariant Relations and Termination
• Invariant Relations and Correctness
• Invariant Relations and Predicate Transformers
• Conclusion: Summary and Prospects
• References
Invariant Relations and Termination

Under what conditions (on input states) do the following loops (on integer variables) terminate?

- while x ≠ 0 {x = x - 1;}
- while x ≠ 0 {x = x - 5;}
- while x > 0 {x = x - 1;}
- while x > 0 {x - x + y;}

29
IN Variant R Elations AND T ermination

Under what conditions (on input states) do the following loops (on integer variables) terminate?

- while $x \neq 0 \{x = x - 1;\}$
  - $x \geq 0$.

- while $x \neq 0 \{x = x - 5;\}$
  - $x \geq 0 \land x \text{ mod } 5 = 0$.

- while $x > 0 \{x = x - 1;\}$
  - true.

- while $x > 0 \{x = x + y;\}$
  - $(x \leq 0) \lor (y < 0)$. 
INVARIANT RELATIONS AND TERMINATION

Definition (sort of): The termination condition of a loop $w$ is the domain of $W$, which is represented by $WL$.

Proposition: If $R$ is an invariant relation for $w$ then

$$R\bar{T} \supseteq WL.$$ 

A necessary condition of termination.

Analysis Step (under Mathematica):

simplify[exists s': R(s,s')\&\&not t(s')].
Invariant Relations and Termination

Illustration:

- while x≠0 {x=x-5;}
  - \( R = \{(s,s') \mid x \geq x' \land x \mod 5 = x' \mod 5\} \).
  - \( R^T = \{(s,s') \mid \exists s'': x \geq x'' \land x \mod 5 = x'' \mod 5 \land x''=0\} \).
    
    \[
    = \{(s,s') \mid x \geq 0 \land x \mod 5 = 0\}.
    \]

Further Illustration

- Terminator
- Demo
## Invariant Relations and Termination

<table>
<thead>
<tr>
<th>ID</th>
<th>Loop</th>
<th>Terminator</th>
<th>cpu time (secs)</th>
<th>Invariant Relations</th>
<th>cpu time (secs)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>while (x &gt; 0) ({x = x + y})</td>
<td>((x \leq 0 \lor y &lt; 0))</td>
<td>NA</td>
<td>((x \leq 0 \lor y &lt; 0))</td>
<td>0.038</td>
<td>T: correct, IR: correct</td>
</tr>
<tr>
<td>E2</td>
<td>int (x, y, z); while (x &gt; 0) ({x = x + y; y = y + z})</td>
<td>(x \leq 0 \lor (x + y) \leq 0) (\lor x + 2y + z \leq 0) (\lor x + 3y + 3z \leq 0) (\lor y &lt; 0 \land z \leq 0) (\lor z &lt; 0)</td>
<td>24</td>
<td>(x \leq 0 \lor (x \geq 1 \lor (z = 0 \land y &lt; 0) \lor (z \geq 1 \lor y \leq \frac{z - \sqrt{8zx}}{2} \lor (z &lt; 0))))</td>
<td>0.312</td>
<td>T: correct, IR: necessary but not sufficient</td>
</tr>
<tr>
<td>E3</td>
<td>while (x \geq 0) ({x = -2^2x + 10;})</td>
<td>((x &lt; 0) \lor (x &gt; 5))</td>
<td>2</td>
<td><strong>true</strong></td>
<td>0.043</td>
<td>T: unnecessary, IR: correct</td>
</tr>
<tr>
<td>E4</td>
<td>while ((x &gt; 1)) ({\text{if } (x % 2 = 0) {x = x/2;}\text{ else }{x = 3^2x + 1;}})</td>
<td>No result</td>
<td>NA</td>
<td>(x \geq 0)</td>
<td>0.143</td>
<td>T: fails, IR: correct</td>
</tr>
<tr>
<td>E5</td>
<td>while ((x &gt; 0)) ({x = x - y; y = y + 1;})</td>
<td>no result</td>
<td>NA</td>
<td><strong>true</strong></td>
<td>0.068</td>
<td>T: unknown, IR: correct</td>
</tr>
<tr>
<td>E6</td>
<td>while ((x \leq N)) ({\text{if } y &lt; 0 {x = 2^2x + y; y = y + 1;}\text{ else }{x = x + 1;}})</td>
<td>((x &gt; N) \lor (x + y \geq 0))</td>
<td>4</td>
<td><strong>true</strong></td>
<td>0.098</td>
<td>T: unnecessary, IR: correct</td>
</tr>
<tr>
<td>E7</td>
<td>while ((x! = y)) ({\text{if } (x &gt; y) {x = x - y;}\text{ else }{y = y - x;}})</td>
<td>((x = y) \lor (x &gt; 0 \land y &gt; 0))</td>
<td>17</td>
<td>((x = y) \lor (x &gt; 0 \land y &gt; 0))</td>
<td>12.1</td>
<td>T: correct, IR: correct</td>
</tr>
</tbody>
</table>
Invariant Relations and Termination

Remark:
- Given a while loop on space $S$, and given its execution on some initial state $s$, we have found that if $w$ terminates on $s$, then all the states it generates during its execution (initial, final, intermediates) are in $\text{dom}(W)$. Hence we can take $S=\text{dom}(W)$ without loss of generality.
- Once we have computer the domain of $W$, we can redefine $S$ as $\text{dom}(W)$ without loss of generality.
- Doing so makes $W$ vacuously total.
- We assume $W$ total henceforth, without loss of generality.
INVARIANT RELATIONS AND TERMINATION

- demo
PLAN

• Summary of Last Week
• Correctness and Refinement
• Program Semantics
• Invariant Relations and Termination
• Invariant Relations and Correctness
• Invariant Relations and Predicate Transformers
• Conclusion: Summary and Prospects
• References
Invariant Relations and Correctness

A while loop $w$ on space $S$, a candidate specification $R$ on space $S$. Default option:
- Compute the function of the loop, say $W$.
- Check the refinement property: $W$ refines $R$?

Computing $W$: more effort than we need; more information than we have. Using invariant relations, we can generate
- A necessary condition of correctness (hence a sufficient condition of incorrectness).
- A sufficient condition of correctness.
- Collect enough information until one or the other concedes.
INVARIANT RELATIONS AND CORRECTNESS

A Necessary Condition of Correctness

Proposition. Let $w$ be a while loop of the form \texttt{while } $t$ \texttt{do } $b$ that terminates for all states in $S$, and let $R$ be an invariant relation for $w$. If $w$ computes function $U$ then $(R \cap U)^\top = L$.

Analysis Step (under Mathematica):
\texttt{simplify}[\exists s': \
\quad R(s,s') \land U(s,s') \land \neg t(s')].

If this expression does not reduce to \texttt{true}, we infer that $w$ does not compute $U$. 
INARIANT RELATIONS AND CORRECTNESS

Illustration:

- **S:** \( n, f, k: \) natural; \( 1 \leq k \leq n+1. \)

- **w:** \( \text{while } (k\neq n+1) \{f=f\times k; k=k+1;\} \)

- **R:** \( R = \left\{ (s,s') \mid \frac{f}{(k-1)!} = \frac{f'}{(k'-1)!} \right\}. \)

- **U:** \( U = \{(s,s') \mid n'=n \land f'=n! \land k'=n+1\}. \)
**Invariant Relations and Correctness**

\[(R \cap U)^\rightarrow\]

\[
= \{ \text{substitutions} \} \\
\{(s, s') | n' = n \land f' = n! \land k' = n + 1 \land \frac{f'}{n-1} = \frac{f'}{n-1} \} \circ \overline{T} \\
= \{ \text{simplifications} \} \\
\{(s, s') | f = (k - 1)! \land n' = n \land f' = n! \land k' = n + 1 \} \circ \overline{T} \\
= \{ \text{relational product} \} \\
\{(s, s') | \exists s'' : f = (k - 1)! \land n'' = n \land f'' = n! \land k'' = n + 1 \land k'' = n'' + 1 \} \\
= \{ \text{factoring, logical simplification} \} \\
\{(s, s') | f = (k - 1)! \} \\
\neq \{ \text{inspection} \} \\
L.

Hence, \(w\) does not compute function \(U\) (we don’t know what function it computes, but we know what function it does not compute).
INVARIANT RELATIONS AND CORRECTNESS

A Sufficient Condition of Correctness.

Proposition. Given a while loop \( w: \text{while } t \text{ do } b \), that terminates for all \( s \) in \( S \), and given a specification \( C \) on \( S \), and an invariant relation \( R \), if \( R \cap C \cap (C \cup R \cap \hat{T}) = C \)
then \( w \) is correct with respect to \( C \).

Analysis Step: simplify \[ (\exists s'' : R(s,s'') \land \neg t(s')) \land (\exists s'' : C(s,s'')) \land (C(s,s') \lor R(s,s') \land \neg t(s')) = C(s,s') \].

If this expression does reduce to \textbf{true}, we infer that \( w \) is correct with respect to \( C \).
Invariant Relations and Correctness

Illustration:

- **S**: \( n, f, k: \) natural; \( 1 \leq k \leq n+1 \).

- **w**: \( \text{while } (k \neq n+1) \{ f = f \cdot k; \ k = k+1; \} \)

- **R** = \( \left\{ (s, s') \mid \frac{f}{(k-1)!} = \frac{f'}{(k'-1)!} \right\} \).

- **C** = \( \{(s, s') \mid f = (k-1)! \land f' = n'! \} \).
Hence \( w \) is correct with respect to \( C \). The same invariant that was used to rule out function \( U \) (previous example) can be used to rule in specification \( C \).
Juggling Necessary and Sufficient Conditions:
- Generate more and more invariant relations, \( R1, R2, R3 \ldots \)
- As soon as one is found not to verify the necessary condition of correctness, rule out the candidate.
- As soon as the intersection of all invariant relations so far is found to satisfy the sufficient condition of correctness, rule in the candidate.
- Why the asymmetry?
  - One contradiction is sufficient to rule out an unsuccessful candidate.
  - Many confirmations may be required to rule in a successful candidate.
PLAN

• Summary of Last Week
• Correctness and Refinement
• Program Semantics
• Invariant Relations and Termination
• Invariant Relations and Correctness
• Invariant Relations and Predicate Transformers
• Conclusion: Summary and Prospects
• References
**Invariant Relations and Predicate Transformers**

Computing Strongest Postconditions.

**Definition.** We consider a while loop, \( w : \text{while } t \text{ do } b \), and we let \( P \) be a vector that represents a precondition for \( w \). A postcondition for \( P \) is any superset of \( \hat{W}P \); the weakest precondition is \( \hat{W}P \).

**Proposition.** Let \( w \) be a while loop on space \( S \) and let \( R \) be an invariant relation for \( w \) and \( P \) be a precondition for \( w \); the vector \( (\overline{T} \cap \hat{R}P) \) is a postcondition of \( w \) with respect to \( P \).
Invariant Relations and Predicate Transformers

Analysis Step:

\[\text{simplify}[\neg t(s)) \land (\exists s'': R(s'',s) \land p(s''))].\]

Taking the intersection of enough postconditions: we can find the strongest postcondition.

Each invariant relation produces a postcondition.
Invariant Relations and Correctness

Illustration:

- **S:** $n, f, k$: natural; $1 \leq k \leq n+1$.

- **w:** while ($k \neq n+1$) { $f=f\times k; k=k+1;$ }

- **R** = $\left\{ (s,s') \mid \frac{f}{(k-1)!} = \frac{f'}{(k'-1)!} \right\}$.

- **P** = $\{(s,s') \mid f = (k-1)! \}$. 
INVARIANT RELATIONS AND CORRECTNESS

\[ Q = (\overline{T} \cap \hat{RP}) \]
\[ = \{(s, s')|k = n + 1 \land \exists s'': \frac{f}{(k - 1)!} = \frac{f''}{(k'' - 1)!} \land f'' = (k'' - 1)!\} \]
\[ = \{(s, s')|k = n + 1 \land f = n!\}. \]
Invariant Relations and Predicate Transformers

Computing Weakest Preconditions.

Definition. We consider a while loop, \( w: \text{while} \ t \ \text{do} \ b \), and we let \( Q \) be a vector that represents a postcondition for \( w \). A precondition for \( Q \) is any subset of \( WQ \); the weakest precondition is \( WQ \).

We have no way to approximate weakest preconditions: either we have enough invariant relations to compute \( W \), or no deal.

Analysis Step:
\[ \text{simplify}[(\exists s': W(s,s') \land q(s'))]. \]
Invariant Relations and Correctness

Illustration:

- **S:** $n, f, k$: natural; $1 \leq k \leq n+1$.

- **w:** while $(k \neq n+1)$ \{ $f = f \times k; \ k = k + 1;$ \}

- **R:**
  $$R = \left\{(s, s') \mid \frac{f}{(k-1)!} = \frac{f'}{(k'-1)!}\right\}.$$

- **Q:**
  $$Q = \{(s, s') \mid f = n!\}.$$


**Invariant Relations and Correctness**

\[
WQ = \{(s, s') : \exists s': n' = n \land k' = n + 1 \land f' = n! \times \frac{f}{(k-1)!} \land f' = n'!\}
\]

\[
= \{(s, s') : f = (k - 1)!\}.
\]
INVARIANT RELATIONS AND CORRECTNESS

demo
PLAN

• Summary of Last Week
• Correctness and Refinement
• Program Semantics
• Invariant Relations and Termination
• Invariant Relations and Correctness
• Invariant Relations and Predicate Transformers
• Conclusion: Summary and Prospects
• References
CONCLUSION: SUMMARY AND PROSPECTS

Given an Invariant relation R of while loop w,

- \( R \cap \hat{T} \) represents a necessary condition of termination,
- \( \hat{R}P \) is an approximation (superset) of \( W \),
- \( \hat{R}P \) is an invariant assertion for precondition \( P \),
- \( WQ \) is a weakest precondition of \( w \) for postcondition \( Q \),
- \( \hat{R}P \cap \hat{T} \) is a postcondition for precondition \( P \),
- \( (R \cap U)\hat{T} = L \) is a necessary condition of correctness,
- \( R\hat{T} \cap CL \cap (C \cup R \cap \hat{T}) = C \) is a sufficient condition of correctness,
- \( \pi_R \) if \( R \) is symmetric, is an invariant function for \( w \).
CONCLUSION: SUMMARY AND PROSPECTS

- Automated Tool for generating invariant relations from static analysis of the source code.
- Relational Formulas are represented in the syntax of Mathematica (© Wolfram Research).
- Mathematica enables us to simplify expressions and solve equations.
- Performance of invariant relations generation dependent on recognizer database.
- Prospects include migration from syntactic to semantic matching, production of a streamlined recognizer database, production of domain dependent recognizer databases.
PLAN

• Summary of Last Week
• Correctness and Refinement
• Program Semantics
• Invariant Relations and Termination
• Invariant Relations and Correctness
• Invariant Relations and Predicate Transformers
• Conclusion: Summary and Prospects
• References
REFERENCES


REFERENCES


PLAN

• Summary of Last Week
• Correctness and Refinement
• Program Semantics
• Invariant Relations and Termination
• Invariant Relations and Correctness
• Invariant Relations and Predicate Transformers
• Conclusion: Summary and Prospects
• References