Free Variables: Terms

Informally: free variables of an expression are variables that have an occurrence in an expression that is not bound. Written $fv(e)$ for expression $e$.

Free variables of terms defined by structural induction over terms; written

- $fv(x) = \{x\}$
- $fv(t_1, . . . , t_n) = \bigcup_{i=1,...,n} fv(t_i)$

Note:
- Free variables of term just variables occurring in term; no bound variables
- No free variables in constants
- Example: $fv(add(1, abs(x))) = \{x\}$

Free Variables: Formulae

Defined by structural induction on formulae; uses $fv$ on terms

- $fv(true) = fv(false) = \{\}$
- $fv(\neg \phi) = \bigcup_{i=1}^{3} fv(\psi_i)$
- $fv(\phi \land \psi) = fv(\phi) \cup fv(\psi)$
- $fv(\phi \lor \psi) = fv(\phi) \cup fv(\psi)$
- $fv(\exists x. \phi) = \{x\}$
- $fv(\forall x. \phi) = \emptyset$

Example:

$$\forall x. (\exists y. (z \geq (y - x)) \lor (z \geq y)) = \{x, z\}$$

Variable occurrence at quantifiers are binding occurrences.

Occurrence that is not free and not binding is a bound occurrence.

Syntactic Substitution versus Assignment Update

- When interpreting universal quantification ($\forall x. \phi$), wanted to check interpretation of every instance of $\phi$ where $\phi$ was replaced by element of semantic domain $D$.
- How: semantically - interpret $\phi$ with assignment updated by $\phi \mapsto d$ for every $d \in D$.
- Syntactically?
- Answer: substitution.

Substitution in Terms

- Substitution of term $t$ for variable $x$ in term $s$ (written $s[t/x]$) gotten by replacing every instance of $x$ in $s$ by $t$
- $x$ called redex; $t$ called residue
- Yields instance of $s$

Formally defined by structural induction on terms:

- $x[t/x] = t$
- $y[t/x] = y$ for variable $y$ where $y \neq x$
- $f(t_1, . . . , t_n)[t/x] = f(t_1[t/x], . . . , t_n[t/x])$

Example:

$$(add(1, abs(x))[add(x, y)/x] = add(1, abs(add(x, y))))$$
Substitution in Formulae: Two Approaches

- Want to define by structural induction, similar to terms
- Quantifiers must be handled with care
  - Substitution only replaces free occurrences of variable
  - Need to avoid free variable capture
- Define the swapping of two variables in a term $t[x \leftrightarrow y]$ by structural induction on $t$...
- Examples
  - $(x > 3 \land (\exists y. (z \geq (y - x))) \lor (z \geq y))$ not defined
  - $(x > 3 \land (\exists y. (z \geq (y - x))) \land ((x + y) \geq y))$

Substitution in Formulae

- When quantifier would capture free variable of redex, can’t substitute in formula as is
- Solution 1: Make substitution partial function – undefined in this case
- Solution 2: Define equivalence relation based on renaming bound variables; define substitution on equivalence classes
- Will take Solution 1 here
- Still need definition of equivalence up to renaming bound variables

Substitution in Formulae: Problems

- Defined by structural induction; uses substitution in terms
- Read equations below as saying left is not defined if any expression on right not defined
  - true$t/x$ = true
  - false$t/x$ = false
  - $r;(t_1, \ldots, t_n)[t/x] = r((t_1)[t/x], \ldots, t_n[t/x])$
  - $(\psi)[t/x] = (\psi[t/x])$ for $\psi \in \{\land, \lor, \Rightarrow\}$
  - $(\exists x. \psi)[t/x] = Q x. \psi$ for $Q \in \{\forall, \exists\}$
  - $(\forall x. \psi)[t/x] = Q y. (\psi[t/x])$ if $x \neq y$ and $y \not\in \text{fv}(t)$ for $Q \in \{\forall, \exists\}$
  - $(\forall x. \psi)[t/x]$ not defined if $x \neq y$ and $y \in \text{fv}(t)$ for $Q \in \{\forall, \exists\}$

Substitution in Formulae

- Examples
  - $(x > 3 \land (\exists y. (z \geq (y - x))) \lor (z \geq y))[x + y/z]$

Theorem

Assume given structure $S = (G, \mathcal{D}, \mathcal{F}, \varnothing, \gamma, \rho)$, variable $x$, terms $s$ and $t$ over $G$, and a assignment. Let $b = a[x \mapsto T_a(t)]$. Then $T_a(s[t/x]) = T_b(s)$.

Renaming by Swapping: Terms

Define the swapping of two variables in a term $t[x \leftrightarrow y]$ by structural induction on terms:

- $x[x \leftrightarrow y] = y$ and $y[x \leftrightarrow y] = x$
- $z[x \leftrightarrow y] = z$ for $z$ a variable, $z \neq x, z \neq y$
- $f(t_1, \ldots, t_n)[x \leftrightarrow y] = f(t_1[x \leftrightarrow y], \ldots, t_n[x \leftrightarrow y])$

Examples:

- $\text{add}(1, \text{abs}(\text{add}(x, y)))[x \leftrightarrow y] = \text{add}(1, \text{abs}(\text{add}(y, x)))$
- $\text{add}(1, \text{abs}(\text{add}(x, y)))[x \leftrightarrow z] = \text{add}(1, \text{abs}(\text{add}(z, y)))$
Renaming by Swapping: Terms

Theorem
Assume given structure \( S = (G, D, F, \phi, R, p) \), variables \( x \) and \( y \), term \( t \) over \( G \), and a assignment. Let \( b = a(x \mapsto a(y))[y \mapsto a(x)] \). Then
\[ T_a(t[x \leftrightarrow y]) = T_a(t) \]

Proof.
By structural induction on terms, suffices to show theorem for the case where \( t \) variable, and case \( t = f(t_1, \ldots, t_n) \), assuming result for \( t_1, \ldots, t_n \)
- Case: \( t \) variable
  - Subcase: \( t = x \). Then \( T_a(x[x \leftrightarrow y]) = T_a(y) \) and
    \( T_a(x) = b(x) = a(x \mapsto a(y))[y \mapsto a(x)][x \mapsto T_a(y)][x] = a(y) \)
    so \( T_a(x[x \leftrightarrow y]) = T_a(t) \)
  - Subcase: \( t = y \). Then \( T_a(y[x \leftrightarrow y]) = T_a(x) \) and
    \( T_a(y) = b(y) = a(x \mapsto a(y))[y \mapsto a(x)][x] = a(x) \)
    so \( T_a(y[x \leftrightarrow y]) = T_a(t) \)
- Subcase: \( t \) variable, \( z \neq x \) and \( z \neq y \). Then
  \[ T_a(t[x \leftrightarrow y]) = T_a(z) \] and
  \[ T_a(z) = b(z) = a(x \mapsto a(y))[y \mapsto a(x)][z] = a(z) \]
  so \( T_a(t[x \leftrightarrow y]) = T_a(t) \)

Renaming by Swapping: Formulae

Theorem
Assume given structure \( S = (G, D, F, \phi, R, p) \), variables \( x \) and \( y \), formula \( \psi \) over \( G \), and a assignment. If \( x \not\in \text{fv}(t) \) and \( y \not\in \text{fv}(t) \) then
\[ \psi[x \leftrightarrow y] \equiv \psi \]

Proof.
Define the swapping of two variables in a formula \( \psi[x \leftrightarrow y] \) by structural induction, using swapping on terms:
- true\( [x \leftrightarrow y] \equiv \) true \ false\( [x \leftrightarrow y] \equiv \) false
- \( \forall t_1, \ldots, t_n [x \leftrightarrow y] = \forall t_1, \ldots, t_n [x \leftrightarrow y] \)
- \( (\psi) [x \leftrightarrow y] = (\psi) [x \leftrightarrow y] \) \( \neg(\psi) [x \leftrightarrow y] = \neg(\psi) [x \leftrightarrow y] \)
- \( (\psi_1 \circ \psi_2) [x \leftrightarrow y] = (\psi_1 [x \leftrightarrow y]) \circ (\psi_2 [x \leftrightarrow y]) \) for
  - \( \circ \in \{\wedge, \vee, \Rightarrow, \Leftarrow\} \)
- \( (Q \psi) [x \leftrightarrow y] = (Q \psi) [x \leftrightarrow y] \) for \( Q \in \{\forall, \exists\} \)
- \( (Q \psi) [x \leftrightarrow y] = (Q \psi) [x \leftrightarrow y] \) for \( Q \in \{\forall, \exists\} \)
- \( (Q z \psi) [x \leftrightarrow y] = (Q z \psi) [x \leftrightarrow y] \) for \( Q \in \{\forall, \exists\} \)

\( \alpha \)-equivalence

Examples
\[
\begin{align*}
(x > 3 \land (\exists y. (\forall z. z \geq (y - x)) \land (z \geq y)))[x \leftrightarrow y]
&= (y > 3 \land (\exists x. (\forall z. z \geq (x - y)) \land (z \geq x)))
\end{align*}
\]

\[
\begin{align*}
(x > 3 \land (\exists y. (\forall z. z \geq (y - x)) \land (z \geq y)))[y \leftrightarrow z]
&= (x > 3 \land (\exists y. (\forall z. z \geq (x - y)) \land (z \geq x)))[y \leftrightarrow w]
\end{align*}
\]
\( \alpha \)-equivalence: Example

\[
(x > 3 \land (\exists y. (\forall z. z \geq (y - x)) \lor (z \geq y))) \\
\equiv (x > 3 \land (\exists w. (\forall z. z \geq (w - x)) \lor (z \geq w))) \\
\equiv (x > 3 \land (\exists y. (\forall y. y \geq (w - x)) \lor (z \geq w)))
\]

Example

Show

\[
\{ \} \vdash (\exists x. \forall y. x \leq y) \Rightarrow (\forall x. \exists y. y \leq x)
\]

Example

Show

\[
\{ \exists x. \forall y. x \leq y \} \vdash \exists y. y \leq x \\
(\exists x. \forall y. x \leq y) \vdash \forall x. \exists y. y \leq x \\
\{ \} \vdash (\exists x. \forall y. x \leq y) 
\]

Example

Show

\[
\{ \exists y. y \leq x \} \vdash \exists x. \forall y. x \leq y \\
(\exists x. \forall y. x \leq y) \vdash \exists y. y \leq x \\
\{ \} \vdash (\exists x. \forall y. x \leq y) 
\]

Proof Rules

Will give Sequent version of Natural Deduction rules
All rules from Propositional Logic included

\[
\Gamma \vdash \psi[t/x] \quad \text{Ex I} \\
\Gamma \vdash \exists x. \psi \quad \text{Ex E} \\
\text{provided } \psi \equiv \psi' \\
\Gamma \vdash \forall x. \psi \quad \text{All I} \\
\Gamma \vdash \exists x. \forall y. \psi \quad \text{All E} \\
\text{provided } \psi \equiv \psi' \\
\Gamma \vdash \psi[y/x] \\
\Gamma \vdash \forall x. \psi \\
\text{provided } y \notin \{\psi(x) \cup \{x\}\} \cup \{\psi'\}
\]

Example

Show

\[
\{ \exists x. \forall y. x \leq y \} \vdash \forall x. \exists y. y \leq x \quad \text{Imp I} \\
\{ \} \vdash (\exists x. \forall y. x \leq y) 
\]

Example

Show

\[
\{ \exists x. \forall y. x \leq y \} \vdash \exists y. y \leq x \\
(\exists x. \forall y. x \leq y) \vdash \forall x. \exists y. y \leq x \\
\{ \} \vdash (\exists x. \forall y. x \leq y) 
\]

Example

Show

\[
\{ \exists x. \forall y. x \leq y \} \vdash \exists y. y \leq x \\
(\exists x. \forall y. x \leq y) \vdash \forall x. \exists y. y \leq x \\
\{ \} \vdash (\exists x. \forall y. x \leq y) 
\]
Example

Show

\[
\{\exists x.\forall y. x \leq y\} \vdash \exists x.\forall y. x \leq y
\]

Hyp

\[
\{\exists y. y \leq z; \forall y. z \leq y; x \leq y\} \vdash \exists y. y \leq x
\]

All E

\[
\{\exists y. y \leq z; \forall y. z \leq y; x \leq y\} \vdash \exists y. y \leq x
\]

Ex E

Example

Show

\[
\{\exists x.\forall y. x \leq y\} \vdash \exists x.\forall y. x \leq y
\]

Hyp

\[
\{\exists x.\forall y. x \leq y\} \vdash \exists x.\forall y. x \leq y
\]

All E

Example

Show

\[
\{\exists x.\forall y. x \leq y\} \vdash \exists x.\forall y. x \leq y
\]

Hyp

\[
\{\exists x.\forall y. x \leq y\} \vdash \exists x.\forall y. x \leq y
\]

All E

Example of Failure

Let's try to show

\[
\{\} \vdash (\forall x.3y. y \leq x) \Rightarrow (\exists x.\forall y. x \leq y)
\]
Example of Failure

Let's try to show

\[
\forall x. \exists y. y \leq x \vdash \exists x. \forall y. x \leq y
\]

Imp I

\{(x, y) \leq x \} \vdash (\exists x. \forall y. x \leq y)

Ex I

\{(x, y) \leq x \} \vdash \forall y. z \leq y

All E

\{(x, y) \leq x \} \vdash \forall y. z \leq y

Ex I

\{(x, y) \leq x \} \vdash (3x. \forall y. x \leq y)

Imp I

\{(x, y) \leq x \} \vdash (3x. \forall y. x \leq y)
**Example of Failure**

Let's try to show

\[
\text{Hyp} \{ \forall x. \exists y. y \leq x \} \vdash \forall x. \exists y. y \leq x
\]

\[
\text{Hyp} \{ \forall x. \exists y. y \leq x; \exists y. y \leq x \} \vdash \exists y. y \leq x
\]

\[
\text{All } E
\]

\[
\{ \forall x. \exists y. y \leq x \} \vdash \exists x. \forall y. x \leq y
\]

**Floyd-Hoare Logic**

- Also called **Axiomatic Semantics**
- Based on formal logic (first order predicate calculus)
- Logical system built from axioms and inference rules
- Mainly suited to simple imperative programming languages
- Ideas applicable quite broadly

**Floyd-Hoare Logic**

- **Goal**: Derive statements of form

\[
\{ P \} C \{ Q \}
\]

- \( P, Q \) logical statements about state, \( P \) precondition, \( Q \) postcondition, \( C \) program
- **Example**:

\[
\{ x = 1 \} \ x := x + 1 \ \{ x = 2 \}
\]

**Partial vs Total Correctness**

- An expression \( \{ P \} C \{ Q \} \) is a partial correctness statement
- For total correctness must also prove that \( C \) terminates (i.e. doesn't run forever)
  - Written: \( [P] C [Q] \)
- Will only consider partial correctness here
Simple Imperative Language

- We will give rules for simple imperative language

\[
\langle \text{command} \rangle ::= \langle \text{variable} \rangle := \langle \text{term} \rangle \\
\langle \text{command} \rangle ; \ldots ; \langle \text{command} \rangle \\
\text{if} \langle \text{statement} \rangle \text{then} \langle \text{command} \rangle \text{else} \langle \text{command} \rangle \\
\text{while} \langle \text{statement} \rangle \text{do} \langle \text{command} \rangle
\]

- Could add more features, like for-loops

Substitution

- Notation: \( P[e/v] \) \((\text{sometimes } P[v \rightarrow e])\)
- Meaning: Replace every \( v \) in \( P \) by \( e \)
- Example:

\[
(x + 2)[y − 1/x] = ((y − 1) + 2)
\]

The Assignment Rule

\[
\{P[e/x]\} x := e \{P\}
\]

Example:

\[
\{ ? \} x := y \{ x = 2 \}
\]

Example:

\[
\{ x = 2 \} x := y \{ x = 2 \}
\]

Example:

\[
\{ x = 2 \} x := y \{ x = 2 \}
\]

Examples:

\[
\{ y = 2 \} x := y \{ x = 2 \}
\]

\[
\{ y = 2 \} x := 2 \{ y = x \}
\]

\[
\{ x + 1 = n + 1 \} x := x + 1 \{ x = n + 1 \}
\]

\[
\{ 2 = 2 \} x := 2 \{ x = 2 \}
\]
Precondition Strengthening

\[(P \Rightarrow P') \land (P') \land (Q) \Rightarrow (P') \land (Q)\]

- **Meaning:** If we can show that \(P\) implies \(P'\) (i.e. \(P \Rightarrow P'\)) and we can show that \(P'\) and \(Q\), then we know that \(P\) implies \(Q\).
- \(P\) is **stronger** than \(P'\) means \(P \Rightarrow P'\).

Which Inferences Are Correct?

\[
\begin{align*}
\{x > 0 \land x < 5\} &\quad x := x + y \quad \{x + y = wx\} \\
\{x = 3\} &\quad x := x + y \quad \{x + y = wx\}
\end{align*}
\]

\[
\begin{align*}
\{x > 0 \land x < 5\} \land \{x = 3\} &\quad x := x + y \quad \{x + y = wx\} \quad \text{YES}
\end{align*}
\]

The Assignment Rule – Your Turn

- **What is the weakest precondition of**
  \[
  x := x + y \quad \{x + y = wx\}
  \]

Which Inferences Are Correct?

\[
\begin{align*}
\{x = 3\} &\quad x := x + y \quad \{x + y = wx\} \\
\{x > 0 \land x < 5\} &\quad x := x + y \quad \{x + y = wx\}
\end{align*}
\]

\[
\begin{align*}
\{x > 0 \land x < 5\} \land \{x = 3\} &\quad x := x + y \quad \{x + y = wx\} \quad \text{YES}
\end{align*}
\]

The Assignment Rule – Your Turn

- **What is the weakest precondition of**
  \[
  x := x + y \quad \{x + y = wx\}
  \]

Examples:

\[
\begin{align*}
{x > 0 \land x < 5} &\quad x := x \ast x \quad \{x < 25\} \\
{x = 3} &\quad x := x \ast x \quad \{x < 25\}
\end{align*}
\]

\[
\begin{align*}
{x > 0 \land x < 5} &\quad x := x \ast x \quad \{x < 25\} \quad \text{YES}
\end{align*}
\]

\[
\begin{align*}
{x > 0 \land x < 5} &\quad x := x \ast x \quad \{x < 25\}
\end{align*}
\]
### Which Inferences Are Correct?

<table>
<thead>
<tr>
<th>Inference</th>
<th>Correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{x &gt; 0 ∧ x &lt; 5} x := x ∗ x \{x &lt; 25\}</code></td>
<td>YES</td>
</tr>
<tr>
<td><code>{x = 3} x := x ∗ x \{x &lt; 25\}</code></td>
<td>YES</td>
</tr>
<tr>
<td><code>{x &gt; 0 ∧ x &lt; 5} x := x ∗ x \{x &lt; 25\}</code></td>
<td>NO</td>
</tr>
<tr>
<td><code>{x ∗ x &lt; 25} x := x ∗ x \{x &lt; 25\}</code></td>
<td>YES</td>
</tr>
<tr>
<td><code>{x &gt; 0 ∧ x &lt; 5} x := x ∗ x \{x &lt; 25\}</code></td>
<td>YES</td>
</tr>
<tr>
<td><code>{x = 3} x := x ∗ x \{x &lt; 25\}</code></td>
<td>NO</td>
</tr>
<tr>
<td><code>{x ∗ x &lt; 25} x := x ∗ x \{x &lt; 25\}</code></td>
<td>YES</td>
</tr>
</tbody>
</table>

Elsa L. Gunter  
CS477 Formal Software Dev Methods  
February 21, 2018  
35 / 35