5.1 Groundness Analysis

A variable $X$ is called ground w.r.t. a program execution if starting program entry and entering a program point $v$, if $X$ is bound to a variable-free term.

Goal:
- Find all variables which are ground whenever a particular program point is reached!
- Find all arguments of a predicate which are ground whenever the predicate is called!

Observation:
- In PuP, functions must be simulated through predicates.
- These then have designated input- and output parameters.
- Input parameters are those which are instantiated with a variable-free term whenever the predicate is called.
  These are also called ground.
- In the example, the first parameter of app is an input parameter.
- Unification with such a parameter can be implemented as pattern matching!
- Then we see that app in fact is deterministic !!!

Idea:
- Describe groundness by values from $B$:
  
  \[
  1 \quad \text{variable-free term;}
  
  0 \quad \text{term which contains variables.}
  \]
- A set of variable assignments is described by Boolean functions $\Rightarrow$
  
  \[
  X \leftrightarrow Y \quad \text{if } Y \text{ is ground.}
  
  X \wedge Y \quad \text{if } X \text{ and } Y \text{ are ground.}
  \]
Idea:

- Describe groundness by values from $B$:
  
  $1 \quad \Rightarrow$ variable-free term; 
  
  $0 \quad \Rightarrow$ term which contains variables. 

- A set of variable assignments is described by Boolean functions $:\rightarrow$

  $X \leftrightarrow Y \quad \Rightarrow X$ is ground iff $Y$ is ground. 

  $X \land Y \quad \Rightarrow X$ and $Y$ are ground. 

Idea (cont.):

- The constant function $0$ denotes an unreachable program point. 

- Occurring sets of variable assignments are closed under substitution.
  
  This means that for every occurring function $\phi \neq 0$, 

  $$\phi(1, \ldots, 1) = 1$$

  These functions are called positive. 

- The set of all positive functions is called $\text{Pos}$. 

  Ordering: $\phi_1 \sqsubseteq \phi_2$ if $\phi_1 \Rightarrow \phi_2$. 

- In particular, the least element is $0 \quad \rightarrow$
Example:

\[
\begin{array}{c}
00, 01, 10, 11 \\
1 \\
00, 10, 11 \\
01, 10, 11 \\
X \rightarrow Y \\
X \lor Y \\
X \rightarrow Y \\
X \leftrightarrow Y \\
11 \\
X \land Y \\
0 \\
01, 11 \\
\end{array}
\]

Remarks:

- Not all positive functions are monotonic !!!
- For \( k \) variables, there are \( 2^{2^k-1} + 1 \) many functions.
- The height of the complete lattice is \( 2^k \).
- We construct an interprocedural analysis which for every predicate \( p \) determines a (monotonic) transformation

\[ [p]^2 : \text{Pos} \rightarrow \text{Pos} \]

- For every clause, \( p(X_1, \ldots, X_k) \Leftarrow g_1, \ldots, g_n \) we obtain the constraint:

\[ [p]^2 \psi \equiv \exists X_{k+1}, \ldots, X_m, [g_1]^2 \left( \ldots \left[ g_n]^2 \psi \right) \ldots \right) \]

// \( m \) number of clause variables

Abstract Unification:

\[ [X = t]^2 \psi = \psi \land (X \leftrightarrow X_1 \land \ldots \land X_k) \]

if \( \text{Vars}(t) = \{X_1, \ldots, X_k\} \).

Abstract Literal:

\[ [q(s_1, \ldots, s_k)]^2 \psi = \text{combine}^2_{s_1, \ldots, s_k}(\psi, [g]^2 \left( \text{enter}^2_{s_1, \ldots, s_k} \psi \right)) \]

// analogous to procedure call !!!
Thereby:

\[
\text{enter}_{s_1, \ldots, s_k}^t \psi = \text{ren} (\exists X_1, \ldots, X_m. [\overline{X}_1 = s_1, \ldots, \overline{X}_k = s_k]^t \psi)
\]

\[
\text{combine}_{s_1, \ldots, s_k}(\psi, \psi_1) = \exists X_1, \ldots, X_r. \psi \land [\overline{X}_1 = s_1, \ldots, \overline{X}_k = s_k]^t (\text{ren} \psi_1)
\]

where

\[
\exists X. \phi = \phi[0/X] \lor \phi[1/X]
\]

\[
\text{ren} \phi = \phi[X_1/\overline{X}_1, \ldots, X_k/\overline{X}_k]
\]

\[
\text{ren} \phi = \phi[X_1/\overline{X}_1, \ldots, X_r/\overline{X}_r]
\]

---

Abstract Unification:

\[
[X = t]^t \psi = \psi \land (X \leftrightarrow X_1 \land \ldots \land X_n)
\]

if \( \text{Vars}(t) = \{X_1, \ldots, X_n\} \).

Abstract Literal:

\[
[q(s_1, \ldots, s_k)]^t \psi = \text{combine}_{s_1, \ldots, s_k}(\psi, [q]^t (\text{enter}_{s_1, \ldots, s_k}^t \psi))
\]

// analogous to procedure call !!

---

Example:

\[
\text{app}(X, Y, Z) \leftarrow X = [], Y = Z
\]

\[
\text{app}(X, Y, Z) \leftarrow X = [H|X'], Z = [H|Z'], \text{app}(X', Y, Z')
\]

Then

\[
[\text{app}]^t(X) \equiv X \land (Y \leftrightarrow Z)
\]

\[
[\text{app}]^t(X) \equiv \text{let } \psi = X \land H \land X' \land (Z \leftrightarrow Z') \in \exists \overline{H}, \overline{X}, \overline{Z}, \text{combine}_{\overline{H}, \overline{X}, \overline{Z}}(\psi, [\text{app}]^t(\text{enter}_{\overline{H}, \overline{X}, \overline{Z}}(\psi)))
\]

where for \( \psi = X \land H \land X' \land (Z \leftrightarrow Z') \):

\[
\text{enter}_{\overline{H}, \overline{X}, \overline{Z}}(\psi) = X
\]

\[
\text{combine}_{\overline{H}, \overline{X}, \overline{Z}}(\psi, X \land (Y \leftrightarrow Z)) = (X \land H \land X' \land (Z \leftrightarrow Z')) \land (Y \leftrightarrow Z')
\]
Remarks:

- Not all positive functions are monotonic !!!
- For $k$ variables, there are $2^{2^k} - 1$ many functions.
- The height of the complete lattice is $2^k$.
- We construct an interprocedural analysis which for every predicate $p$ determines a (monotonic) transformation

$$[p]^2 : \text{Pos} \to \text{Pos}$$

- For every clause, $p(x_1, \ldots, x_n) \iff g_1, \ldots, g_n$ we obtain the constraint:

$$[p]^2 \psi \equiv \exists x_{k+1}, \ldots, x_m. [g_n]^2 \cdots ( [g_1]^2 \psi) \cdots$$

// $m$ number of clause variables

Example (Cont.):

Furthermore,

$$[\text{app}]^1(Z) \equiv X \land Y \land Z$$

$$[\text{app}]^1(Z) \equiv \text{let } \psi = X \land H \land X' \land Z \land Z' \text{ in } \exists H, X', Z'. \text{combine}_2^2 (\psi, [\text{app}]^2(\text{enter}_2^2(\psi)))$$

where for $\psi = Z \land H \land Z' \land (X \leftrightarrow X')$:

$$\text{enter}_2^2(\psi) = Z$$

$$\text{combine}_2^2 (\psi, X \land Y \land Z) = X \land H \land X' \land Y \land Z \land Z'$$

Fixpoint iteration therefore yields:

$$[\text{app}]^2(X) = X \land (Y \leftrightarrow Z)$$

$$[\text{app}]^2(Z) = X \land Y \land Z$$

Example (Cont.):

Furthermore,

$$[\text{app}]^1(Z) \equiv X \land Y \land Z$$

$$[\text{app}]^1(Z) \equiv \text{let } \psi = X \land H \land X' \land Z \land Z' \text{ in } \exists H, X', Z'. \text{combine}_2^2 (\psi, [\text{app}]^2(\text{enter}_2^2(\psi)))$$

where for $\psi = Z \land H \land Z' \land (X \leftrightarrow X')$:

$$\text{enter}_2^2(\psi) = Z$$

$$\text{combine}_2^2 (\psi, X \land Y \land Z) = X \land H \land X' \land Y \land Z \land Z'$$

Fixpoint iteration therefore yields:

$$[\text{app}]^2(X) = X \land (Y \leftrightarrow Z)$$

$$[\text{app}]^2(Z) = X \land Y \land Z$$
Example:

\[
\text{app}(X, Y, Z) \leftarrow X = [], Y = Z \\
\text{app}(X, Y, Z) \leftarrow X = [H|X'], Z = [H|Z'], \text{app}(X', Y, Z')
\]

Then

\[
[\text{app}]^2(X) \equiv X \land (Y \leftrightarrow Z) \\
[\text{app}]^3(X) \equiv \text{let } \psi = X \land H \land X' \land (Z \leftrightarrow Z') \\
in \exists H, X', Z'. \text{combine}_2^3(\psi, [\text{app}]^2(\text{enter}_1^2(\psi)))
\]

where for \( \psi = X \land H \land X' \land (Z \leftrightarrow Z'):

\[
\text{enter}_1^4(\psi) = X \\
\text{combine}_2^4(\psi, X \land (Y \leftrightarrow Z)) = (X \land H \land X' \land (Z \leftrightarrow Z') \land (Y \leftrightarrow Z'))
\]

Example (Cont.):

Furthermore,

\[
[\text{app}]^1(Z) \equiv X \land Y \land Z \\
[\text{app}]^2(Z) \equiv \text{let } \psi = X \land H \land X' \land Z \land Z' \\
in \exists H, X', Z'. \text{combine}_2^3(\psi, [\text{app}]^2(\text{enter}_1^2(\psi)))
\]

where for \( \psi = Z \land H \land Z' \land (X \leftrightarrow X'):

\[
\text{enter}_1^4(\psi) = Z \\
\text{combine}_2^4(\psi, X \land Y \land Z) = X \land H \land X' \land Y \land Z \land Z'
\]

Fixpoint iteration therefore yields:

\[
[\text{app}]^2(X) = X \land (Y \leftrightarrow Z) \\
[\text{app}]^2(Z) = X \land Y \land Z
\]

Discussion:

- Exhaustive tabulation of the transformation \([\text{app}]^2\) is not feasible.
- Therefore, we rely on demand-driven fixpoint iteration!
- The evaluation starts with the evaluation of the query \( q \), i.e., with the evaluation of \([q]^2 1\).
- The set of inspected fixpoint variables \([p]^2 \psi \) yields a description of all possible calls \((-\))
- For an efficient representation of functions \( \psi \in \text{Pos} \) we rely on binary decision diagrams (BDDs).
Background 6:  Binary Decision Diagrams

Idea (1):

- Choose an ordering \( x_1, \ldots, x_k \) on the arguments ...
- Represent the function \( f : \mathbb{B} \rightarrow \ldots \rightarrow \mathbb{B} \) by \([f]_D \) where:
  \[
  [b]_k = b \\
  [f]_{i-1} = \text{fun } x_i \rightarrow \text{if } x_i \text{ then } [f]_i, \text{ else } [f]_i, \]

Example: \( f(x_1, x_2, x_3) = x_1 \land (x_2 \leftrightarrow x_3) \)

... yields the tree:

Idea (2):

- Decision trees are exponentially large \(-(-)
- Often, however, many sub-trees are isomorphic \(-(-)
- Isomorphic sub-trees need to be represented only once ...

... yields the tree:
Discussion:

- This representation of the Boolean function $f$ is unique!

Equality of functions is efficiently decidable!!

- For the representation to be useful, it should support the basic operations: $\land, \lor, \neg, \exists x_j$ ...

\[
[b_1 \land b_2]_i = b_1 \land b_2 \\
[f \land g]_{i-1} = \text{fun } x_i \rightarrow \text{if } x_i \text{ then } [f \land g]_1_i, \\
\text{else } [f \land g]_i
\]

// analogous for the remaining operators

884

885

- Operations are executed bottom-up
- Root nodes of already constructed sub-graphs are stored in a unique-table

Isomorphy can be tested in constant time!
- The operations thus are polynomial in the size of the input BDDs :-)

886

Example: $(x_1 \leftrightarrow x_2) \land (x_3 \leftrightarrow x_4)$

887
Discussion (3):

- Not all Boolean functions have small BDDs :-(
- Difficult functions:
  - multiplication;
  - indirect addressing ...

⇒⇒⇒ data-intensive programs cannot be analyzed in this way :-(

Discussion (3):

- Not all Boolean functions have small BDDs :-(
- Difficult functions:
  - multiplication;
  - indirect addressing ...

⇒⇒⇒ data-intensive programs cannot be analyzed in this way :-(