Towards Decidability of Freeness

Sebastian Danicic

February 2, 2007

Freeness

A Schema is Free if and only if it has no repeating predicate terms.

Freeness

A Schema is not free if and only if it has repeating predicate terms.

Decidability of Freeness

In order to decide freeness we look for repeating predicate terms. If we find one then it's not free and if there isn't one then it's free.

Decidability of Freeness

We decided to consider simple case first.
The simplest case is schemas with no variables.

Schemas with no variables

Lemma

A schema with no variables is free if and only if it contains no loops.

Decidability of Freeness for Schemas with no variable

Lemma

Freeness is decidable for schemas with no variables.

Schemas with exactly one variable

Lemma

A schema with one variable is free if and only if for every loop body S

- *Every path through S contains a non–constant assignment to the variable.*
- *No path through S contains a constant assignment to the variable.*

Schemas with exactly one variable

Alternatively, thinking of a path as a state function:

Lemma

A schema with one variable, x is free if and only if for every loop body, S every path through S maps x to a 'proper' term containing x.

Schemas with exactly one variable

Alternatively, thinking of a path as a state function:

Lemma

A schema with one variable, x is free if and only if for every loop body, S every path through S maps x to a 'proper' term containing x.

A proper term is a term that isn't a variable

Decidability of Freeness of Schemas with exactly one variable

Lemma

Freeness is decidable for schemas with exactly one variable.

Freeness of a particular predicate symbol p

Definition

A predicate p is free if and only if there are no paths which give rise to repeated predicate terms containing p.

Repeating Sets of Variables at point p

Definition

A set of variables, V repeats at point p means there is a path where all the variables in V have the same value at more than one occurrence of p.

Freeness

Lemma

A Schema is free if and only if it is free with respect to all its predicate symbols.

Freeness

Lemma

A predicate $p(V)$ is free if and only if the set of variables V does not repeat at p.

p-Cycles

Definition

Let p be a predicate or function symbol. A p-cycle is a path from p to p containing no intermediate occurrences of p.

Repeating Lemma

Lemma

Variable set V repeats at p if and only if there is a composition of p-cycles whose state function is σ, say, such that $(\sigma \circ \sigma) \restriction V = \sigma \restriction V$.

An Informal Idea

- Represent the schema as a "labelled directed graph" where the nodes are the predicates and the arcs are labelled with the 'variable set abstracted' state functions which takes us from one predicate to the next.

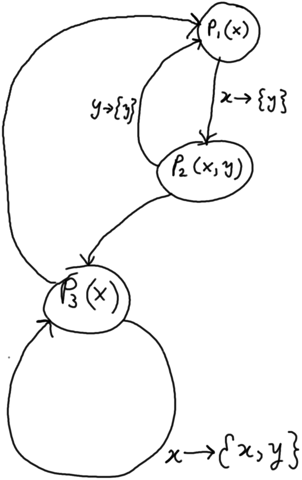
An Informal Idea

- Represent the schema as a "labelled directed graph" where the nodes are the predicates and the arcs are labelled with the 'variable set abstracted' state functions which takes us from one predicate to the next.
- Compute the *"closure"* of this graph.

Example

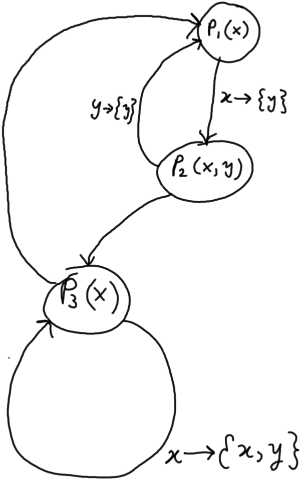
```
while p1(x)
{
   x=f(y);
   if p2(x,y)
      y=g(y)
   else while p3(x)
         {
            x=h(x,y)
         }
}
```

Example




```
while p1(x)
{
   x=f(y);
   if p2(x,y)
      y=g(y)
   else while p3(x)
         {
            x=h(x,y)
         }
}
```

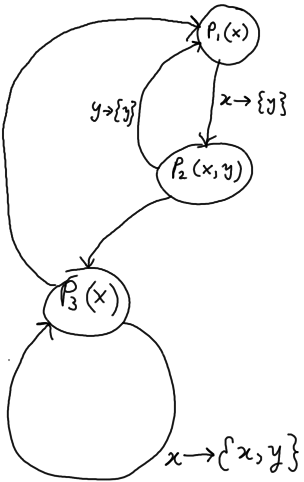
Example




```
while p1(x)
{
   x=f(y);
   if p2(x,y)
      y=g(y)
   else while p3(x)
         {
            x=h(x,y)
         }
}
```

Analyse the Graph

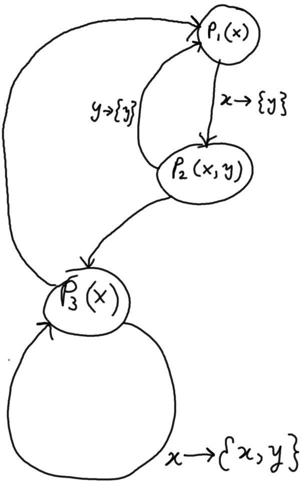
Example




```
while p1(x)
{
   x=f(y);
   if p2(x,y)
      y=g(y)
   else while p3(x)
        {
          x=h(x,y)
        }
}
```

Its not free because p1 repeats. (p1 p2 p3 p1)

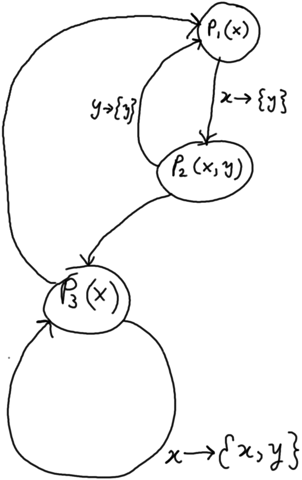
Example




```
while p1(x)
{
   x=f(y);
   if p2(x,y)
      y=g(y)
   else while p3(x)
         {
            x=h(x,y)
         }
}
```

p2 also repeats. (p2 p3 p1 p2)

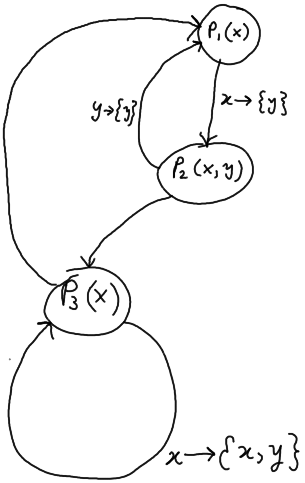
Example




```
while p1(x)
{
   x=f(y);
   if p2(x,y)
      y=g(y)
   else while p3(x)
         {
            x=h(x,y)
         }
}
```

anything else?

Example




```
while p1(x)
{
   x=f(y);
   if p2(x,y)
      y=g(y)
   else while p3(x)
        {
           x=h(x,y)
        }
}
```

p3(x) also repeats (p3 p1 p2 p3)

Constants

To decide whether p repeats we look at all the p-cycles. Constants are very useful:

Constants

To decide whether p repeats we look at all the p-cycles. Constants are very useful:

Definition

Given some state functions, $\{\sigma_i\}$, variable x is *constant variable* with respect to the $\{\sigma_i\}$ iff one of the σ_i maps x to a term with no variables or a term containing only variables which are mapped to themselves (unchanged) in all the $\{\sigma_i\}$.

Constants

To decide whether p repeats we look at all the p-cycles. Constants are very useful:

Definition

Given some state functions, $\{\sigma_i\}$, variable x is *constant variable* with respect to the $\{\sigma_i\}$ iff one of the σ_i maps x to a term with no variables or a term containing only variables which are mapped to themselves (unchanged) in all the $\{\sigma_i\}$.

Definition

Given some state functions, $\{\sigma_i\}$, term t is *constant* with respect to the $\{\sigma_i\}$ iff all the variables it contains are constant w.r.t the $\{\sigma_i\}$.

Repeating Claim

Lemma

Let $p(V)$ be a predicate.
p does not repeat if and only if there exists a composition of p-cycles which maps each v in V either to itself or to a constant.

Lemma

Let $p(V)$ be a predicate.
p does not repeat if and only if there exists a composition of p-cycles which maps each v in V either to itself or to a constant.

This one is important can we argue about it please?

Flattened Terms

Given a term t, we can define the flattened equivalence class $[t]$ to be the set of all terms which mention exactly the same symbols and variables as t.

Flattened Terms

Given a term t, we can define the flattened equivalence class $[t]$ to be the set of all terms which mention exactly the same symbols and variables as t.

Example of terms in the same flattened equivalence class:

Flattened Terms

Given a term t, we can define the flattened equivalence class $[t]$ to be the set of all terms which mention exactly the same symbols and variables as t.

Example of terms in the same flattened equivalence class:

f(x,y)

Flattened Terms

Given a term t, we can define the flattened equivalence class $[t]$ to be the set of all terms which mention exactly the same symbols and variables as t.

Example of terms in the same flattened equivalence class:

f(x,y)

f(f(x,y),y)

Flattened Terms

Given a term t, we can define the flattened equivalence class $[t]$ to be the set of all terms which mention exactly the same symbols and variables as t.

Example of terms in the same flattened equivalence class:

f(x,y)

f(f(x,y),y)

f(f(f(x,y),y),y)

Flattened Terms

Given a term t, we can define the flattened equivalence class $[t]$ to be the set of all terms which mention exactly the same symbols and variables as t.

Example of terms in the same flattened equivalence class:

f(x,y)
f(f(x,y),y)
f(f(f(x,y),y),y)
f(y,x)

Flattened States

Similarly, given any state σ, we can define the flattened equivalence class $[\sigma]$ to the set of all states τ such that for all variables v, $[\tau v] = [\sigma v]$.

Flattened States

Similarly, given any state σ, we can define the flattened equivalence class $[\sigma]$ to the set of all states τ such that for all variables v, $[\tau v] = [\sigma v]$.
Example of states in the same flattened equivalence class:

Flattened States

Similarly, given any state σ, we can define the flattened equivalence class $[\sigma]$ to the set of all states τ such that for all variables v, $[\tau v] = [\sigma v]$.
Example of states in the same flattened equivalence class:
$\{x \rightarrow f(x,y), y \rightarrow y\}$

Flattened States

Similarly, given any state σ, we can define the flattened equivalence class $[\sigma]$ to the set of all states τ such that for all variables v, $[\tau v] = [\sigma v]$.

Example of states in the same flattened equivalence class:

$\{x \to f(x, y), y \to y\}$

$\{x \to f(f(x, y), y), y \to y\}$

Flattened States

Similarly, given any state σ, we can define the flattened equivalence class $[\sigma]$ to the set of all states τ such that for all variables v, $[\tau v] = [\sigma v]$.
Example of states in the same flattened equivalence class:

$\{x \rightarrow f(x, y), y \rightarrow y\}$
$\{x \rightarrow f(f(x, y), y), y \rightarrow y\}$
$\{x \rightarrow (f(f(x, y), y), y \rightarrow y\}$

Composing Sets of States

Definition

Given two sets Σ_1, Σ_2 of states, we define

$$\Sigma_1 \circ \Sigma_2 = \{\sigma_1 \circ \sigma_2 |\ (\sigma_1, \sigma_2) \in \Sigma_1 \times \Sigma_2\}$$

Flattening is presevered by Composition

Lemma

$[\sigma] \circ [\tau] \subseteq [\sigma \circ \tau]$

Flattening is presevered by Composition

Lemma

$[\sigma] \circ [\tau] \subseteq [\sigma \circ \tau]$

So, when composing two states and flattening the result, any representative from the same equivalence class is as good as any other.

Flattening Conjecture

Lemma

If term t is constant with respect to $\{\sigma_1, \cdots, \sigma_n\}$ then for all τ_i in $[\sigma_i]$, t is constant with respect to $\{\tau_1, \cdots \tau_n\}$.

The Closure of a set of States

Definition

Let $\Sigma = \{\sigma_1, \cdots, \sigma_n\}$ be a set of states. Then Σ^* is the set of all possible compositions of the elements of Σ.

Lemma

Let $\Sigma = \{\sigma_1, \cdots, \sigma_n\}$ *be a set of states. Then there exists a finite set* S *of states such that* $[S] = [\Sigma^*]$.
We call S *a finite representation for* Σ^*.

Finite Representations for Finite Sets are Computable

Lemma

Let $\Sigma = \{\sigma_1, \cdots, \sigma_n\}$ be a set of states.
There exists an algorithm for finding a finite representation for Σ^*.

Finite Representations for Finite Sets are Computable

Lemma

Let $\Sigma = \{\sigma_1, \cdots, \sigma_n\}$ be a set of states.
There exists an algorithm for finding a finite representation for Σ^*.

i=1

0: list m=nil;

1:Generate all the compositions of length i

2: For each of these, add it to m if there isnt already a state in m which is equivalent to it.

3:i=i+1

4:go to 1

Finite Representations for Finite Sets are Computable

Lemma

Let $\Sigma = \{\sigma_1, \cdots, \sigma_n\}$ be a set of states.
There exists an algorithm for finding a finite representation for Σ^*.

i=1

0: list m=nil;

1:Generate all the compositions of length i

2: For each of these, add it to m if there isnt already a state in m which is equivalent to it.

3:i=i+1

4:go to 1

The proof follows because flattening is preserved by composition.

Flattened States Claim

Lemma

We only need to consider finitely many p-cycles to decide freeness.

Proof: Follows from the Repeating Claim and the Flattening Conjecture.

An Algorithm for Deciding Freeness

- Consider each predicate p in turn.

An Algorithm for Deciding Freeness

- Consider each predicate p in turn.
- For each predicate p, work out the finite representation for the set of states for each inner loop containing p.

An Algorithm for Deciding Freeness

- Consider each predicate p in turn.
- For each predicate p, work out the finite representation for the set of states for each inner loop containing p.
- Replace each inner loop with this finite representation and work outwards.

An Algorithm for Deciding Freeness

- Consider each predicate p in turn.
- For each predicate p, work out the finite representation for the set of states for each inner loop containing p.
- Replace each inner loop with this finite representation and work outwards.
- When there are no loops left we will end up with a finite representation for the p cycles.

An Algorithm for Deciding Freeness

- Consider each predicate p in turn.
- For each predicate p, work out the finite representation for the set of states for each inner loop containing p.
- Replace each inner loop with this finite representation and work outwards.
- When there are no loops left we will end up with a finite representation for the p cycles.
- Check whether the variables in p repeat.

The End!

Questions? Cuonter–examples?

Increases

Definition

A state function "increases x" if it maps x to a proper term containing x.

Lemma

Lemma

Let $p(V)$ be a predicate. If for all p-cycles, σ, σ increases v for some v in V, then p does not repeat.

Lemma

Let $p(V)$ be a predicate. If for all p-cycles, σ, σ increases v for some v in V, then p does not repeat.

Wrong! Consider:

```
while p1(x,y)
{
   if p2(x,y)
      y=g()
      x=f(x)
   else
      x=h()
      y=k(y)
}
```