Lecture 7
Loops and Recursion

Viktor Kuncak

Loops

Loops: Example

Consider the set of variables $V = \{x, y\}$ and this program L:

```
while (x > 0) {
  x = x - y
}
```

When the loop terminates, what is the (strongest) relation $\rho(L)$ between state (x, y) before loop started executing and the final state (x', y')?

Loops: Example

Consider the set of variables $V = \{x, y\}$ and this program L:

```
while (x > 0) {
  x = x - y
}
```

When the loop terminates, what is the (strongest) relation $\rho(L)$ between state (x, y) before loop started executing and the final state (x', y')?
Let k be the number of times loop executes.

Loops: Example

Consider the set of variables $V = \{x, y\}$ and this program L:

```
while (x > 0) {
  x = x - y
}
```

When the loop terminates, what is the (strongest) relation $\rho(L)$ between state (x, y) before loop started executing and the final state (x', y')?

Let k be the number of times loop executes.

- $k = 0$:

Loops: Example

Consider the set of variables $V = \{x, y\}$ and this program L:

```
while (x > 0) {
  x = x - y
}
```

When the loop terminates, what is the (strongest) relation $\rho(L)$ between state (x, y) before loop started executing and the final state (x', y')?
Let k be the number of times loop executes.

- $k = 0$: $x \leq 0 \land x' = x \land y' = y$

Loops: Example

Consider the set of variables $V = \{x, y\}$ and this program L:

```
while (x > 0) {
  x = x − y
}
```

When the loop terminates, what is the (strongest) relation $\rho(L)$ between state (x, y) before loop started executing and the final state (x', y')?
Let k be the number of times loop executes.

- $k = 0$: $x \leq 0 \wedge x' = x \wedge y' = y$
- $k = 1$:

Loops: Example

Consider the set of variables $V = \{x, y\}$ and this program L:

```
while (x > 0) {
  x = x − y
}
```

When the loop terminates, what is the (strongest) relation $\rho(L)$ between state (x, y) before loop started executing and the final state (x', y')?
Let k be the number of times loop executes.

- $k = 0$: $x \leq 0 \wedge x' = x \wedge y' = y$
- $k = 1$: $x > 0 \wedge x' = x - y \wedge y' = y \wedge x' \leq 0$

Loops: Example

Consider the set of variables $V = \{x, y\}$ and this program L:

```
while (x > 0) {
  x = x - y
}
```

When the loop terminates, what is the (strongest) relation $\rho(L)$ between state (x, y) before loop started executing and the final state (x', y')?
Let k be the number of times loop executes.

- $k = 0$: $x \leq 0 \land x' = x \land y' = y$
- $k = 1$: $x > 0 \land x' = x - y \land y' = y \land x' \leq 0$
- $k > 0$:

Loops: Example

Consider the set of variables $V = \{x, y\}$ and this program L:

```
while (x > 0) {
  x = x - y
}
```

When the loop terminates, what is the (strongest) relation $\rho(L)$ between state (x, y) before loop started executing and the final state (x', y')?

Let k be the number of times loop executes.

- $k = 0$: $x \leq 0 \land x' = x \land y' = y$
- $k = 1$: $x > 0 \land x' = x - y \land y' = y \land x' \leq 0$
- $k > 0$: $x > 0 \land x' = x - ky \land x' \leq 0 \land y' = y$

Solution:

$$
\begin{array}{l}
(x \leq 0 \land x' = x \land y' = y) \lor \\
(\exists k.\ k > 0 \land x > 0 \land x' = x - ky \land x' \leq 0 \land y' = y)
\end{array}
$$

Heuristically Eliminating a Quantifier from non-PA formula

$$\exists k.\ k > 0 \land x > 0 \land x' = x - ky \land x' \leq 0 \land y' = y$$

This implies $y > 0$.

Heuristically Eliminating a Quantifier from non-PA formula

$$\exists k.\ k > 0 \wedge x > 0 \wedge x' = x - ky \wedge x' \leq 0 \wedge y' = y$$

This implies $y > 0$.

$$\exists k.\ y > 0 \wedge k > 0 \wedge x > 0 \wedge ky = x - x' \wedge x' \leq 0 \wedge y' = y$$

Heuristically Eliminating a Quantifier from non-PA formula

$$\exists k.\ k > 0 \wedge x > 0 \wedge x' = x - ky \wedge x' \leq 0 \wedge y' = y$$

This implies $y > 0$.

$$\exists k.\ y > 0 \wedge k > 0 \wedge x > 0 \wedge ky = x - x' \wedge x' \leq 0 \wedge y' = y$$

$$\exists k.\ y > 0 \wedge k > 0 \wedge x > 0 \wedge y | (x - x') \wedge k = (x - x')/y \wedge x' \leq 0 \wedge y' = y$$

Heuristically Eliminating a Quantifier from non-PA formula

$$\exists k.\ k > 0 \wedge x > 0 \wedge x' = x - ky \wedge x' \leq 0 \wedge y' = y$$

This implies $y > 0$.

$$\exists k.\ y > 0 \wedge k > 0 \wedge x > 0 \wedge ky = x - x' \wedge x' \leq 0 \wedge y' = y$$

$$\exists k.\ y > 0 \wedge k > 0 \wedge x > 0 \wedge y | (x - x') \wedge k = (x - x')/y \wedge x' \leq 0 \wedge y' = y$$

$$y > 0 \wedge (x - x')/y > 0 \wedge x > 0 \wedge y | (x - x') \wedge x' \leq 0 \wedge y' = y$$

Heuristically Eliminating a Quantifier from non-PA formula

$$\exists k.\ k > 0 \land x > 0 \land x' = x - ky \land x' \leq 0 \land y' = y$$

This implies $y > 0$.

$$\exists k.\ y > 0 \land k > 0 \land x > 0 \land ky = x - x' \land x' \leq 0 \land y' = y$$

$$\exists k.\ y > 0 \land k > 0 \land x > 0 \land y | (x - x') \land k = (x - x')/y \land x' \leq 0 \land y' = y$$

$$y > 0 \land (x - x')/y > 0 \land x > 0 \land y | (x - x') \land x' \leq 0 \land y' = y$$

$$y > 0 \land x - x' > 0 \land x > 0 \land y | (x - x') \land x' \leq 0 \land y' = y$$

Integer Programs with Loops

Even if loop body is in Presburger arithmetic, the semantics of a loop need not be.

Integer programs with loops are Turing complete and can compute all computable functions.

Even if we cannot find Presburger arithmetic formula, we may be able to find

- a formula in a richer logic
- a property of the meaning of the loop
  (e.g. formula for the superset)

To help with these tasks, we give mathematical semantics of loops
Useful concept for this is transitive closure: $r^* = \bigcup_{n \geq 0} r^n$
(We may or may not have a general formula for r^n or r^*)

Towards meaning of loops: unfolding

Loops can describe an infinite number of basic paths
(for a larger input, program takes a longer path)
Consider loop

$$L \equiv \textbf{while}(F)c$$

We would like to have

$$\begin{array}{rcl} L & \equiv & \textbf{if}\,(F)\;(c\,;L) \\ & \equiv & \textbf{if}\,(F)\;(c\,;\textbf{if}\,(F)\;(c\,;L)) \end{array}$$

For $r_L = \rho(L)$, $r_c = \rho(c)$, $\Delta_f = \Delta_{S(F)}$, $\Delta_{nf} = \Delta_{S(\neg F)}$ we have

$$\begin{array}{rcl} r_L & = & (\Delta_f \circ r_c \circ r_L) \cup \Delta_{nf} \\ & = & (\Delta_f \circ r_c \circ ((\Delta_f \circ r_c \circ r_L) \cup \Delta_{nf})) \cup \Delta_{nf} \\ & = & \Delta_{nf} \cup \\ & & (\Delta_f \circ r_c) \circ \Delta_{nf} \cup \\ & & (\Delta_f \circ r_c)^2 \circ r_L \end{array}$$

Unfolding Loops

$$\begin{aligned} r_L \quad = \quad & \Delta_{nf} \cup \\ & (\Delta_f \circ r_c) \circ \Delta_{nf} \cup \\ & (\Delta_f \circ r_c)^2 \circ \Delta_{nf} \cup \\ & (\Delta_f \circ r_c)^3 \circ r_L \end{aligned}$$

We prove by induction that for every $n \geq 0$,

$$(\Delta_f \circ r_c)^n \circ \Delta_{nf} \subseteq r_L$$

So, $(\Delta_f \circ r_c)^* \circ \Delta_{nf} \subseteq r_L$.
We define r_L to be:

$$r_L = (\Delta_f \circ r_c)^* \circ \Delta_{nf}$$

THEREFORE:

$$\rho(\mathbf{while}(F)c) = (\Delta_{S(F)} \circ \rho(c))^* \circ \Delta_{S(\neg F)}$$

Using Loop Semantics in Example

ρ of L:

while (x $>$ 0) {
  x $=$ x $-$ y
}

is:

Using Loop Semantics in Example

ρ of L:

while (x > 0) {
  x = x − y
}

is:

$$(\Delta_{S(x>0)} \circ \rho(x = x - y))^* \circ \Delta_{S(\neg(x>0))}$$

Compute each relation:

$$\begin{aligned}
\Delta_{S(x>0)} &= \{((x,y),(x,y)) \mid x > 0\} \\
\Delta_{S(\neg(x>0))} &= \{((x,y),(x,y)) \mid x \leq 0\} \\
\rho(x = x - y) &= \{((x,y),(x-y,y)) \mid x,y \in \mathbb{Z}\} \\
\Delta_{S(x>0)} \circ \rho(x = x - y) &= \\
(\Delta_{S(x>0)} \circ \rho(x = x - y))^k &= \\
(\Delta_{S(x>0)} \circ \rho(x = x - y))^* &= \\
\rho(L) &=
\end{aligned}$$

Semantics of a Program with Loop

Compute and simplify relation for this program:

```
x = 0
while (y > 0) {
  x = x + y
  y = y - 1
}
```

$$\rho(x = 0) \circ$$
$$(\Delta_{S(y>0)} \circ \rho(x = x + y; y = y - 1))^* \circ$$
$$\Delta_{S(y \leq 0)}$$

Approximate Semantics of Loops

Instead of computing exact semantics, it can be sufficient to compute approximate semantics.

Observation: $r_1 \subseteq r_2 \to r_1^* \subseteq r_2^*$

Suppose we only wish to show that the semantics satisfies $y' \leq y$

```
x = 0
while (y > 0) {
  x = x + y
  y = y - 1
}
```

$\rho(x = 0)\circ$

$(\Delta_{S(y>0)} \circ \rho(x = x + y; y = y - 1))^*\circ$

$\Delta_{S(y\leq 0)}$

Recursion

Example of Recursion

For simplicity assume no parameters
(we can simulate them using global variables)

```
def f =
 if (x > 0) {
   if (x % 2 == 0) {
     x = x / 2;
     f;
     y = y * 2
   } else {
     x = x - 1;
     y = y + x;
     f
   }
 }
```

$$
\begin{aligned}
&E(r_f) = \\
&\Delta_{S(x>0)} \circ (\\
&\quad (\Delta_{x\%2=0} \circ \\
&\quad \rho(x = x/2) \circ \\
&\quad r_f \circ \\
&\quad \rho(y = y * 2)) \\
&\cup \\
&\quad (\Delta_{x\%2\neq 0} \circ \\
&\quad \rho(x = x - 1) \circ \\
&\quad \rho(y = y + x) \circ \\
&\quad r_f) \\
&) \cup \Delta_{S(x\leq 0)}
\end{aligned}
$$

Assume recursive function call denotes some relation r_f

Need to find relation r_f such that $r_f = E(r_f)$

Simpler Example of Recursion

```
def f =
  if (x > 0) {
    x = x − 1
    f
    y = y + 2
  }
```

$$
\begin{aligned}
E(r_f) = \ & \Big(\Delta_{S(x>0)} \circ \Big(\\
& \quad \rho(x = x - 1) \circ \\
& \quad r_f \circ \\
& \quad \rho(y = y + 2)\Big)\Big) \\
& \Big) \cup \Delta_{S(x \leq 0)}
\end{aligned}
$$

Simpler Example of Recursion

```
def f =
  if (x > 0) {
    x = x − 1
    f
    y = y + 2
  }
```

$$
\begin{aligned}
E(r_f) = \; & (\Delta_{S(x>0)} \circ (\\
& \quad \rho(x = x - 1)\circ \\
& \quad r_f \circ \\
& \quad \rho(y = y + 2)) \\
&) \cup \Delta_{S(x \leq 0)}
\end{aligned}
$$

What is $E(\emptyset)$?

Simpler Example of Recursion

```
def f =
  if (x > 0) {
    x = x − 1
    f
    y = y + 2
  }
```

$$E(r_f) = \Big(\Delta_{S(x>0)} \circ \big(\rho(x = x - 1) \circ r_f \circ \rho(y = y + 2)\big) \Big) \cup \Delta_{S(x \leq 0)}$$

What is $E(\emptyset)$?
What is $E(E(\emptyset))$?

Simpler Example of Recursion

```
def f =
  if (x > 0) {
    x = x − 1
    f
    y = y + 2
  }
```

$$
\begin{aligned}
E(r_f) = \ & (\Delta_{S(x>0)} \circ (\\
& \quad \rho(x = x - 1) \circ \\
& \quad r_f \circ \\
& \quad \rho(y = y + 2)) \\
&) \cup \Delta_{S(x \le 0)}
\end{aligned}
$$

What is $E(\emptyset)$?
What is $E(E(\emptyset))$?
$E^k(\emptyset)$?

Sequence of Bounded Recursions

Consider the sequence of relations $r_0 = \emptyset$, $r_k = E^k(\emptyset)$.
What is the relationship between r_k and r_{k+1}?

Sequence of Bounded Recursions

Consider the sequence of relations $r_0 = \emptyset$, $r_k = E^k(\emptyset)$.
What is the relationship between r_k and r_{k+1}?
Define

$$s = \bigcup_{k \geq 0} r_k$$

Then

$$E(s) = E(\bigcup_{k \geq 0} r_k) \stackrel{?}{=} \bigcup_{k \geq 0} E(r_k) = \bigcup_{k \geq 0} r_{k+1} = \bigcup_{k \geq 1} r_k = \emptyset \cup \bigcup_{k \geq 1} r_k = s$$

If $E(s) = s$ we say s is a **fixed point (fixpoint)** of function E

Exercise with Fixpoints of Real Functions

1. Find all fixpoints of function $f : \mathbb{R} \to \mathbb{R}$ defined as

$$f(x) = x^2 - x - 3$$

Exercise with Fixpoints of Real Functions

1. Find all fixpoints of function $f : \mathbb{R} \to \mathbb{R}$ defined as

$$f(x) = x^2 - x - 3$$

2. Compute the fixpoint that is smaller than all other fixpoints

Union of Finite Unfoldings is Least Fixpoint

C - a collection (set) of sets (e.g. sets of pairs, i.e. relations)
$E : C \to C$ such that for $r_0 \subseteq r_1 \subseteq r_2 \ldots$
we have

$$E(\bigcup_i r_i) = \bigcup_i E(r_i)$$

Then $s = \bigcup_i E^i(\emptyset)$ is such that

1. $E(s) = s$ (we have shown this)
2. if r is such that $E(r) \subseteq r$ (special case: if $E(r) = r$), then $s \subseteq r$

Prove this theorem.

Least Fixpoint

$$s = \bigcup_i E^i(\emptyset)$$

Suppose $E(r) \subseteq r$.
Showing $s \subseteq r$

$$\bigcup_i E^i(\emptyset) \subseteq r$$

Consequence of s being smallest

```
def f =
  if (x > 0) {
    x = x - 1
    f
    y = y + 2
  }
```

$$E(r_f) = \Big(\Delta_{S(x>0)} \circ \big(\rho(x = x - 1) \circ r_f \circ \rho(y = y + 2)\big) \Big) \cup \Delta_{S(x \le 0)}$$

What does it mean that $E(r) \subseteq r$?

Consequence of s being smallest

```
def f =
  if (x > 0) {
    x = x − 1
    f
    y = y + 2
  }
```

$$E(r_f) = \Big(\Delta_{S(x>0)} \circ \big(\rho(x = x-1) \circ r_f \circ \rho(y = y+2)\big)\Big) \cup \Delta_{S(x \leq 0)}$$

What does it mean that $E(r) \subseteq r$?
Plugging r instead of the recursive call results in something that conforms to r

Justifies modular reasoning for recursive functions

To prove that recursive procedure with body E satisfies specification r, show

- $E(r) \subseteq r$
- then because procedure meaning s is least, $s \subseteq r$

Proving that recursive function meets specification

Prove that if s is the relation denoting the recursive function below, then

$$((x, y), (x', y')) \in s \rightarrow y' \geq y$$

```
def f =
  if (x > 0) {
    x = x - 1
    f
    y = y + 2
  }
```

$$E(r_f) = \Big(\Delta_{S(x>0)} \circ \big(\rho(x = x - 1) \circ r_f \circ \rho(y = y + 2)\big) \Big) \cup \Delta_{S(x \leq 0)}$$

Multiple Procedures

Two mutually recursive procedures $r_1 = E_1(r_1), \quad r_2 = E_2(r_2)$

Extend the approach to work on pairs of relations:

$$(r_1, r_2) = (E_1(r_1), E_2(r_2))$$

Define $\bar{E}(r_1, r_2) = (E_1(r_1), E_2(r_2))$, let $\bar{r} = (r_1, r_2)$

$$\bar{E}(\bar{r}) \sqsubseteq \bar{r}$$

where $(r_1, r_2) \sqsubseteq (r_1', r_2')$ iff $r_1 \subseteq r_1'$ and $r_2 \subseteq r_2'$
Even though pairs of relations are not sets, we can analogously define set-like operations on them. Most theorems still hold.

Generalizing: the entire theory works when we have certain ordering relation

Lattices as a generalization of families of sets