

Lecture 6

Program Paths, Loops, and Recursion

Viktor Kuncak

2013

Loop-Free Programs as Relations $\Delta = \{(\vec{v}, \vec{v}') \mid \vec{v} \in \mathcal{Z}^k\}$

command c	$R(c)$	$\rho(c)$
$(x = t)$	$x' = t \wedge \bigwedge_{v \in V \setminus \{x\}} v' = v$	
$c_1 ; c_2$	$\exists \bar{z}. R(c_1)[\bar{x}' := \bar{z}] \wedge R(c_2)[\bar{x} := \bar{z}]$	$\rho(c_1) \circ \rho(c_2)$
if (*) c_1 else c_2	$R(c_1) \vee R(c_2)$	$\rho(c_1) \cup \rho(c_2)$
assume (F)	$F \wedge \bigwedge_{v \in V} v' = v$	$\Delta_{S(F)}$

$\rho(v_i = t) = \{((v_1, \dots, v_i, \dots, v_n), (v_1, \dots, v'_i, \dots, v_n) \mid v'_i = t)\}$

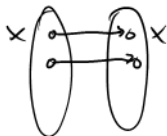
$S(F) = \{\vec{v} \mid F\}$, $\Delta_A = \{(\vec{v}, \vec{v}) \mid \vec{v} \in A\}$ (diagonal relation on A)

Δ (without subscript) is identity on entire set of states (no-op)

We always have: $\rho(c) = \{(\vec{v}, \vec{v}') \mid R(c)\}$

Shorthands:

$$\frac{\mathbf{if}(*)\ c_1\ \mathbf{else}\ c_2}{\mathbf{assume}(F)} \quad \Bigg| \quad \begin{array}{c} c_1 \parallel c_2 \\ [F] \end{array}$$



Examples:

$$\mathbf{if}(F)\ c_1\ \mathbf{else}\ c_2 \equiv [F]; c_1 \parallel [\neg F]; c_2$$

$$\mathbf{if}(F)\ c \equiv [F]; c \parallel [\neg F]$$

Loop-Free Programs

c - a loop-free program whose assignments, havocs, and assumes are c_1, \dots, c_n

The relation $\rho(c)$ is of the form $E(\rho(c_1), \dots, \rho(c_n))$; it composes meanings of c_1, \dots, c_n using union (\cup) and composition (\circ)

<pre>(if (x > 0) x = x - 1 else x = 0); (if (y > 0) y = y - 1 else y = x + 1)</pre>	<pre>([x > 0]; x = x - 1 ∪ [¬(x>0)]; x = 0)); ([y > 0]; y = y - 1 ∪ [¬(y>0)]; y = x+1)</pre>	<pre>($\Delta_{S(x>0)} \circ \rho(x = x - 1)$ ∪ $\Delta_{S(\neg(x>0))} \circ \rho(x = 0)$)◦ ($\Delta_{S(y>0)} \circ \rho(y = y - 1)$ ∪ $\Delta_{S(\neg(y>0))} \circ \rho(y = x + 1)$)</pre>
---	--	---

Note: \circ binds stronger than \cup , so $r \circ s \cup t = (r \circ s) \cup t$

Normal Form for Loop-Free Programs



Composition distributes through union:

$$(r_1 \cup r_2) \circ (s_1 \cup s_2) = r_1 \circ s_1 \cup r_1 \circ s_2 \cup r_2 \circ s_1 \cup r_2 \circ s_2$$

Example corresponding to two if-else statements one after another:

$$\begin{aligned} & \left(\begin{array}{l} \Delta_1 \circ r_1 \\ \cup \\ \Delta_2 \circ r_2 \end{array} \right) \circ \left(\begin{array}{l} \Delta_3 \circ r_3 \\ \cup \\ \Delta_4 \circ r_4 \end{array} \right) \\ & \equiv \begin{array}{l} \Delta_1 \circ r_1 \circ \Delta_3 \circ r_3 \cup \\ \Delta_1 \circ r_1 \circ \Delta_4 \circ r_4 \cup \\ \Delta_2 \circ r_2 \circ \Delta_3 \circ r_3 \cup \\ \Delta_2 \circ r_2 \circ \Delta_4 \circ r_4 \end{array} \end{aligned}$$

Each such composition of basic statements is called basic path.
Loop-free code describes finitely many (exponentially many) paths.

Expressions using \cup and \circ

$$4) E(r) = E_1(r) \circ E_2(r)$$
$$r_1 \subseteq r_2 \rightarrow E_1(r_1) \circ E_2(r_1) \subseteq E_1(r_2) \circ E_2(r_2)$$

For a program with k integer variables, $S = \mathbb{Z}^k$

Consider relations that are subsets of $S \times S$ (i.e. S^2)

The set of all such relations is

$$C = \{r \mid r \subseteq S^2\}$$

Let $E(r)$ be given by an expression built from relation r and some additional relations b_1, \dots, b_n , using \cup and \circ .

Example: $E(r) = (b_1 \circ r) \cup b_2$

$E(r)$ is function $C \rightarrow C$, maps relations to relations

Theorem

E is monotonic function on C :

$$r_1 \subseteq r_2 \rightarrow E(r_1) \subseteq E(r_2)$$

$E(r)$ is r
 $r_1 \subseteq r_2 \rightarrow r_1 \subseteq r_2$

$E(r)$ is b_i
 $r_1 \subseteq r_2 \rightarrow b_i \subseteq b_i$

$$3) E(r) = E_1(r) \cup E_2(r)$$

$$r_1 \subseteq r_2 \rightarrow E_1(r_1) \cup E_2(r_1) \subseteq E_1(r_2) \cup E_2(r_2)$$

Expressions using \cup and \circ

e.g.

$$E(r) = (b_1 \circ r) \cup b_2$$

\cup
 r_i

Prove or disprove.

E distributes over unions, that is, if $r_i, i \in I$ is family of relations,

$$E\left(\bigcup_{i \in I} r_i\right) = \bigcup_{i \in I} E(r_i)$$

$$E(r) = r \circ r$$

$$E(r_1 \cup r_2) = (r_1 \cup r_2) \circ (r_1 \cup r_2) = r_1 \circ r_1 \cup r_1 \circ r_2 \cup r_2 \circ r_1 \cup r_2 \circ r_2$$

$$E(r_1) \cup E(r_2) = r_1 \circ r_1 \cup r_2 \circ r_2$$

for: |
against:

Expressions using \cup and \circ

Does distributivity

$$E\left(\bigcup_{i \in I} r_i\right) = \bigcup_{i \in I} E(r_i)$$

hold, for each of these cases

1. If $E(r)$ is given by an expression containing r at most once?
2. If $E(r)$ contains r any number of times, but I is a finite or countably infinite increasing sequence of relations
 $r_1 \subseteq r_2 \subseteq \dots$
3. If $E(r)$ contains r any number of times, but $r_i, i \in I$ is a **directed family** of relations: for each i, j there exists k such that $r_i \cup r_j \subseteq r_k$.

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 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 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 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 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.

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

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 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 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 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.

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

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 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.

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

Solution:

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

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 $y > 0$.

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

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

This implies $y > 0$.

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

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

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

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

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

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

We would like to have

$$\begin{aligned} L &\equiv \mathbf{if}(F)(c; L) \\ &\equiv \mathbf{if}(F)(c; \mathbf{if}(F)(c; L)) \end{aligned}$$

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

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

Unfolding Loops

$$r_L = \Delta_{nf} \cup (\Delta_f \circ r_c) \circ \Delta_{nf} \cup (\Delta_f \circ r_c)^2 \circ \Delta_{nf} \cup (\Delta_f \circ r_c)^3 \circ r_L$$

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

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

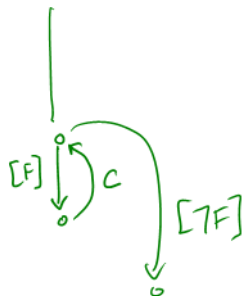
So, $(\Delta_f \circ r_c)^* \circ \Delta_{nf} \subseteq r_L$.

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:

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

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: $r_1 \subseteq r_2 \rightarrow r_1^* \subseteq r_2^*$

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 =	$E(r_f) =$
if (x > 0) {	$\Delta_{S(x>0)} \circ ($
if (x % 2 == 0) {	$(\Delta_{x\%2=0} \circ$
x = x / 2;	$\rho(x = x/2) \circ$
f;	$r_f \circ$
y = y * 2	$\rho(y = y * 2))$
else {	\cup
x = x - 1;	$(\Delta_{x\%2 \neq 0} \circ$
y = y + x;	$\rho(x = x - 1) \circ$
f	$\rho(y = y + x) \circ$
}	$r_f)$
}) $\cup \Delta_{S(x \leq 0)}$
}	

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  
  }
```

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

Simpler Example of Recursion

def f =

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

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

What is $E(\emptyset)$? $E(\emptyset) = \Delta_{S(x \leq 0)}$

Simpler Example of Recursion

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

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

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  
  }
```

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

What is $E(\emptyset)$?

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

$E^k(\emptyset)$?

Sequence of Bounded Recursions

$$r_{k+1} = E(r_k)$$

Consider the sequence of relations $r_0 = \emptyset$, $r_k = E^k(\emptyset)$.

What is the relationship between r_k and r_{k+1} ?

$$\underbrace{\emptyset \subseteq E(\emptyset) \subseteq E(E(\emptyset))}_{r_0 \quad r_1}$$

$$r_k \subseteq r_{k+1} \quad / \cdot E$$

$$E(r_k) \subseteq E(r_{k+1})$$

$$r_{k+1} \subseteq r_{k+2}$$

$$r_0 \subseteq r_1 \subseteq r_2 \subseteq \dots$$

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

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

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

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 $f : \mathbb{R} \rightarrow \mathbb{R}$ defined as

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

$$f(x) = x$$

$$x^2 - x - 3 = x$$

$$x^2 - 2x - 3 = 0$$

$$(x+1)(x-3) = 0$$

$$f(-1) = (-1)^2 - (-1) - 3 = 1 + 1 - 3 = -1$$

Exercise with Fixpoints of Real Functions

1. Find all fixpoints of function $f : \mathbb{R} \rightarrow \mathbb{R}$ 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)

$E : C \rightarrow C$ such that for $r_0 \subseteq r_1 \subseteq r_2 \dots$

we have

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

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 $s \subseteq r$

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

$$\forall i \quad E^i(\emptyset) \subseteq r$$

$$1) \quad i=0 \quad E^0(\emptyset) \subseteq r \\ \emptyset \subseteq r$$

$$2) \quad E^i(\emptyset) \subseteq r / E \\ E(E^i(\emptyset)) \subseteq E(r) \subseteq r$$

Consequence of s being smallest

def $f =$

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

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

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

Consequence of s being smallest

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

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

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

Multiple Procedures

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

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

This leads us to consider LATTICES