Lecture 20
Fixed Point Theorems
Abstract Interpretation Framework
Predicate Abstraction

Viktor Kunčak

Fixpoints

Definition: Given a set A and a function $f : A \to A$ we say that $x \in A$ is a fixed point (fixpoint) of f if $f(x) = x$.

Definition: Let $(A, \leq)$ be a partial order, let $f : A \to A$ be a monotonic function on $(A, \leq)$, and let the set of its fixpoints be $S = \{x \mid f(x) = x\}$. If the least element of S exists, it is called the **least fixpoint**, if the greatest element of S exists, it is called the **greatest fixpoint**.

Fixpoints

Let $(A, \sqsubseteq)$ be a complete lattice and $G : A \rightarrow A$ a monotonic function.

Definition:
Post $= \{x \mid G(x) \sqsubseteq x\}$ - the set of *postfix points* of G
(e.g. $\top$ is a postfix point)
Pre $= \{x \mid x \sqsubseteq G(x)\}$ - the set of *prefix points* of G
Fix $= \{x \mid G(x) = x\}$ - the set of *fixed points* of G.

Note that Fix $\subseteq$ Post.

Tarski's fixed point theorem

Theorem: Let $a = \sqcap \text{Post}$. Then a is the least element of Fix (dually, $\sqcup \text{Pre}$ is the largest element of Fix).

Proof:

Let x range over elements of Post.

- applying monotonic G from $a \sqsubseteq x$ we get $G(a) \sqsubseteq G(x) \sqsubseteq x$
- so $G(a)$ is a lower bound on Post, but a is the greatest lower bound, so $G(a) \sqsubseteq a$
- therefore $a \in \text{Post}$
- Post is closed under G, by monotonicity, so $G(a) \in \text{Post}$
- a is a lower bound on Post, so $a \sqsubseteq G(a)$
- from $a \sqsubseteq G(a)$ and $G(a) \sqsubseteq a$ we have $a = G(a)$, so $a \in \text{Fix}$
- a is a lower bound on Post so it is also a lower bound on a smaller set Fix

In fact, the set of all fixpoints Fix is a lattice itself.

Tarski's fixed point theorem

Tarski's Fixed Point theorem shows that in a complete lattice with a monotonic function G on this lattice, there is at least one fixed point of G, namely the least fixed point $\sqcap$Post.

- Tarski's theorem guarantees fixpoints in complete lattices, but the above proof does not say how to find them.
- How difficult it is to find fixpoints depends on the structure of the lattice.

Let G be a monotonic function on a lattice. Let $a_0 = \bot$ and $a_{n+1} = G(a_n)$. We obtain a sequence $\bot \sqsubseteq G(\bot) \sqsubseteq G^2(\bot) \sqsubseteq \cdots$. Let $a_* = \bigsqcup_{n \geq 0} a_n$.

Lemma: The value a_* is a prefix point.
Observation: a_* need not be a fixpoint (e.g. on lattice [0,1] of real numbers).

Omega continuity

Definition: A function G is ω-continuous if for every chain $x_0 \sqsubseteq x_1 \sqsubseteq \ldots \sqsubseteq x_n \sqsubseteq \ldots$ we have

$$G(\bigsqcup_{i \geq 0} x_i) = \bigsqcup_{i \geq 0} G(x_i)$$

Lemma: For an ω-continuous function G, the value $a_* = \bigsqcup_{n \geq 0} G^n(\bot)$ is the least fixpoint of G.

Iterating sequences and omega continuity

Lemma: For an ω-continuous function G, the value $a_* = \bigsqcup_{n\geq 0} G^n(\bot)$ is the least fixpoint of G.

Proof:

- By definition of ω-continuous we have $G(\bigsqcup_{n\geq 0} G^n(\bot)) = \bigsqcup_{n\geq 0} G^{n+1}(\bot) = \bigsqcup_{n\geq 1} G^n(\bot)$.
- But $\bigsqcup_{n\geq 0} G^n(\bot) = \bigsqcup_{n\geq 1} G^n(\bot) \sqcup \bot = \bigsqcup_{n\geq 1} G^n(\bot)$ because $\bot$ is the least element of the lattice.
- Thus $G(\bigsqcup_{n\geq 0} G^n(\bot)) = \bigsqcup_{n\geq 0} G^n(\bot)$ and a_* is a fixpoint.

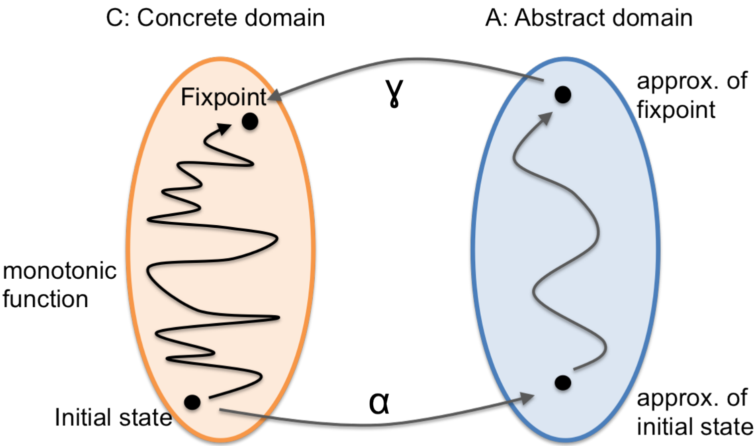
Now let's prove it is the least. Let c be such that $G(c) = c$. We want $\bigsqcup_{n\geq 0} G^n(\bot) \sqsubseteq c$. This is equivalent to $\forall n \in \mathbb{N}. G^n(\bot) \sqsubseteq c$.
We can prove this by induction : $\bot \sqsubseteq c$ and if $G^n(\bot) \sqsubseteq c$, then by monotonicity of G and by definition of c we have $G^{n+1}(\bot) \sqsubseteq G(c) \sqsubseteq c$.

Iterating sequences and omega continuity

Lemma: For an ω-continuous function G, the value $a_* = \bigsqcup_{n \geq 0} G^n(\bot)$ is the least fixpoint of G.

When the function is not ω-continuous, then we obtain a_* as above (we jump over a discontinuity) and then continue iterating. We then take the limit of such sequence, and the limit of limits etc., ultimately we obtain the fixpoint.

Abstract Interpretation Big Picture



Galois Connection

Galois connection (named after Évariste Galois) is defined by two monotonic functions $\alpha : C \to A$ and $\gamma : A \to C$ between partial orders $\leq$ on C and $\sqsubseteq$ on A, such that

$$\forall c, a. \quad \alpha(c) \sqsubseteq a \iff c \leq \gamma(a) \qquad (*)$$

(intuitively the condition means that c is approximated by a).

Lemma: The condition $(*)$ holds iff the conjunction of these two conditions:

$$\begin{array}{rcl} c & \leq & \gamma(\alpha(c)) \\ \alpha(\gamma(a)) & \sqsubseteq & a \end{array}$$

holds for all c and a.

Exercise

A Galois connection is defined by two monotonic functions $\alpha : C \to A$ and $\gamma : A \to C$ between partial orders $\leq$ on C and $\sqsubseteq$ on A, such that

$$\forall a, c. \quad \alpha(c) \sqsubseteq a \iff c \leq \gamma(a) \qquad (*)$$

(intuitively, the condition means that c is approximated by a).

a) Show that the condition $(*)$ is equivalent to the conjunction of these two conditions:

$$\begin{array}{lrcl} \forall c. & c & \leq & \gamma(\alpha(c)) \\ \forall a. & \alpha(\gamma(a)) & \sqsubseteq & a \end{array}$$

b) Let α and γ satisfy the condition of a Galois connection. Show that the following three conditions are equivalent:

   1. $\alpha(\gamma(a)) = a$ for all a
   2. α is a surjective function
   3. γ is an injective function

c) State the condition for $c = \gamma(\alpha(c))$ to hold for all c. When C is the set of sets of concrete states and A is a domain of static analysis, is it more reasonable to expect that $c = \gamma(\alpha(c))$ or $\alpha(\gamma(a)) = a$ to be satisfied, and why?

Abstract Interpretation Recipe: Setup

Given control-flow graph: (V, E, r) where

- $V = \{v_1, \ldots, v_n\}$ is set of program points
- $E \subseteq V \times V$ are control-flow graph edges
- $r : E \to 2^{S \times S}$, so each $r(v, v') \subseteq S \times S$ is relation describing the meaning of command between v and v'

Key steps:

- design abstract domain A that represents sets of program states
- define $\gamma : A \to C$ giving meaning to elements of A
- define lattice ordering $\sqsubseteq$ on A such that $a_1 \sqsubseteq a_2 \to \gamma(a_1) \subseteq \gamma(a_2)$
- define $sp^{\#} : A \times 2^{S \times S} \to A$ that maps an abstract element and a CFG statement to new abstract element, such that $sp(\gamma(a), r) \subseteq \gamma(sp^{\#}(a, r))$
  For example, by defining function α so that (α, γ) becomes a *Galois Connection* and defining $sp^{\#}(a) = \alpha(sp(\gamma(a), r))$.

Running Abstract Interpretation

- Extend $sp^{\#}$ to work on control-flow graphs, by defining $F^{\#} : (V \to A) \to (V \to A)$ as follows (below, $g^{\#} : V \to A$)

$$F^{\#}(g^{\#})(v') = Init(v') \sqcup \bigsqcup_{(v,v') \in E} sp^{\#}(g^{\#}(v), r(v, v'))$$

- Compute $g_*^{\#} = lfp(F^{\#})$ (this is easier than computing semantics because lattice A^n is simpler than C^n):

$$g_*^{\#} = \bigsqcup_{n \geq 0} (F^{\#})^n(\perp^{\#})$$

  where $\perp^{\#}(v) = \perp_A$ for all $v \in V$.

The resulting fixpoint describes an inductive program invariant.

Concrete Domain: Sets of States

Because there is only one variable:

- state is an element of $\mathbb{Z}$ (value of x)
- sets of states are sets of integers, $C = 2^{\mathbb{Z}}$ (concrete domain)
- for each command K, strongest postcondition function $sp(\cdot, K) : C \to C$

Strongest Postondition

Compute *sp* on example statements:

$$sp(P, x := 0) = \{0\}$$

$$sp(P, x := x + 3) = \{x + 3 \mid x \in P\}$$

$$sp(P, assume(x < 10)) = \{x \mid x \in P \wedge x < 10\}$$

$$sp(P, assume(\neg(x < 10)) = \{x \mid x \in P \wedge x \geq 10\}$$

Sets of States at Each Program Point

Collecting semantics computes with sets of states at each program point

$$g : \{v_0, v_1, v_2, v_3\} \to C$$

We sometimes write g_i as a shorthand for $g(v_i)$, for $i \in \{0, 1, 2, 3\}$.
In the initial state the value of variable is arbitrary: $I = \mathbb{Z}$

post Function for the Collecting Semantics
From here we can derive F that maps g to new value of g:

$$\begin{aligned}
&F(g_0, g_1, g_2, g_3) = \\
&\;(\mathbb{Z}, \\
&\;sp(g_0, x := 0) \cup sp(g_2, x := x + 3), \\
&\;sp(g_1, assume(x < 10)), \\
&\;sp(g_1, assume(\neg(x < 10))))
\end{aligned}$$

Sets of States at Each Program Point

The fixpoint condition $F(g) = g$ becomes a system of equations

$$
\begin{aligned}
g_0 &= \mathbb{Z} \\
g_1 &= sp(g_0, x := 0) \cup sp(g_2, x := x + 3) \\
g_2 &= sp(g_1, assume(x < 10)) \\
g_3 &= sp(g_1, assume(\neg(x < 10)))
\end{aligned}
$$

whereas the postfix point (see Tarski's fixpoint theorem) becomes

$$
\begin{aligned}
&\mathbb{Z} \subseteq g_0 \\
&sp(g_0, x := 0) \cup sp(g_2, x := x + 3) \subseteq g_1 \\
&sp(g_1, assume(x < 10)) \subseteq g_2 \\
&sp(g_1, assume(\neg(x < 10))) \subseteq g_3
\end{aligned}
$$

Computing Fixpoint

To find the fixpoint, we compute the sequence $F^n(\emptyset, \emptyset, \emptyset, \emptyset)$ for $n \geq 0$:

$$
\begin{array}{l}
(\emptyset, \emptyset, \emptyset, \emptyset) \\
(\mathbb{Z}, \emptyset, \emptyset, \emptyset) \\
(\mathbb{Z}, \{0\}, \emptyset, \emptyset) \\
(\mathbb{Z}, \{0\}, \{0\}, \emptyset) \\
(\mathbb{Z}, \{0, 3\}, \{0\}, \emptyset) \\
(\mathbb{Z}, \{0, 3\}, \{0, 3\}, \emptyset) \\
(\mathbb{Z}, \{0, 3, 6\}, \{0, 3\}, \emptyset) \\
(\mathbb{Z}, \{0, 3, 6\}, \{0, 3, 6\}, \emptyset) \\
(\mathbb{Z}, \{0, 3, 6, 9\}, \{0, 3, 6, 9\}, \emptyset) \\
(\mathbb{Z}, \{0, 3, 6, 9, 12\}, \{0, 3, 6, 9\}, \emptyset) \\
(\mathbb{Z}, \{0, 3, 6, 9, 12\}, \{0, 3, 6, 9\}, \{12\}) \\
(\mathbb{Z}, \{0, 3, 6, 9, 12\}, \{0, 3, 6, 9\}, \{12\})
\end{array}
$$

Thus, all subsequent values remain the same and $(\mathbb{Z}, \{0, 3, 6, 9, 12\}, \{0, 3, 6, 9\}, \{12\})$ is the fixpoint of collecting semantics equations. In general we may need infinitely many iterations to converge.

Question

Suppose that we have a program that terminates for every possible initial state. Can we always find a finite constant n such that

$$F^n(\emptyset, \ldots, \emptyset) = F^{n+1}(\emptyset, \ldots, \emptyset)$$

i.e. the sequence such as the one above is guaranteed to stabilize?

Example: Assume an arbitrary initial value and consider the loop. Compute a sequence of sets of states at the point after the increment statement in the loop, following the equations for collecting semantics.

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

What always works from omega continuity:

$$lfp(F) = \bigcup_{n \geq 0} F^n(\emptyset, \ldots, \emptyset)$$

where $\bigcup$ on a tuple above means taking union of each component separately, so $(A, B) \cup (A', B') = (A \cup A', B \cup B')$.

Variable Range Analysis for Example Program

The general form of abstract interpretation of the collecting semantics is analogous to collecting semantics, but replaces operations on sets with operations on the lattice:

$$F^{\#} : (V \to A) \to (V \to A)$$

$$F(g^{\#})(v') = g^{\#}_{init}(v') \sqcup \bigsqcup_{(v,v') \in E} sp^{\#}(g^{\#}(v), r(v, v'))$$

Here $g^{\#}_{init}(v')$ will be $\bot$ except at the entry into our control-flow graph, where it approximates the set of initial states at the entry point.

Abstract Analysis Domain

Before we had representation for all possible sets of states:

$$C = 2^{\mathbb{Z}}$$

Here we have representation of only certain states, namely intervals:

$$\begin{aligned} A = \; & \{\bot\} \cup \\ & \{(-\infty, q] \mid q \in \mathbb{Z}\} \cup \\ & \{[p, +\infty) \mid p \in \mathbb{Z}\} \cup \\ & \{[p, q] \mid p \leq q\} \cup \\ & \{\top\} \end{aligned}$$

Abstract Analysis Domain

The meaning of domain elements is given by a monotonic *concretization function* $\gamma : A \to C$:

$$\begin{array}{rcl}
\gamma(\bot) & = & \emptyset \\
\gamma(\{(-\infty, q]) & = & \{x \mid x \leq q\} \\
\gamma(\{[p, +\infty)) & = & \{x \mid p \leq x\} \\
\gamma(\{[p, q]) & = & \{x \mid p \leq x \wedge x \leq q\} \\
\gamma(\top) & = & \mathbb{Z}
\end{array}$$

From monotonicity and $a_1 \sqsubseteq a_1 \sqcup a_2$ it follows

$$\gamma(a_1) \;\subseteq\; \gamma(a_1 \sqcup a_2)$$

and thus

$$\gamma(a_1) \cup \gamma(a_2) \;\subseteq\; \gamma(a_1 \sqcup a_2)$$

We try to define γ to be as small as possible while satisfying this condition.

Abstract Analysis Domain

Define *abstraction function* $\alpha : C \to A$ such that

- $\alpha(s) = [\min s, \max s]$ if those values exist (set is bounded from below and above)
- $\alpha(s) = [\min s, +\infty)$ if there is lower but no upper bound
- $\alpha(s) = (-\infty, \max s]$ if there is upper but no lower bound
- $\alpha(s) = \top$ if there is no upper and no lower bound
- $\alpha(\emptyset) = \bot$

Lemma: The pair (α, γ) form a Galois Connection.

Abstract Analysis Domain

By property of Galois Connection, the condition $\gamma(a_1) \cup \gamma(a_2) \;\subseteq\; \gamma(a_1 \sqcup a_2)$ is equivalent to

$$\alpha(\gamma(a_1) \cup \gamma(a_2)) \;\sqsubseteq\; a_1 \sqcup a_2$$

To make $a_1 \sqcup a_2$ as small as possible, we let the equality hold, defining

$$a_1 \sqcup a_2 \;=\; \alpha(\gamma(a_1) \cup \gamma(a_2))$$

For example,

$$\begin{array}{rcl} [0,2] \sqcup [5,8] & = & \alpha(\gamma([0,2]) \cup \gamma([5,8])) \\ & = & \alpha(\{0,1,2,5,6,7,8\}) \\ & = & [0,8] \end{array}$$

Abstract Postcondition

We had: $sp(\cdot, c) : C \rightarrow C$
Now we have: $sp^{\#}(\cdot, c) : A \rightarrow A$
For correctness, we need that for each $a \in A$ and each command r:

$$sp(\gamma(a), r) \subseteq \gamma(sp^{\#}(a, r))$$

We would like $sp^{\#}$ to be *as small as possible so that this condition holds.*
By property of Galois Connection, the condition $sp(\gamma(a), r) \subseteq \gamma(sp^{\#}(a, r))$ is equivalent to

$$\alpha(sp(\gamma(a), r)) \sqsubseteq sp^{\#}(a, r)$$

Because we want $sp^{\#}$ to be as small as possible (to obtain correct result), we let equality hold:

$$sp^{\#}(a, r) = \alpha(sp(\gamma(a), r))$$

Because we know α, γ, sp, we can compute the value of $sp^{\#}(a, r)$ by simplifying certain expressions involving sets of states.

Example

For $p \leq q$ we have:

$$\begin{aligned} sp^{\#}([p,q], x := x+3) &= \alpha(sp(\gamma([p,q]), x := x+3)) \\ &= \alpha(sp(\{x \mid \ p \leq x \wedge x \leq q\}, x := x+3)) \\ &= \alpha(\{x+3 \mid p \leq x \wedge x \leq q\}) \\ &= \alpha(\{y \mid p+3 \leq y \wedge y \leq q+3\}) \\ &= [p+3, q+3] \end{aligned}$$

For K an integer constant and $a \neq \bot$, we have

$$sp^{\#}(a, x := K) = [K, K]$$

Note that for every command given by relation r, we have

$$\begin{aligned} sp^{\#}(\bot, r) &= \alpha(sp(\gamma(\bot), r)) \\ &= \alpha(sp(\emptyset, r)) \\ &= \alpha(\emptyset) \\ &= \bot \end{aligned}$$

Abstract Semantic Function for the Program

In Collecting Semantics for Example Program we had

$$
\begin{aligned}
&F(g_0, g_1, g_2, g_3) = \\
&\ (\mathbb{Z}, \\
&\ sp(g_0, x := 0) \cup sp(g_2, x := x + 3), \\
&\ sp(g_1, assume(x < 10)), \\
&\ sp(g_1, assume(\neg(x < 10))))
\end{aligned}
$$

Here we have:

$$
\begin{aligned}
&F^{\#}(g_0^{\#}, g_1^{\#}, g_2^{\#}, g_3^{\#}) = \\
&\ (\top, \\
&\ sp^{\#}(g_0^{\#}, x := 0) \sqcup sp^{\#}(g_2^{\#}, x := x + 3), \\
&\ sp^{\#}(g_1^{\#}, assume(x < 10)), \\
&\ sp^{\#}(g_1^{\#}, assume(\neg(x < 10))))
\end{aligned}
$$

Solving Abstract Function

Doing the analysis means computing $(F^{\#})^n(\bot, \bot, \bot, \bot)$ for $n \geq 0$:

$$
\begin{array}{l}
(\bot, \bot, \bot, \bot) \\
(\top, \bot, \bot, \bot) \\
(\top, [0,0], \bot, \bot) \\
(\top, [0,0], [0,0], \bot) \\
(\top, [0,3], [0,3], \bot) \\
(\top, [0,3], [0,3], \bot) \\
(\top, [0,6], [0,3], \bot) \\
(\top, [0,6], [0,6], \bot) \\
(\top, [0,9], [0,9], \bot) \\
(\top, [0,12], [0,9], \bot) \\
(\top, [0,12], [0,9], [10,12]) \\
(\top, [0,12], [0,9], [10,12]) \\
\ldots
\end{array}
$$

Note the approximation (especially in the last step) compared to the collecting semantics we have computed before for our example program.

Exercises

Exercise 1:

Consider an analysis that has two integer variables, for which we track intervals, and one boolean variable, whose value we track exactly.
Give the type of $F^{\#}$ for such program.

Exercise 2:

Consider the program that manipulates two integer variables x, y.
Consider any assignment $x = e$, where e is a linear combination of integer variables, for example,

$$x = 2 * x - 5 * y$$

Consider an interval analysis that maps each variable to its value.
Describe an algorithm that will, given a syntax tree of $x = e$ and intervals for x (denoted $[a_x, b_x]$) and y (denoted $[a_y, b_y]$) find the new interval $[a, b]$ for x after the assignment statement.

Exercise 3

a)

For a program whose state is one integer variable and whose abstraction is an interval, derive general transfer functions $sp^{\#}(a, c)$ for the following statements c, where K is an arbitrary compile-time constant known in the program:

- x= K;
- x= x + K;
- assume(x $\leq$ K)
- assume(x $\geq$ K)

b)

Consider a program with two integer variables, x,y. Consider analysis that stores one interval for each variable.

- Define the domain of lattice elements a that are computed for each program point.
- Give the definition for statement $sp^{\#}(a, y = x + y + K)$

Exercise 3 c)

Draw the control-flow graph for the following program.

Run abstract interpretation that maintains an interval for x at each program point, until you reach a fixpoint.

What are the fixpoint values at program points v_4 and v_5?

```
// v0
x := 0;
// v1
while (x < 10) {
  // v2
  x := x + 3;
}
// v3
if (x >= 0) {
  if (x <= 15) {
    a[x]=7; // made sure index is within range
  } else {
    // v4
    error;
  }
} else {
  // v5
  error;
}
```

Termination and Efficiency of Abstract Interpretation Analysis

Definition: A **chain** of length n is a sequence $s_0, s_1, \ldots, s_n$ such that

$$s_0 \sqsubset s_1 \sqsubset s_2 \sqsubset \ldots \sqsubset s_n$$

where $x \sqsubset y$ means, as usual, $x \sqsubseteq y \wedge x \neq y$

Definition: A partial order has a **finite height** n if it has a chain of length n and every chain is of length at most n.

A finite lattice is of finite height.

Example

The constant propagation lattice $\mathbb{Z} \cup \{\bot, \top\}$ is an infinite lattice of height 2. One example chain of length 2 is

$$\bot \sqsubset 42 \sqsubset \top$$

Here the γ function is given by

- $\gamma(k) = \ldots$ when $k \in \mathbb{Z}$
- $\gamma(\top) = \ldots$
- $\gamma(\bot) = \ldots$

The ordering is given by $a_1 \subseteq a_2$ iff $\gamma(a_1) \subseteq \gamma(a_2)$

Example

If a state of a (one-variable) program is given by an integer, then a concrete lattice element is a set of integers. This lattice has infinite height. There is a chain

$$\{0\} \subset \{0,1\} \subset \{0,1,2\} \subset \ldots \subset \{0,1,2,\ldots,n\}$$

for every n.

Convergence in Lattices of Finite Heignth

Consider a finite-height lattice $(L, \sqsubseteq)$ of height n and function

$$F : L \to L$$

What is the maximum length of sequence $\bot, F(\bot), F^2(\bot), \ldots$?
Give an effectively computable expression for $lfp(F)$.

Computing the Height when Combining Lattices

Let $H(L, \leq)$ denote the height of the lattice $(L, \leq)$.

Product

Given lattices $(L_1, \sqsubseteq_1)$ and $(L_2, \sqsubseteq_2)$, consider product lattice with set $L_1 \times L_2$ and potwise order

$$(x_1, x_2) \sqsubseteq (x_1', x_2')$$

iff ...

What is the height of the product lattice?

Exponent

Given lattice $(L, \sqsubseteq)$ and set V, consider the lattice $(L^V, \sqsubseteq')$ defined by

$$g \sqsubset' h$$

iff $\forall v \in V . g(v) \sqsubseteq h(v)$.

What is the height of the exponent lattice?

Computing the Height when Combining Lattices

Let $H(L, \leq)$ denote the height of the lattice $(L, \leq)$.

Product

Given lattices $(L_1, \sqsubseteq_1)$ and $(L_2, \sqsubseteq_2)$, consider product lattice with set $L_1 \times L_2$ and potwise order

$$(x_1, x_2) \sqsubseteq (x_1', x_2')$$

iff ...

What is the height of the product lattice?

Exponent

Given lattice $(L, \sqsubseteq)$ and set V, consider the lattice $(L^V, \sqsubseteq')$ defined by

$$g \sqsubset' h$$

iff $\forall v \in V . g(v) \sqsubseteq h(v)$.

What is the height of the exponent lattice?

Answer: height of L times the cardinality of V.

Predicate Abstraction

Abstract interpretation domain is determined by a set of formulas (predicates) $\mathcal{P}$ on program variables.
Example: $\mathcal{P} = \{P_0, P_1, P_2, P_3\}$ where

$$\begin{array}{rcl} P_0 & \equiv & \text{false} \\ P_1 & \equiv & 0 < x \\ P_2 & \equiv & 0 < y \\ P_3 & \equiv & x < y \end{array}$$

Analysis tries to construct invariants from these predicates using

- conjunctions, e.g. $P_1 \wedge P_3$
- more generally, conjunctions and disjunctions, e.g. $P_3 \wedge (P_1 \vee P_2)$

Predicate Abstraction

Abstract interpretation domain is determined by a set of formulas (predicates) $\mathcal{P}$ on program variables.
Example: $\mathcal{P} = \{P_0, P_1, P_2, P_3\}$ where

$$
\begin{array}{lcl}
P_0 & \equiv & \text{false} \\
P_1 & \equiv & 0 < x \\
P_2 & \equiv & 0 < y \\
P_3 & \equiv & x < y
\end{array}
$$

Analysis tries to construct invariants from these predicates using

- conjunctions, e.g. $P_1 \wedge P_3$
- more generally, conjunctions and disjunctions, e.g. $P_3 \wedge (P_1 \vee P_2)$

For now: we consider only conjunctions.
We assume $P_0 \equiv$ false, other predicates in $\mathcal{P}$ are arbitrary

- expressed in a logic for which we have a theorem prover

Example of Analysis Result

$$\mathcal{P} = \{false, 0 < x, 0 <= x, 0 < y, x < y, x = 0, y = 1, x < 1000, 1000 \leq x\}$$

```
x = 0;
y = 1;
// 0<y, x<y,x=0,y=1, x<1000
// 0<y, 0≤x, x<y
while (x < 1000) {
  // 0<y, 0≤x, x<y, x<1000
  x = x + 1;
  // 0<y, 0≤x, 0<x
  y = 2*x;
  // 0<y, 0≤x, 0<x, x<y
  y = y + 1;
  // 0<y, 0≤x, 0<x, x<y
  print(y);
}
// 0<y, 0≤x, x<y, 1000 ≤ x
```

Lattice of Conjunctions of Predicates and Concretization

$\mathcal{P} = \{P_0, P_1, \ldots, P_n\}$ - predicates

- formulas whose free variables denote program variables

$A = 2^{\mathcal{P}}$, so for $a \in A$ we have $a \subseteq \mathcal{P}$

Example: $a_0 = \{0 < x, x < y\}$.

$s \models F$ means: formula F is true for variables given by the program state s

$$\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$$

Shorthand: $\bigwedge a$ means $\bigwedge_{P \in a} P$

Example: $\gamma(a_0) =$

Lattice of Conjunctions of Predicates and Concretization

$\mathcal{P} = \{P_0, P_1, \ldots, P_n\}$ - predicates

- formulas whose free variables denote program variables

$A = 2^{\mathcal{P}}$, so for $a \in A$ we have $a \subseteq \mathcal{P}$
Example: $a_0 = \{0 < x, x < y\}$.
$s \models F$ means: formula F is true for variables given by the program state s

$$\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$$

Shorthand: $\bigwedge a$ means $\bigwedge_{P \in a} P$
Example: $\gamma(a_0) = \{s \mid s \models 0 < x \wedge x < y\}$.

Lattice of Conjunctions of Predicates and Concretization

$\mathcal{P} = \{P_0, P_1, \ldots, P_n\}$ - predicates

- formulas whose free variables denote program variables

$A = 2^{\mathcal{P}}$, so for $a \in A$ we have $a \subseteq \mathcal{P}$

Example: $a_0 = \{0 < x, x < y\}$.

$s \models F$ means: formula F is true for variables given by the program state s

$$\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$$

Shorthand: $\bigwedge a$ means $\bigwedge_{P \in a} P$

Example: $\gamma(a_0) = \{s \mid s \models 0 < x \wedge x < y\}$. We often assume states are pairs (x, y). Then $\gamma(a_0) =$

Lattice of Conjunctions of Predicates and Concretization

$\mathcal{P} = \{P_0, P_1, \ldots, P_n\}$ - predicates

- formulas whose free variables denote program variables

$A = 2^{\mathcal{P}}$, so for $a \in A$ we have $a \subseteq \mathcal{P}$

Example: $a_0 = \{0 < x, x < y\}$.

$s \models F$ means: formula F is true for variables given by the program state s

$$\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$$

Shorthand: $\bigwedge a$ means $\bigwedge_{P \in a} P$

Example: $\gamma(a_0) = \{s \mid s \models 0 < x \wedge x < y\}$. We often assume states are pairs (x, y). Then $\gamma(a_0) = \{(x, y) \mid 0 < x \wedge x < y\}$.

Lattice of Conjunctions of Predicates and Concretization

$\mathcal{P} = \{P_0, P_1, \ldots, P_n\}$ - predicates

- formulas whose free variables denote program variables

$A = 2^{\mathcal{P}}$, so for $a \in A$ we have $a \subseteq \mathcal{P}$

Example: $a_0 = \{0 < x, x < y\}$.

$s \models F$ means: formula F is true for variables given by the program state s

$$\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$$

Shorthand: $\bigwedge a$ means $\bigwedge_{P \in a} P$

Example: $\gamma(a_0) = \{s \mid s \models 0 < x \land x < y\}$. We often assume states are pairs (x, y). Then $\gamma(a_0) = \{(x, y) \mid 0 < x \land x < y\}$.

If $a_1 \subseteq a_2$ then $\bigwedge a_2$ implies $\bigwedge a_1$, so $\gamma(a_2) \subseteq \gamma(a_1)$.

Lattice of Conjunctions of Predicates and Concretization

$\mathcal{P} = \{P_0, P_1, \ldots, P_n\}$ - predicates

- formulas whose free variables denote program variables

$A = 2^{\mathcal{P}}$, so for $a \in A$ we have $a \subseteq \mathcal{P}$

Example: $a_0 = \{0 < x, x < y\}$.

$s \models F$ means: formula F is true for variables given by the program state s

$$\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$$

Shorthand: $\bigwedge a$ means $\bigwedge_{P \in a} P$

Example: $\gamma(a_0) = \{s \mid s \models 0 < x \wedge x < y\}$. We often assume states are pairs (x, y). Then $\gamma(a_0) = \{(x, y) \mid 0 < x \wedge x < y\}$.

If $a_1 \subseteq a_2$ then $\bigwedge a_2$ implies $\bigwedge a_1$, so $\gamma(a_2) \subseteq \gamma(a_1)$.

Define:

$$a_1 \sqsubseteq a_2 \quad \Longleftrightarrow \quad a_2 \subseteq a_1$$

Lemma: $a_1 \sqsubseteq a_2 \rightarrow \gamma(a_1) \subseteq \gamma(a_2)$

Lattice of Conjunctions of Predicates and Concretization

$\mathcal{P} = \{P_0, P_1, \ldots, P_n\}$ - predicates

- formulas whose free variables denote program variables

$A = 2^{\mathcal{P}}$, so for $a \in A$ we have $a \subseteq \mathcal{P}$
Example: $a_0 = \{0 < x, x < y\}$.
$s \models F$ means: formula F is true for variables given by the program state s

$$\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$$

Shorthand: $\bigwedge a$ means $\bigwedge_{P \in a} P$
Example: $\gamma(a_0) = \{s \mid s \models 0 < x \wedge x < y\}$. We often assume states are pairs (x, y). Then $\gamma(a_0) = \{(x, y) \mid 0 < x \wedge x < y\}$.
If $a_1 \subseteq a_2$ then $\bigwedge a_2$ implies $\bigwedge a_1$, so $\gamma(a_2) \subseteq \gamma(a_1)$.
Define:

$$a_1 \sqsubseteq a_2 \quad \iff \quad a_2 \subseteq a_1$$

Lemma: $a_1 \sqsubseteq a_2 \rightarrow \gamma(a_1) \subseteq \gamma(a_2)$
Does the converse hold?

Size of the Lattice

$$\{\mathit{false}, 0 < x, x < y\} \sqsubseteq \{0 < x, 0 < y\} \sqsubseteq \{0 < x\} \sqsubseteq \emptyset$$

Draw the Hasse diagram for the lattice $(A, \sqsubseteq)$ i.e. $(2^{\mathcal{P}}, \supseteq)$ for $\mathcal{P} = \{P_0, P_1, P_2\}$ a three-element set.

Size of the Lattice

$$\{false, 0 < x, x < y\} \sqsubseteq \{0 < x, 0 < y\} \sqsubseteq \{0 < x\} \sqsubseteq \emptyset$$

Draw the Hasse diagram for the lattice $(A, \sqsubseteq)$ i.e. $(2^{\mathcal{P}}, \supseteq)$ for $\mathcal{P} = \{P_0, P_1, P_2\}$ a three-element set.

What is the top and what is the bottom element of this lattice?

Size of the Lattice

$$\{\mathit{false}, 0 < x, x < y\} \sqsubseteq \{0 < x, 0 < y\} \sqsubseteq \{0 < x\} \sqsubseteq \emptyset$$

Draw the Hasse diagram for the lattice $(A, \sqsubseteq)$ i.e. $(2^{\mathcal{P}}, \supseteq)$ for $\mathcal{P} = \{P_0, P_1, P_2\}$ a three-element set.

What is the top and what is the bottom element of this lattice?
What is the height of the lattice?

Size of the Lattice

$$\{false, 0 < x, x < y\} \sqsubseteq \{0 < x, 0 < y\} \sqsubseteq \{0 < x\} \sqsubseteq \emptyset$$

Draw the Hasse diagram for the lattice $(A, \sqsubseteq)$ i.e. $(2^{\mathcal{P}}, \supseteq)$ for $\mathcal{P} = \{P_0, P_1, P_2\}$ a three-element set.

What is the top and what is the bottom element of this lattice?
What is the height of the lattice?
What is the height of such lattice when $\mathcal{P} = \{P_0, P_1, \ldots, P_n\}$?

Size of the Lattice

$$\{\mathit{false}, 0 < x, x < y\} \sqsubseteq \{0 < x, 0 < y\} \sqsubseteq \{0 < x\} \sqsubseteq \emptyset$$

Draw the Hasse diagram for the lattice $(A, \sqsubseteq)$ i.e. $(2^{\mathcal{P}}, \supseteq)$ for $\mathcal{P} = \{P_0, P_1, P_2\}$ a three-element set.

What is the top and what is the bottom element of this lattice?
What is the height of the lattice?
What is the height of such lattice when $\mathcal{P} = \{P_0, P_1, \ldots, P_n\}$?
Do $\sqcup$ and $\sqcap$ exist?

Galois Connection

For $\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$ we define

$$\alpha(c) = \{P \in \mathcal{P} \mid \forall s \in c.\ s \models P\}$$

$$\alpha(\{(-1, 1)\}) \quad =$$

Galois Connection

For $\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$ we define

$$\alpha(c) = \{P \in \mathcal{P} \mid \forall s \in c.\ s \models P\}$$

$$\alpha(\{(-1, 1)\}) \quad = \quad \{y > 0\}$$

Galois Connection

For $\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$ we define

$$\alpha(c) = \{P \in \mathcal{P} \mid \forall s \in c.\ s \models P\}$$

$$\begin{array}{rcl} \alpha(\{(-1,1)\}) & = & \{y > 0\} \\ \alpha(\{(1,1),(2,2),(3,6)\}) & = & \end{array}$$

Galois Connection

For $\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$ we define

$$\alpha(c) = \{P \in \mathcal{P} \mid \forall s \in c.\ s \models P\}$$

$$\begin{array}{rcl} \alpha(\{(-1,1)\}) & = & \{y > 0\} \\ \alpha(\{(1,1),(2,2),(3,6)\}) & = & \{x > 0, y > 0\} \end{array}$$

Galois Connection

For $\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$ we define

$$\alpha(c) = \{P \in \mathcal{P} \mid \forall s \in c.\ s \models P\}$$

$$\begin{array}{rcl}
\alpha(\{(-1,1)\}) & = & \{y > 0\} \\
\alpha(\{(1,1),(2,2),(3,6)\}) & = & \{x > 0, y > 0\} \\
\alpha(\{(1,0),(0,1)\}) & = &
\end{array}$$

Galois Connection

For $\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$ we define

$$\alpha(c) = \{P \in \mathcal{P} \mid \forall s \in c.\ s \models P\}$$

$$\begin{aligned}
\alpha(\{(-1,1)\}) &= \{y > 0\} \\
\alpha(\{(1,1),(2,2),(3,6)\}) &= \{x > 0, y > 0\} \\
\alpha(\{(1,0),(0,1)\}) &= \emptyset
\end{aligned}$$

Galois Connection

For $\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$ we define

$$\alpha(c) = \{P \in \mathcal{P} \mid \forall s \in c.\ s \models P\}$$

$$\begin{array}{rcl} \alpha(\{(-1,1)\}) & = & \{y > 0\} \\ \alpha(\{(1,1),(2,2),(3,6)\}) & = & \{x > 0, y > 0\} \\ \alpha(\{(1,0),(0,1)\}) & = & \emptyset \\ \gamma(\emptyset) & = & \end{array}$$

Galois Connection

For $\gamma(a) = \{s \mid s \models \bigwedge_{P \in a} P\}$ we define

$$\alpha(c) = \{P \in \mathcal{P} \mid \forall s \in c.\ s \models P\}$$

$$\begin{array}{rcl}
\alpha(\{(-1,1)\}) & = & \{y > 0\} \\
\alpha(\{(1,1),(2,2),(3,6)\}) & = & \{x > 0, y > 0\} \\
\alpha(\{(1,0),(0,1)\}) & = & \emptyset \\
\gamma(\emptyset) & = & S \text{ (set of all states, empty conjunction)}
\end{array}$$

Is (α, γ) a Galois connection between $(A, \sqsubseteq)$ and $(C, \subseteq)$?

Galois Connection for Predicate Abstraction

We show (α, γ) is a Galois Connection. We need to show that

$$c \subseteq \gamma(a) \iff \alpha(c) \supseteq a$$

Galois Connection for Predicate Abstraction

We show (α, γ) is a Galois Connection. We need to show that

$$c \subseteq \gamma(a) \iff \alpha(c) \supseteq a$$

But both conditions easily reduce to

$$\forall P \in a.\ \forall s \in c.\ s \models P$$

Galois Connection for Predicate Abstraction

We show (α, γ) is a Galois Connection. We need to show that

$$c \subseteq \gamma(a) \iff \alpha(c) \supseteq a$$

But both conditions easily reduce to

$$\forall P \in a.\ \forall s \in c.\ s \models P$$

Shorthand: in logic, if M is a set of assignments to variables (structures) and $\mathcal{A}$ is a set of formulas (e.g. axioms), then $M \models \mathcal{A}$ means

$$\forall m \in M. \forall F \in \mathcal{A}.\ m \models F$$

So, both conditions of Galois connection reduce to $c \models a$

Not a Galois Insertion

Is it the case that $\alpha(\gamma(a)) = a$?

Not a Galois Insertion

Is it the case that $\alpha(\gamma(a)) = a$?
We show this is not the case. This is because γ is not injective.
Indeed, take $a_1 = \{false\}$ and $a_2 = \{false, x > 0\}$. Then

$$\gamma(a_1) = \emptyset = \gamma(a_2)$$

Note $\alpha(\gamma(a_1)) = \alpha(\mathcal{P}) = \alpha(\gamma(a_2))$, but $a_1 \neq a_2$, but it is not the case that $a_1 = a_2$. In this particular case,

$$\alpha(\emptyset) = \mathcal{P}$$

and $a_1 \neq \mathcal{P}$ so

$$\alpha(\gamma(a_1)) \neq a_1$$

Not a Galois Insertion

Is it the case that $\alpha(\gamma(a)) = a$?
We show this is not the case. This is because γ is not injective.
Indeed, take $a_1 = \{\mathit{false}\}$ and $a_2 = \{\mathit{false}, x > 0\}$. Then

$$\gamma(a_1) = \emptyset = \gamma(a_2)$$

Note $\alpha(\gamma(a_1)) = \alpha(\mathcal{P}) = \alpha(\gamma(a_2))$, but $a_1 \neq a_2$, but it is not the case that $a_1 = a_2$. In this particular case,

$$\alpha(\emptyset) = \mathcal{P}$$

and $a_1 \neq \mathcal{P}$ so

$$\alpha(\gamma(a_1)) \neq a_1$$

However, the approach works and is sound, even without the condition $\alpha(\gamma(a)) = a$.

Not a Galois Insertion

Is it the case that $\alpha(\gamma(a)) = a$?
We show this is not the case. This is because γ is not injective.
Indeed, take $a_1 = \{\mathit{false}\}$ and $a_2 = \{\mathit{false}, x > 0\}$. Then

$$\gamma(a_1) = \emptyset = \gamma(a_2)$$

Note $\alpha(\gamma(a_1)) = \alpha(\mathcal{P}) = \alpha(\gamma(a_2))$, but $a_1 \neq a_2$, but it is not the case that $a_1 = a_2$. In this particular case,

$$\alpha(\emptyset) = \mathcal{P}$$

and $a_1 \neq \mathcal{P}$ so

$$\alpha(\gamma(a_1)) \neq a_1$$

However, the approach works and is sound, even without the condition $\alpha(\gamma(a)) = a$.
Can you find an example of non-injectivity in our 4 predicates that does not involve false?

Monotonicity of α

Let $c_1 \subseteq c_2$.
We wish to prove that $\alpha(c_1) \supseteq \alpha(c_2)$.

Monotonicity of α

Let $c_1 \subseteq c_2$.
We wish to prove that $\alpha(c_1) \supseteq \alpha(c_2)$.
Let $P \in \alpha(c_2)$. Then for all $(x, y) \in c_2$ we have $P(x, y)$.

Monotonicity of α

Let $c_1 \subseteq c_2$.
We wish to prove that $\alpha(c_1) \supseteq \alpha(c_2)$.
Let $P \in \alpha(c_2)$. Then for all $(x, y) \in c_2$ we have $P(x, y)$.
Then also for all $(x, y) \in c_1$ we have $P(x, y)$, because $c_1 \subseteq c_2$.

Monotonicity of α

Let $c_1 \subseteq c_2$.
We wish to prove that $\alpha(c_1) \supseteq \alpha(c_2)$.
Let $P \in \alpha(c_2)$. Then for all $(x, y) \in c_2$ we have $P(x, y)$.
Then also for all $(x, y) \in c_1$ we have $P(x, y)$, because $c_1 \subseteq c_2$.
Therefore $P \in \alpha(c_1)$. We showed $c_2 \subseteq c_1$, so $c_1 \sqsubseteq c_2$.

Computing Approximate Strongest Postcondition

$\mathcal{P} = \{\mathit{false}, 0 < x, 0 < y, x < y\}$
Consider computing $sp^{\#}(\{0 < x\}, y := x + 1)$. We can test for each predicate $P' \in \mathcal{P}$ whether

$$x > 0 \wedge (y' = x + 1 \wedge x' = x) \implies P'(x', y')$$

We obtain that the condition holds for $0 < x$, $0 < y$, and for $x < y$, but not for *false*. Thus,

$$sp^{\#}(\{0 < x\}, y := x + 1) = \{0 < x, 0 < y, x < y\}$$

Compute

$$sp^{\#}(\{0 < x\}, y := x - 1) =$$

Computing Approximate Strongest Postcondition

$\mathcal{P} = \{\mathit{false}, 0 < x, 0 < y, x < y\}$
Consider computing $sp^{\#}(\{0 < x\}, y := x + 1)$. We can test for each predicate $P' \in \mathcal{P}$ whether

$$x > 0 \wedge (y' = x + 1 \wedge x' = x) \implies P'(x', y')$$

We obtain that the condition holds for $0 < x$, $0 < y$, and for $x < y$, but not for *false*. Thus,

$$sp^{\#}(\{0 < x\}, y := x + 1) = \{0 < x, 0 < y, x < y\}$$

Compute

$$sp^{\#}(\{0 < x\}, y := x - 1) = \{0 < x\}$$

Computing Approximate Strongest Postcondition

$\mathcal{P} = \{\mathit{false}, 0 < x, 0 < y, x < y\}$
Consider computing $sp^{\#}(\{0 < x\}, y := x + 1)$. We can test for each predicate $P' \in \mathcal{P}$ whether

$$x > 0 \wedge (y' = x + 1 \wedge x' = x) \implies P'(x', y')$$

We obtain that the condition holds for $0 < x$, $0 < y$, and for $x < y$, but not for *false*. Thus,

$$sp^{\#}(\{0 < x\}, y := x + 1) = \{0 < x, 0 < y, x < y\}$$

Compute

$$sp^{\#}(\{0 < x\}, y := x - 1) = \{0 < x\}$$
$$sp^{\#}(\{0 < x, x < y\}, x := x - 1) =$$

Computing Approximate Strongest Postcondition

$\mathcal{P} = \{\mathit{false}, 0 < x, 0 < y, x < y\}$
Consider computing $sp^{\#}(\{0 < x\}, y := x + 1)$. We can test for each predicate $P' \in \mathcal{P}$ whether

$$x > 0 \wedge (y' = x + 1 \wedge x' = x) \implies P'(x', y')$$

We obtain that the condition holds for $0 < x$, $0 < y$, and for $x < y$, but not for *false*. Thus,

$$sp^{\#}(\{0 < x\}, y := x + 1) = \{0 < x, 0 < y, x < y\}$$

Compute

$$sp^{\#}(\{0 < x\}, y := x - 1) = \{0 < x\}$$
$$sp^{\#}(\{0 < x, x < y\}, x := x - 1) = \{0 < y, x < y\}$$

Computing Approximate Strongest Postcondition

$\mathcal{P} = \{\mathit{false}, 0 < x, 0 < y, x < y\}$
Consider computing $sp^{\#}(\{0 < x\}, y := x + 1)$. We can test for each predicate $P' \in \mathcal{P}$ whether

$$x > 0 \wedge (y' = x + 1 \wedge x' = x) \implies P'(x', y')$$

We obtain that the condition holds for $0 < x$, $0 < y$, and for $x < y$, but not for *false*. Thus,

$$sp^{\#}(\{0 < x\}, y := x + 1) = \{0 < x, 0 < y, x < y\}$$

Compute

$$sp^{\#}(\{0 < x\}, y := x - 1) = \{0 < x\}$$
$$sp^{\#}(\{0 < x, x < y\}, x := x - 1) = \{0 < y, x < y\}$$

What is the relation between $\{0 < x, x < y\}$ and $\{0 < x, 0 < y, x < y\}$?

Deriving Rule for Computing sp

Fix some command given by relation r.
Denote $a' = sp^{\#}(a, r)$. We are computing a'. For correctness we need

$$sp(\gamma(a), r) \subseteq \gamma(a')$$

Thanks to Galois connection, this is equivalent to

$$\alpha(sp(\gamma(a), r)) \sqsubseteq a'$$

We wish to find the smallest lattice element a', which is the largest set (this gives the tightest approximation). So we let

$$a' = \alpha(sp(\gamma(a), r))$$

Given that $\gamma(a) = \{s \mid s \models \bigwedge a\}$, and $\alpha(c) = \{P \in \mathcal{P} \mid \forall s \in c.\ s \models P\}$,

$$a' = \{P' \in \mathcal{P} \mid \forall (x', y') \in sp(\gamma(a), r).\ P'(x', y')\}$$

Continuing the Derivation of sp

$$a' = \{P' \in \mathcal{P} \mid \forall (x', y').(x', y') \in sp(\gamma(a), r) \rightarrow P'(x', y')\}$$

Let $R(x, y, x', y')$ denote the meaning of relation r

Continuing the Derivation of sp

$$a' = \{P' \in \mathcal{P} \mid \forall (x', y').(x', y') \in sp(\gamma(a), r) \rightarrow P'(x', y')\}$$

Let $R(x, y, x', y')$ denote the meaning of relation r
Then $(x', y') \in sp(\gamma(a), r)$ means

$$\exists x, y.(x, y) \in \gamma(a) \wedge R(x, y, x', y')$$

which, after expanding γ, gives

$$\exists x, y.(\bigwedge_{P \in a} P(x, y)) \wedge R(x, y, x', y')$$

We then plug this expression back into a' definition. Because the existentials are left of implication, the result is:

$$a' = \{P' \in \mathcal{P} \mid \forall x, y, x'y'.\ (\bigwedge_{P \in a} P(x, y)) \wedge R(x, y, x', y') \rightarrow P'(x', y')\}$$

Example of Analysis Result

$$\mathcal{P} = \{false, 0 < x, 0 <= x, 0 < y, x < y, x = 0, y = 1, x < 1000, 1000 \leq x\}$$

```
x = 0;
y = 1;
// 0<y, x<y,x=0,y=1, x<1000
// 0<y, 0≤x, x<y
while (x < 1000) {
   // 0<y, 0≤x, x<y, x<1000
   x = x + 1;
   // 0<y, 0≤x, 0<x
   y = 2*x;
   // 0<y, 0≤x, 0<x, x<y
   y = y + 1;
   // 0<y, 0≤x, 0<x, x<y
   print(y);
}
// 0<y, 0≤x, x<y, 1000 ≤ x
```

Formulation in terms of Removing Predicates

At program entry: $\top$, which is:

Formulation in terms of Removing Predicates

At program entry: $\top$, which is: $\emptyset$ of predicates

At all other points: $\bot$, which is:

Formulation in terms of Removing Predicates

At program entry: $\top$, which is: $\emptyset$ of predicates
At all other points: $\bot$, which is: the set of all predicates $\mathcal{P}$
Lattice elements grow in CFG $\leadsto$ the set of predicates decrease

Formulation in terms of Removing Predicates

At program entry: $\top$, which is: $\emptyset$ of predicates

At all other points: $\bot$, which is: the set of all predicates $\mathcal{P}$

Lattice elements grow in CFG $\leadsto$ the set of predicates decrease

We remove predicates that do not hold