

CS766: Analysis of concurrent programs 2020

Lecture 2: Symbolic operator: strongest post

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2020-01-20

Computing reachable states

- ▶ Proving safety is computing reachable states.
- ▶ states are infinite \implies enumeration impossible
- ▶ To compute reachable states, we need
 - ▶ **finite representations of transition relation and set of states** and
 - ▶ For example, $x > 0$ represents infinite set $\{1, 2, 3, \dots\}$
 - ▶ **ability to compute transitive closure** of transition relation
- ▶ **Idea:** use logic for the above goals

Topic 2.1

Program statements as formulas

Program statements as formulas (Notation)

- ▶ In logical representation, we add a new variable *err* in V to represent error state. Initially, $err = 0$ and $err = 1$ means error has occurred.
- ▶ V' be the vector of variables obtained by adding prime after each variable in V .
 - ▶ V denote the current value of the variables
 - ▶ V' denote the next value of the variables

Example 2.1

Let $V = [x, y, err]$. Therefore, $V' = [x', y', err']$.

Notation : frame

Definition 2.1

$$\text{For } U \subseteq V, \text{ let } \mathit{frame}(U) \triangleq \bigwedge_{x \in V \setminus U} (x' = x)$$

In case of singleton U , we only write the element as parameter.

Exercise 2.1

Let $V = [x, y, err]$

- ▶ $\mathit{frame}(x) :=$
- ▶ $\mathit{frame}(y) :=$
- ▶ $\mathit{frame}(\emptyset) :=$
- ▶ $\mathit{frame}([x, y]) :=$
- ▶ $\mathit{frame}(V) :=$

Program statements as formulas (contd.)

We define logical formula ρ for the data statements as follows.

- ▶ $\rho(x := \text{exp}) \triangleq x' = \text{exp} \wedge \text{frame}(x)$
- ▶ $\rho(x := \text{havoc}()) \triangleq \text{frame}(x)$
- ▶ $\rho(\text{assume}(F)) \triangleq F \wedge \text{frame}(\emptyset)$
- ▶ $\rho(\text{assert}(F)) \triangleq F \Rightarrow \text{frame}(\emptyset)$

Since control locations in a program are always finite, control statements need not be redefined.

Example 2.2

Let $V = [x, y, \text{err}]$.

- ▶ $\rho(x := y + 1) = (x' = y + 1 \wedge y' = y \wedge \text{err}' = \text{err})$
- ▶ $\rho(x := \text{havoc}()) = (y' = y \wedge \text{err}' = \text{err})$
- ▶ $\rho(\text{assume}(x > 0)) = (x > 0 \wedge x' = x \wedge y' = y \wedge \text{err}' = \text{err})$
- ▶ $\rho(\text{assert}(x > 0)) = (x > 0 \Rightarrow (x' = x \wedge y' = y \wedge \text{err}' = \text{err}))$

Exercise 2.2

Show ρ correctly models the assert statement

Executing as satisfaction

We can use ρ to execute the commands.

Give the values for the current state, get the values for the next state.

Example 2.3

Consider command $\rho(x := y + 1) = (x' = y + 1 \wedge y' = y \wedge err' = err)$

Consider current state: $\{x = 1, y = 1, err = 0\}$

To execute the command, we solve the following constraints

$$(x' = 1 + 1 \wedge y' = 1 \wedge err' = 0)$$

We obtain

$$\{x' = 2 \wedge y' = 1 \wedge err' = 0\}$$

Commentary: In the case, we have a unique solution for the primed variables. However, that may not be necessary. For some commands, we may have multiple solutions or none.

Example: executing as satisfaction

Example 2.4

Consider $\rho(\text{assert}(x > 0)) = (x > 0 \Rightarrow (x' = x \wedge y' = y \wedge \text{err}' = \text{err}))$
and current state $\{x = -1, y = 1, \text{err} = 0\}$.

To execute the command, we solve the following constraints

$$(-1 > 0 \Rightarrow (x' = -1 \wedge y' = 1 \wedge \text{err}' = 0))$$

If we simplify the above formula, we obtain

⊤

Any state can be the next state, let us choose the following.

$$\{x = 12345, y = 100000, \text{err} = 1\}$$

Exercise 2.3

What happens if current state is $\{x = 2, y = 1, \text{err} = 0\}$?

Topic 2.2

Aggregated semantics

Aggregate

Another view of executions

sets of valuations \rightarrow sets of valuations

Notation

- ▶ valuation : $\mathbb{Q}^{|V|}$
- ▶ set of valuations : $\mathfrak{p}(\mathbb{Q}^{|V|})$
- ▶ set of valuations \rightarrow set of valuations : $\mathfrak{p}(\mathbb{Q}^{|V|}) \rightarrow \mathfrak{p}(\mathbb{Q}^{|V|})$

We will only refer to the set of reachable valuations/states at a location, not at the whole program.

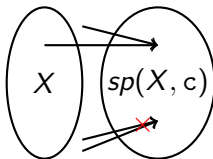
Strongest post: set of valuations to set of valuations

Definition 2.2

Strongest post operator $sp : \mathfrak{p}(\mathbb{Q}^{|V|}) \times \mathcal{P} \rightarrow \mathfrak{p}(\mathbb{Q}^{|V|})$ is defined as follows.

$$sp(X, c) \triangleq \{v' \mid \exists v : v \in X \wedge (v', \text{skip}) \in T^*((v, c))\},$$

where $X \subseteq \mathbb{Q}^{|V|}$ and c is a program.



Example 2.5

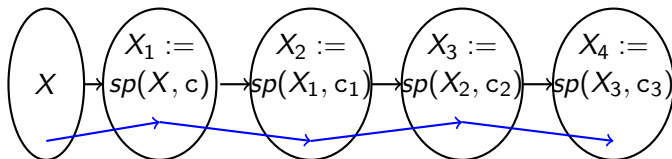
Consider $V = [x]$ and $X = \{[n] \mid n > 0\}$.
 $sp(X, x := x + 1) = \{[n] \mid n > 1\}$

Exercise 2.4

Why use of word
“strongest”?

Reachability and strongest post

No reachable will **escape** the strongest post.



On the other hand, if we do not track all the states in strongest post, we may miss some reachable states.

Symbolic sp

We have discussed that a formula in $\Sigma(V)$ represents a set of valuations.

Hence, we declare symbolic sp that transforms formulas.

$$sp : \Sigma(V) \times \mathcal{P} \rightarrow \Sigma(V)$$

For data statements, the equivalent definition of symbolic sp is

$$sp(F, c) \triangleq (\exists V : F \wedge \rho(c))[V/V'].$$

Example 2.6

Let $V = [x, y, err]$ and $c = x := y + 1$.

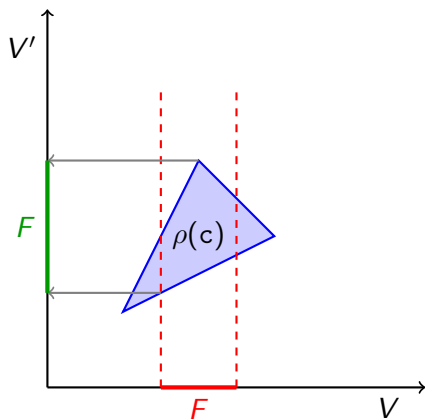
$$\rho(c) = x' = y + 1 \wedge y' = y \wedge err' = err$$

$$sp(y > 2, c) = (\exists x, y, err. (y > 2 \wedge x' = y + 1 \wedge y' = y \wedge err' = err))[V/V']$$

$$= (y' > 2 \wedge x' = y' + 1)[V/V']$$

$$= (y > 2 \wedge x = y + 1)$$

Existence == projection



Exercise : symbolic sp

Exercise 2.5

- ▶ $sp(y > 2 \wedge err = 0, x := havoc()) =$
- ▶ $sp(y > 2 \wedge err = 0, assume(y < 10)) =$
- ▶ $sp(y > 2 \wedge err = 0, assert(y < 0)) =$
- ▶ $sp(\perp, c) =$

Exercise: simplify sp

Exercise 2.6

Show that

- ▶ $sp(F, x := \text{havoc}()) = \exists x.F$
- ▶ $sp(F, \text{assume}(G)) = F \wedge G$
- ▶ $sp(F, \text{assert}(G)) = F \vee \underbrace{\exists V.(F \wedge \neg G)}_{\text{No free variables}}$

Exercise 2.7

Why not simplify $sp(F, x := \text{exp})$ like above?

Symbolic sp for control statements (other than while)

For control statements, the equivalent definitions of symbolic sp are

$$sp(F, c_1; c_2) \triangleq sp(sp(F, c_1), c_2)$$

$$sp(F, c_1 [] c_2) \triangleq sp(F, c_1) \vee sp(F, c_2)$$

$$sp(F, \text{if}(F_1) c_1 \text{ else } c_2) \triangleq sp(F, \text{assume}(F_1); c_1) \vee sp(F, \text{assume}(\neg F_1); c_2)$$

Example 2.7

$$\begin{aligned} sp(x = 0, \text{if}(y > 0) x := x + 1 \text{ else } x := x - 1) &= \\ sp(x = 0, \text{assume}(y > 0); x := x + 1) \vee sp(x = 0, \text{assume}(y \leq 0); x := x - 1) &= \\ = sp(x = 0 \wedge y > 0, x := x + 1) \quad \vee \quad sp(x = 0 \wedge y \leq 0, x := x - 1) &= \\ = (y > 0 \wedge x = 1) \quad \vee \quad (y \leq 0 \wedge x = -1) \end{aligned}$$

Exercise 2.8

1. $sp(x + y > 0, \text{assume}(x > 0); y := y + 1)$
2. $sp(x + y > 0, \text{assume}(x > 0) [] y := y + 1)$

Topic 2.3

Some math: least fixed point

Least fixed point (lfp)

Definition 2.3

For a function f , x is a fixed point of f if $f(x) = x$.

Definition 2.4

For a function f , $\ell = \text{lfp}_x(f(x))$ is the least fixed point of f if

- ▶ $f(\ell) = \ell$ and
- ▶ $\forall y < \ell. f(y) \neq y$.

Definition 2.5

For a function f , $\ell = \text{gfp}_x(f(x))$ is the greatest fixed point of f if

- ▶ $f(\ell) = \ell$ and
- ▶ $\forall y > \ell. f(y) \neq y$.

Example 2.8

Consider function $f(x) = 2/x$. $\sqrt{2}$ and $-\sqrt{2}$ are the fixed points of f .
Therefore,

$$\text{lfp}_x(2/x) = -\sqrt{2} \quad \text{gfp}_x(2/x) = \sqrt{2}$$

Example: fixed-points

Exercise 2.9

Give least fixed point and greatest fixed point of the following functions.

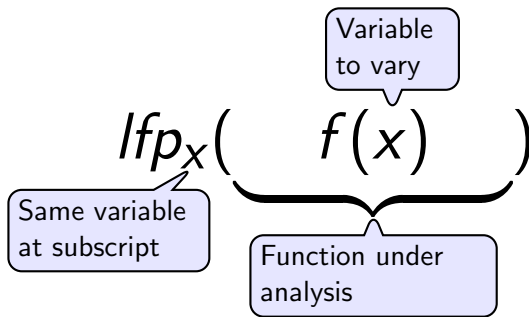
▶ $f(x) = x + 1$

▶ $f(x) = x$

▶ $f(x) = x^2$

▶ $f(x) = x^2 + x - 1$

Notation: least/greatest fixed point



There can be other variables in the function that are assumed to be fixed with respect to the analysis and the answer is parameterized by the free variable.

Example 2.9

Consider

$$lfp_x(x^2 + y) = \frac{-1 - \sqrt{1 - 4y}}{2}$$

Functions for formula

Consider a function like the following

$$f : \Sigma \rightarrow \Sigma$$

Example 2.10

Strongest post $sp(F, c)$ takes two parameters. If we fix c , the function takes a formula as input and returns an output.

- ▶ $sp(x = 0, x := \text{havoc}()) = \top$
- ▶ $sp(y > 2, x := \text{havoc}()) = y > 2$ (fixed point!!)
- ▶ $sp(y + x > 2, x := \text{havoc}()) = \top$

Exercise 2.10

- What is the greatest fixed point for $\text{gfp}_F(sp(F, x := \text{havoc}()))$?*
- What is the least fixed point for $\text{lfp}_F(sp(F, x := \text{havoc}()))$?*

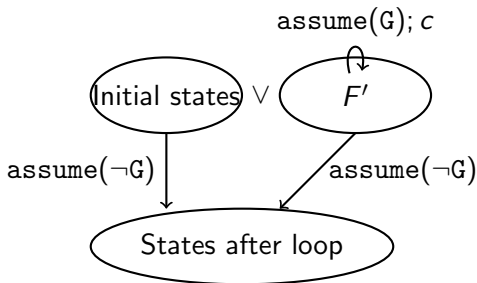
Topic 2.4

sp for loops

Handling while loop

`while(G) c`

F' are set of reachable states at loop head after some number of iterations.



Symbolic for control statements (while)

$$sp(F, \text{while}(G) c) \triangleq sp(\text{lfp}_{F'}(F \vee sp(F' \wedge G, c)), \text{assume}(\neg G))$$

Exercise 2.11

- What is the return type of *lfp* in the above?
- What is the meaning of *sp* in the *lfp*?
- What is the meaning of the whole function in the *lfp*?
- What will happen if we remove ' $F \vee$ ' inside the *lfp*?
- What is the purpose of outside *sp*?

Exercise: symbolic sp for control statements

We have not yet learned an algorithm for sp

Exercise 2.12 (Give intuitive answers!)

1. $sp(x + y > 0, \text{assume}(x > 0); y := y + 1)$
2. $sp(y < 2, \text{while}(y < 10) y := y + 1)$
3. $sp(y > 2, \text{while}(y < 10) y := y + 1)$
4. $sp(y = 0, \text{while}(\top) y := y + 1)$

Safety and symbolic sp

Theorem 2.1

For a program c , if $\not\vdash sp(err = 0, c) \wedge err = 1$ then c is safe.

Exercise 2.13

Prove the above lemma.

We need two key tools from logic to use sp as verification engine.

- ▶ quantifier elimination (for data statements)
- ▶ lfp computation (for loop statement)

There are quantifier elimination algorithms for many logical theories, e.g., integer arithmetic.

However, there is no general algorithm for computing lfp . Otherwise, the halting problem is decidable.

This course is all about developing

incomplete but sound methods for lfp

that work for

some of the programs of our interest.

End of Lecture 2