CS 591: Formal Methods in Security and Privacy

Semantics of programs

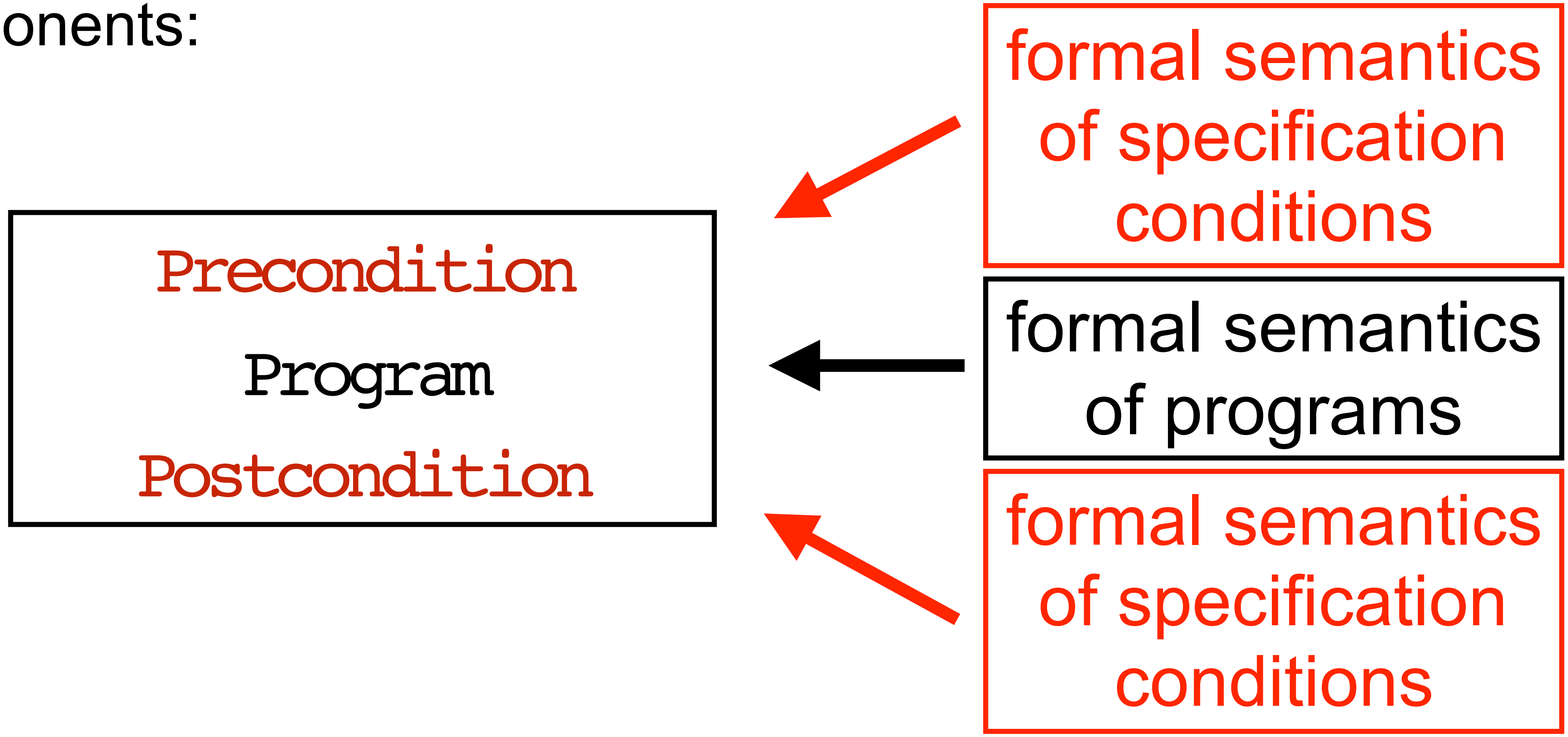
Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

From the previous class

Formal Semantics

We need to assign a formal meaning to the different components:



We also need to describe the rules which combine program and specifications.

A first example

```
FastExponentiation(n, k : Nat) : Nat
  n′:= n; k′:= k; r := 1;
  if k′ > 0 then
    while k′ > 1 do
      if even(k′) then
        n′ := n′ * n′;
        k′ := k′/2;
      else
        r := n′ * r;
        n′ := n′ * n′;
        k′ := (k′ − 1)/2;
    r := n′ * r;
  (* result is r *)
```

Programming Language

```
c::= abort
   | skip
   | x:=e
   | c;c
   | if e then c else c
   | while e do c
```

$x, y, z, \ldots$ program variables

$e_1, e_2, \ldots$ expressions

$c_1, c_2, \ldots$ commands

Expressions

We want to be able to write complex programs with our language.

```
e::= x
   | f(e₁,…,eₙ)
```

Where `f` can be any arbitrary operator.

Some expression examples

`x+5`   `x mod k`   `x[i]`   `(x[i+1] mod 4)+5`

Memories

We can formalize a memory as a map $\mathtt{m}$ from variables to values.

$$\mathtt{m}=[\mathtt{x}_1 \longmapsto \mathtt{v}_1, \ldots, \mathtt{x}_n \longmapsto \mathtt{v}_n]$$

We consider only maps that respect types.

We want to read the value associated to a particular variable:

$$\mathtt{m}(\mathtt{x})$$

We want to update the value associated to a particular variable:

$$\mathtt{m}[\mathtt{x}\leftarrow\mathtt{v}]$$

This is defined as

$$\mathtt{m}[\mathtt{x}\leftarrow\mathtt{v}](\mathtt{y}) = \begin{cases} \mathtt{v} & \text{If } \mathtt{x}=\mathtt{y} \\ \mathtt{m}(\mathtt{y}) & \text{Otherwise} \end{cases}$$

Semantics of Expressions

This is defined on the structure of expressions:

$$\{x\}_m = m(x)$$

$$\{f(e_1,\ldots,e_n)\}_m = \{f\}(\{e_1\}_m,\ldots,\{e_n\}_m)$$

where $\{f\}$ is the semantics associated with the basic operation we are considering.

Semantics of Expressions

Suppose we have a memory

$$\texttt{m=[i}\longmapsto\texttt{1,x}\longmapsto\texttt{[1,2,3],y}\longmapsto\texttt{2]}$$

That {mod} is the modulo operation and {+} is addition, we can derive the meaning of the following expression:

$\{\texttt{(x[i+1] mod y)+5}\}_\texttt{m}$
$= \{\texttt{(x[i+1] mod y)}\}_\texttt{m}\{+\}\{5\}_\texttt{m}$
$= (\{\texttt{x[i+1]}\}_\texttt{m}\ \{\texttt{mod}\}\ \{\texttt{y}\}_\texttt{m})\{+\}\{5\}_\texttt{m}$
$= (\{\texttt{x}\}_\texttt{m}[\{\texttt{i}\}_\texttt{m}\{+\}\{1\}_\texttt{m}]\ \{\texttt{mod}\}\ \{\texttt{y}\}_\texttt{m})\{+\}\{5\}_\texttt{m}$
$= (\{\texttt{x}\}_\texttt{m}[1\{+\}1]\ \{\texttt{mod}\}\ 2)\{+\}5$
$= (\{\texttt{x}\}_\texttt{m}[2]\ \{\texttt{mod}\}\ 2)\{+\}5$
$= (2\ \{\texttt{mod}\}\ 2)\{+\}5 = 0\ \{+\}\ 5 = 5$

Today: more on program semantics

Semantics of Commands

What is the meaning of the following command?

```
k:=2; z:=x mod k; if z=0 then r:=1 else r:=2
```

We can give the semantics as a relation between command, memories and memories or failure.

```
Cmd * Mem * (Mem | ⊥)
```

We will denote this relation as:

{c}$_m$=m′ Or {c}$_m$=⊥

This is commonly typeset as: $$[\![c]\!]_m = m'$$

Semantics of Commands

This is defined on the structure of commands:

$$\{\texttt{abort}\}_m = \bot$$

$$\{\texttt{skip}\}_m = m$$

$$\{\texttt{x:=e}\}_m = m[x \leftarrow \{e\}_m]$$

$\{\texttt{c;c}'\}_m = \{\texttt{c}'\}_{m'}$ If $\{\texttt{c}\}_m = m'$

$\{\texttt{c;c}'\}_m = \bot$ If $\{\texttt{c}\}_m = \bot$

$\{\texttt{if e then } c_t \texttt{ else } c_f\}_m = \{c_t\}_m$ If $\{e\}_m = \texttt{true}$

$\{\texttt{if e then } c_t \texttt{ else } c_f\}_m = \{c_f\}_m$ If $\{e\}_m = \texttt{false}$

Semantics of While

What about while

$$\{\texttt{while e do c}\}_m = \texttt{???}$$

Semantics of While

If $\{\texttt{e}\}_{\texttt{m}}\texttt{=false}$ Then

$$\{\texttt{while e do c}\}_{\texttt{m}} = \texttt{m}$$

What about when $\{\texttt{e}\}_{\texttt{m}}\texttt{=true}$?

Semantics of While

If $\{e\}_m = true$ Then we would like to have:

$$\{while\ e\ do\ c\}_m = \{c; while\ e\ do\ c\}_m$$

Is this well defined?

Approximating While

We could define the following syntactic approximations of a While statement:

$$\texttt{while}^{n} \texttt{ e do c}$$

This can be defined inductively on n as:

$$\texttt{while}^{0} \texttt{ e do c = skip}$$

$$\texttt{while}^{n+1} \texttt{ e do c =}$$
$$\texttt{if e then (c;while}^{n} \texttt{ e do c) else skip}$$

We will write
`if e then c`
for
`if e then c else skip`

Approximating While

We could define the following syntactic approximations of a While statement:

$\texttt{while}^{n}\ \texttt{e do c}$

This can be defined inductively on n as:

$\texttt{while}^{0}\ \texttt{e do c = skip}$

$\texttt{while}^{n+1}\ \texttt{e do c =}$
$\texttt{if e then (c;while}^{n}\ \texttt{e do c)}$

We will write

`if e then c`

for

`if e then c else skip`

Semantics of While

We could go back and try to define the semantics using the approximations:

$$\{\texttt{while e do c}\}_m = \{\texttt{while}^n\ \texttt{e do c}\}_m$$

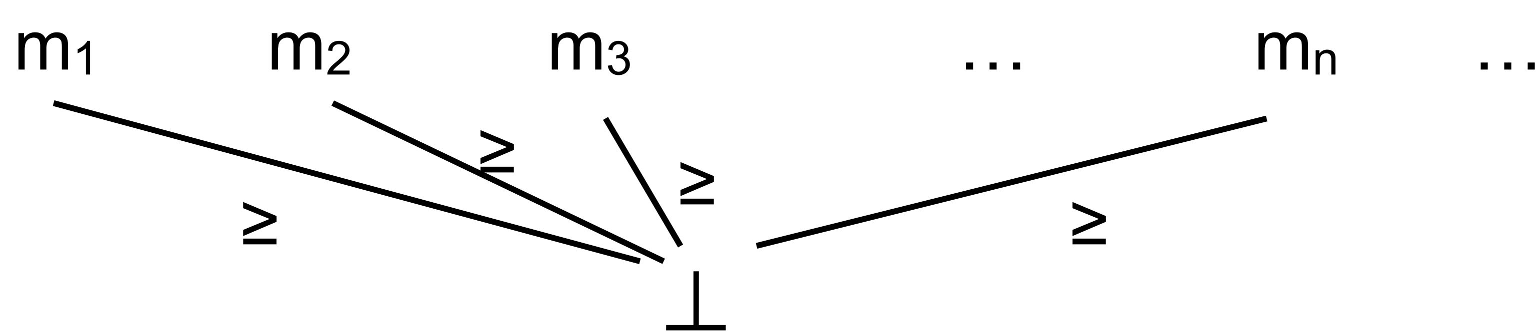
How do we find the n?

Information order

An idea that has been developed to solve this problem is the idea of information order.

This corresponds to the idea of order different possible denotations in term of the information they provide.

In our case we can use the following order on possible outputs:



Semantics of While

Using fixpoint theorems on lattices we can try now to define the semantics using the approximations and a sup operation:

$$\{\texttt{while e do c}\}_m = \sup_{n \in Nat} \{\texttt{while}^n \texttt{ e do c}\}_m$$

Will this work?

We are missing the base case.

Approximating While Revisited

We could define the following lower iteration of a While statement:

$$\texttt{while}_n \ \texttt{e do c}$$

This can be defined using the approximations as:

$$\texttt{while}_n \ \texttt{e do c=(while}^n \ \texttt{e do c);if e then abort}$$

Example

We now have all the components to define the semantics of while:

$$\{\texttt{while e do c}\}_m = \texttt{sup}_{n \in \texttt{Nat}}\{\texttt{while}_n \texttt{ e do c}\}_m$$

Semantics of While

What is the semantics of the following program:

```
n:=3;
r:=1;
while n > 1 do
  r := n * r;
  n := n-1;
```

Semantics of While

What is the semantics of the following program:

```
Fact(n: Nat) : Nat
   r:=1;
   while n > 1 do
        r := n * r;
        n := n-1;
```

Summary of the Semantics of Commands

$\{\texttt{abort}\}_m = \bot$

$\{\texttt{skip}\}_m = m$

$\{x{:=}e\}_m = m[x \leftarrow \{e\}_m]$

$\{c;c'\}_m = \{c'\}_{m'}$ If $\{c\}_m = m'$

$\{c;c'\}_m = \bot$ If $\{c\}_m = \bot$

$\{\texttt{if } e \texttt{ then } c_t \texttt{ else } c_f\}_m = \{c_t\}_m$ If $\{e\}_m = \texttt{true}$

$\{\texttt{if } e \texttt{ then } c_t \texttt{ else } c_f\}_m = \{c_f\}_m$ If $\{e\}_m = \texttt{false}$

$\{\texttt{while } e \texttt{ do } c\}_m = \sup_{n \in Nat}\{\texttt{while}_n \ e \texttt{ do } c\}_m$