Lecture 5

Substitution Example

(2*x > y   /\   x > 0) **[x := u+3]** =

2*(u+3) > y /\ u+3 > 0

(2*x > y   /\   x > 0) **[x := u+3, y:= 7]** =

2*(u+3) > 7   /\ u + 3 > 0

Substitution Defined

$(F_1 \oplus F_2)\ [x := t] \ = \ F_1[x := t] \oplus F_2[x := t]$

$x\ [x:=t] \qquad = t$

$y\ [x:=t] \qquad = y$ (for y not x)

For binders:

$(\exists\ y.\ F)\ [x:=t] = (\exists\ y_1.F')\ \ [x:=t] = \ \exists\ y_1.\ \ F'[x:=t]$

where F' is $F[y:=y_1]$ and y_1 is fresh in F,t

- analogous for $\forall$, λ
- avoids capture, needed when t contains y

Variable Capture

If $\forall x.F$ and t is a term, then $F[x:=t]$

Let F be $\exists y. (x < y)$

t be y

$\forall x. \exists y. (x < y)$ true (about integers)

$\exists y. (y < y)$ false! Not a result of $F[x:=t]$.

Our substitution is *capture avoiding*

Informal Notation for Substitutions

F(x) means:

- F is a formula
- x is some variable (or a vector of variables)
- when we write F(t) later, we will mean F[x:=t] - substitute t instead of x

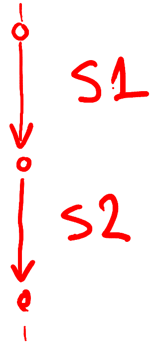
This notation does not say anything about whether x appears in F

- It is about the meaning of future occurrences of F(t)
- One way to think of it: F is function $\lambda z.$ F[x:=z]

Translations of Control Constructs

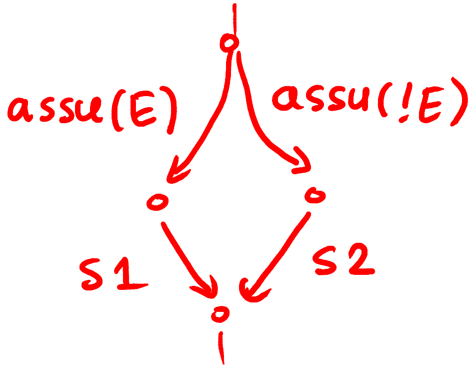
s1 ; s2

s1 ; s2



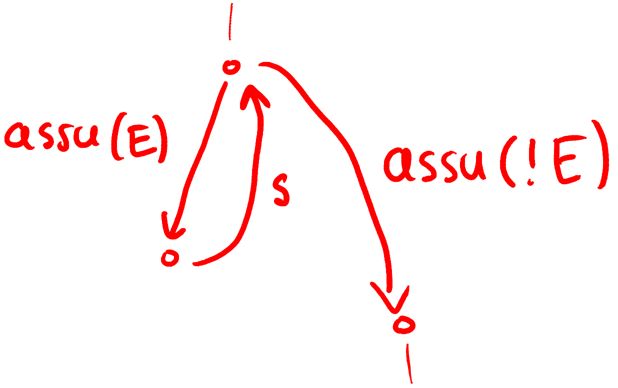
if (E) s1 else s2

(assume (E) ; s1) []
(assume (! E) ; s2)



while (E) s

(assume (E); s)* ;
assume(! E)



Relation Composition

$r_1 = \{(a,b) \mid F_1(a,b)\}$

$r_2 = \{(b,c) \mid F_2(b,c)\}$

$r_1 \circ r_2 = \{(a,c) \mid \exists\, b.\ \ (a,b) \in r_1 \wedge (b,c) \in r_2\} =$

$\{(a,c) \mid \exists\, b.\ F_1(a,b) \wedge F_2(b,c)\} =$

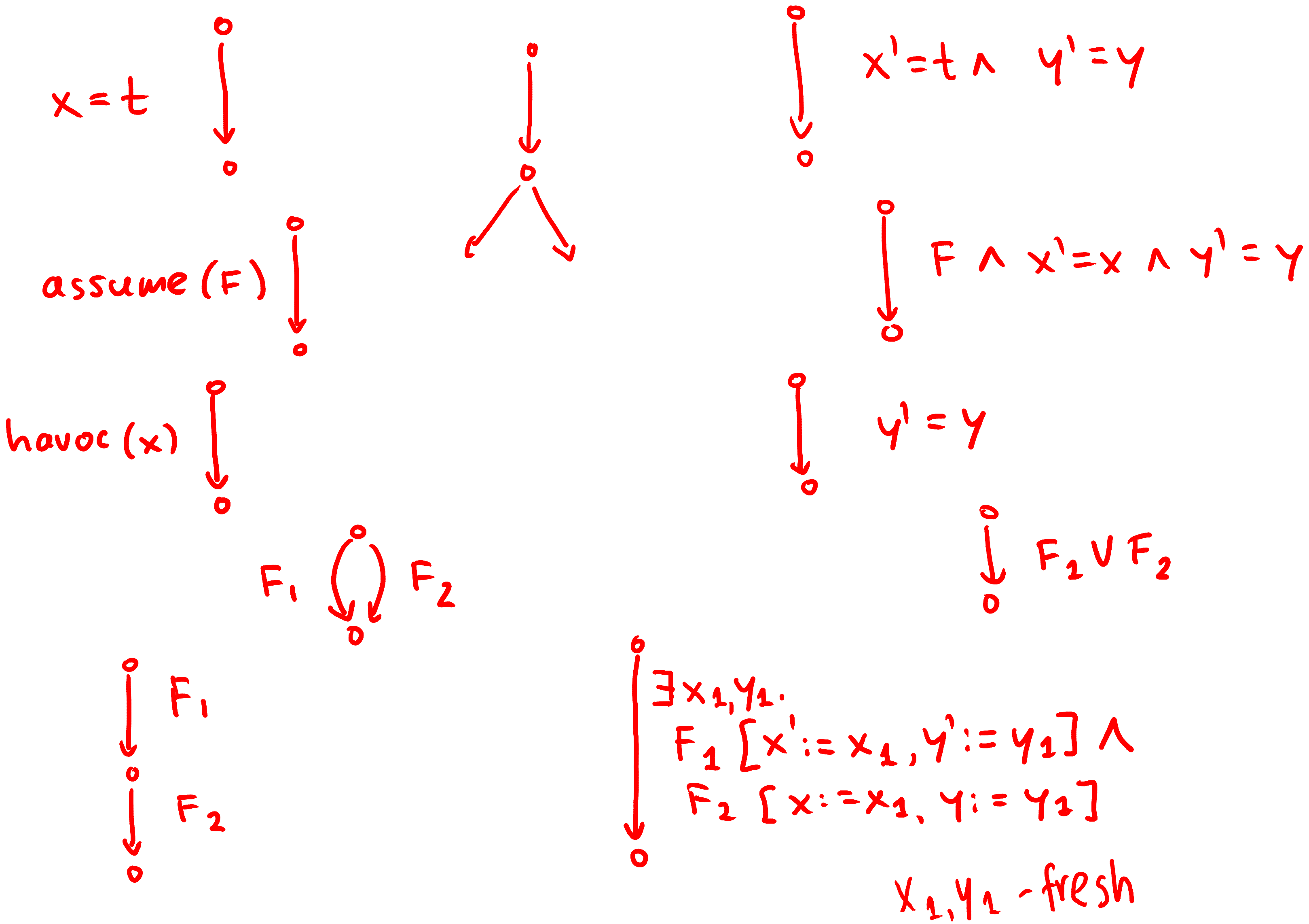
$\{(x,x') \mid \exists\, b.\ F_1(x,b) \wedge F_2(b,x')\}$

Usually formulas are between x and x'

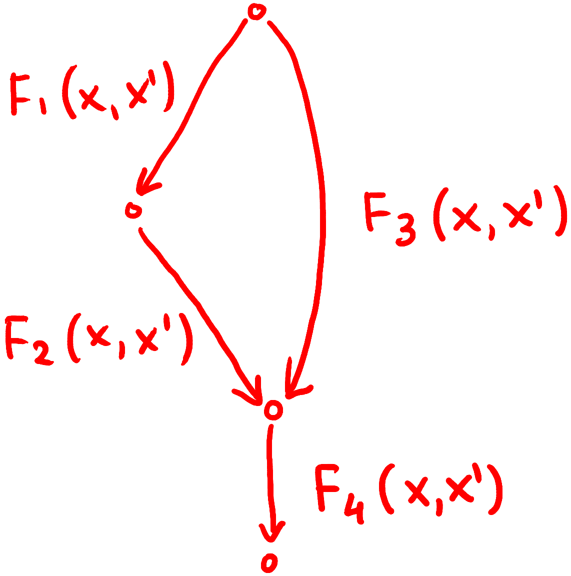
$F_1(x,x')$ is F_1 $\quad$ $F_2(x,x')$ is F_2

$F_1(x,b)$ is $F_1[x':=b]$ $\quad$ $F_2(b,x')$ is $F_2[x:=b]$

Formulas for Loop-Free Code



Translate such CFG into Formula



$$\exists x_1, x_3. \left(\left(\begin{array}{c} F_1(x, x_1) \\ \wedge \\ F_2(x_1, x_3) \end{array} \right) \vee F_3(x, x_3) \right) \wedge F_4(x_3, x')$$

Remember rules, when y does not occur in G

$(\exists y.\ F(y)) \wedge G) \Leftrightarrow \exists y.\ (F(y) \wedge G)$

$(\exists y.\ F(y)) \vee G) \Leftrightarrow \exists y.\ (F(y) \vee G)$

Theorem

Any loop-free CFG labeled with formulas F_1, …, F_n can be translated into a \/,/\ combination of formulas F'_i with some variables renamed by variables that are existentially quantified at the top-level.

We can do this in polynomial time.

Proving that program satisfies spec

Prove:

$$\{pre(x)\}\ (\exists\ y_1, \ldots, y_n.\ F)\ \ \{post(x,x')\}$$

i.e.

$$\forall\ x,x'.\ (pre(x) \wedge (\exists\ y_1, \ldots, y_n.\ F) \rightarrow post(x,x'))$$

$$\forall\ x,x',y_1, \ldots, y_n.\ pre(x) \wedge F \rightarrow post(x,x')$$

i.e. we need to prove that

$$\mathbf{pre(x) \wedge F \wedge \neg post(x,x')}$$

is not satisfiable. No quantifiers in the query.

More on Hoare Triples

Hoare triple transitivity.

Hoare triple distributes over infinite disjunctions of relations.

$\{P\} \cup r_i \{Q\} \quad \Leftrightarrow \quad \forall i.\ \{P\}\ r_i\ \{Q\}$

Use above to derive rule for ***** and for **while**

Rules for computing sp

$sp_F(P(x,y), F(x,y,x',y')) =$

$\exists x_0, y_0. P(x_0,y_0) /\backslash F(x_0, y_0,x,)$

In many cases, this can be simplified.

Rules for computing wp

$$wp_F(F(x,y,x',y'), Q(x,y)) = \forall x', y'.\ (F(x, y,x',y') \rightarrow Q(x',y'))$$

In many cases, this can be simplified.