Synthesis, Analysis, and Verification

Lecture 03b

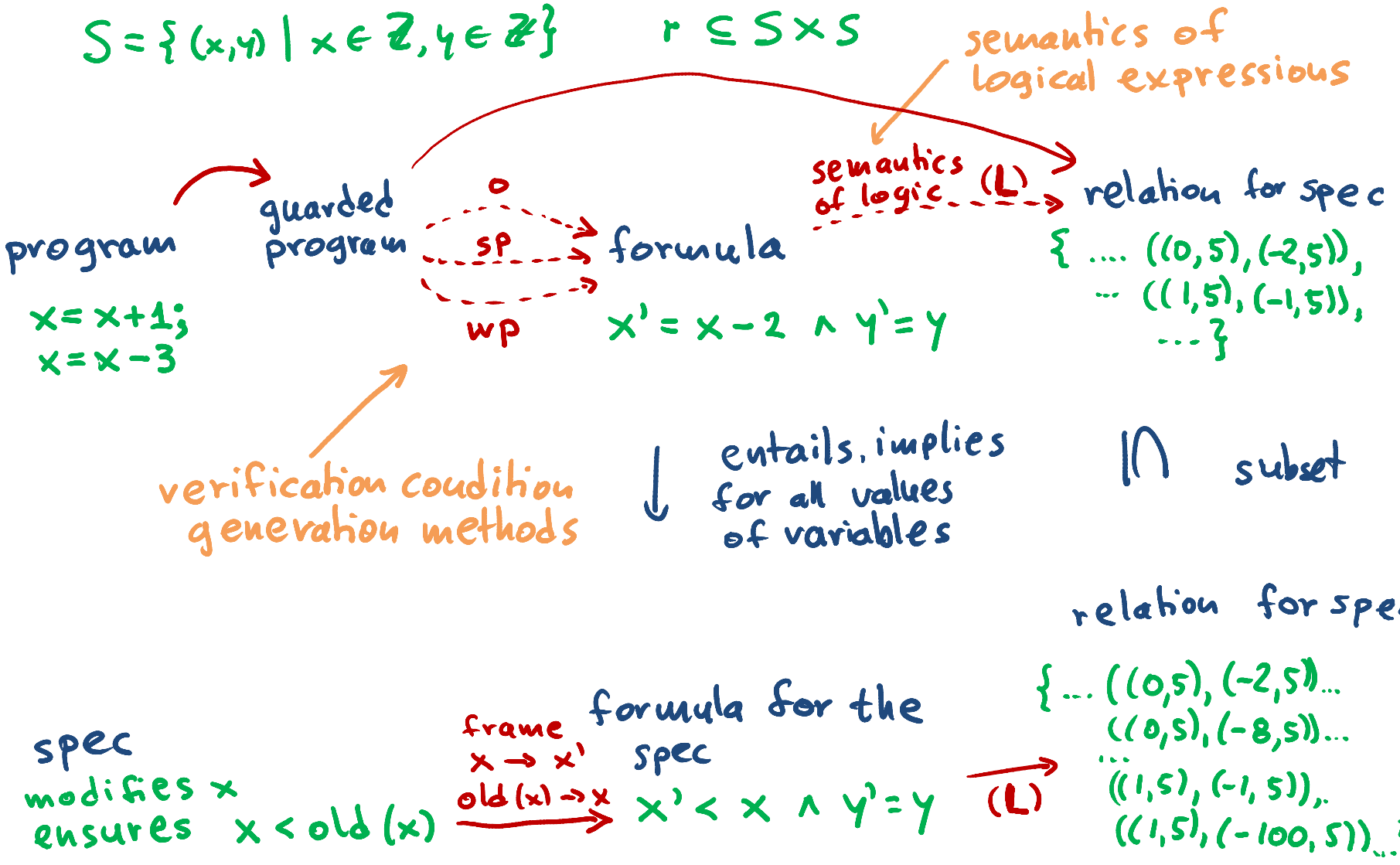
More Hoare Logic. Building Formulas
Substitutions

Lectures:

Viktor Kuncak

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Programs, Relations, Formulas



Forms of Hoare Triple

precondition $\{P\}\ r\ \{Q\}$ postcondition

$\forall s, s'.\quad \underbrace{s \in P}_{q_1} \wedge \underbrace{(s,s') \in r}_{q_2} \rightarrow \underbrace{s' \in Q}_{q_3}$

$(q_1 \wedge q_2 \rightarrow q_3) \leftrightarrow (\neg q_1 \vee (q_2 \rightarrow q_3))$

$\forall s, s'.\quad \neg(s \in P \wedge (s,s') \in r) \vee s' \in Q$

$\forall s'. (\forall s.\ \neg(s \in P \wedge (s,s') \in r)) \vee s' \in Q$

$\forall s'.\ \neg(\exists s.\ s \in P \wedge (s,s') \in r) \vee s' \in Q$

$\forall s'. (\exists s.\ s \in P \wedge (s,s') \in r) \rightarrow s' \in Q$

$\forall s'.\ s' \in sp(P,r) \rightarrow s' \in Q$

$sp(P,r) \subseteq Q$

$\forall s, s'.\quad \underbrace{\neg s \in P}_{q_1} \vee (\underbrace{(s,s') \in r}_{q_2} \rightarrow \underbrace{s' \in Q}_{q_3})$

$\forall s.\ \neg s \in P \vee (\forall s'.\ (s,s') \in r \rightarrow s' \in Q)$

$\forall s.\ s \in P \rightarrow (\forall s'.\ (s,s') \in r \rightarrow s' \in Q)$

$\forall s.\ s \in P \rightarrow s \in wp(r, Q)$

$P \subseteq wp(r, Q)$

Transitivity Rule

$\{P\}\ S_1\ \{Q\} \quad \wedge \quad \{Q\}\ S_2\ \{R\}$

$\forall x,y\ \ x \in P \wedge (x,y) \in S_1 \rightarrow y \in Q \qquad \forall y,z\ \ y \in Q \wedge (y,z) \in S_2 \rightarrow z \in R$

$\forall x,y,z\ \ x \in P \wedge (x,y) \in S_1 \wedge (y,z) \in S_2 \rightarrow z \in R$

$\exists y\ (x,y) \in S_1 \wedge (y,z) \in S_2$

$\forall x,z\ \ x \in P \wedge (x,z) \in S_1 \circ S_2 \rightarrow z \in R$

$\{P\}\ S_1 \circ S_2\ \{R\}$

Expanding Paths

$$\{P\} \bigcup_{i \in J} r_i \{Q\}$$

J can be finite or infinite

$$\forall x,y \quad x \in P \wedge (x,y) \in \bigcup_{i \in J} r_i \rightarrow y \in Q$$

$$\forall x,y \quad x \in P \wedge (\exists i \in J.\ (x,y) \in r_i) \rightarrow y \in Q$$

$$\forall i \in J\ \forall x,y.\quad x \in P \wedge (x,y) \in r_i \hookrightarrow y \in Q$$

$$\forall i \in J \quad \{P\} r_i \{Q\}$$

Transitive Closure

$P = Q$

$$\frac{\{P\}\ r\ \{P\}}{\{P\}\ r^*\ \{P\}}$$

$\{P\}\ r\ \{Q\}$

$\{P\}\ r\ \{Q\} \quad \{P\}\ \Delta\ \{Q\}$

$\{P\}\ r \circ r\ \{Q\}$

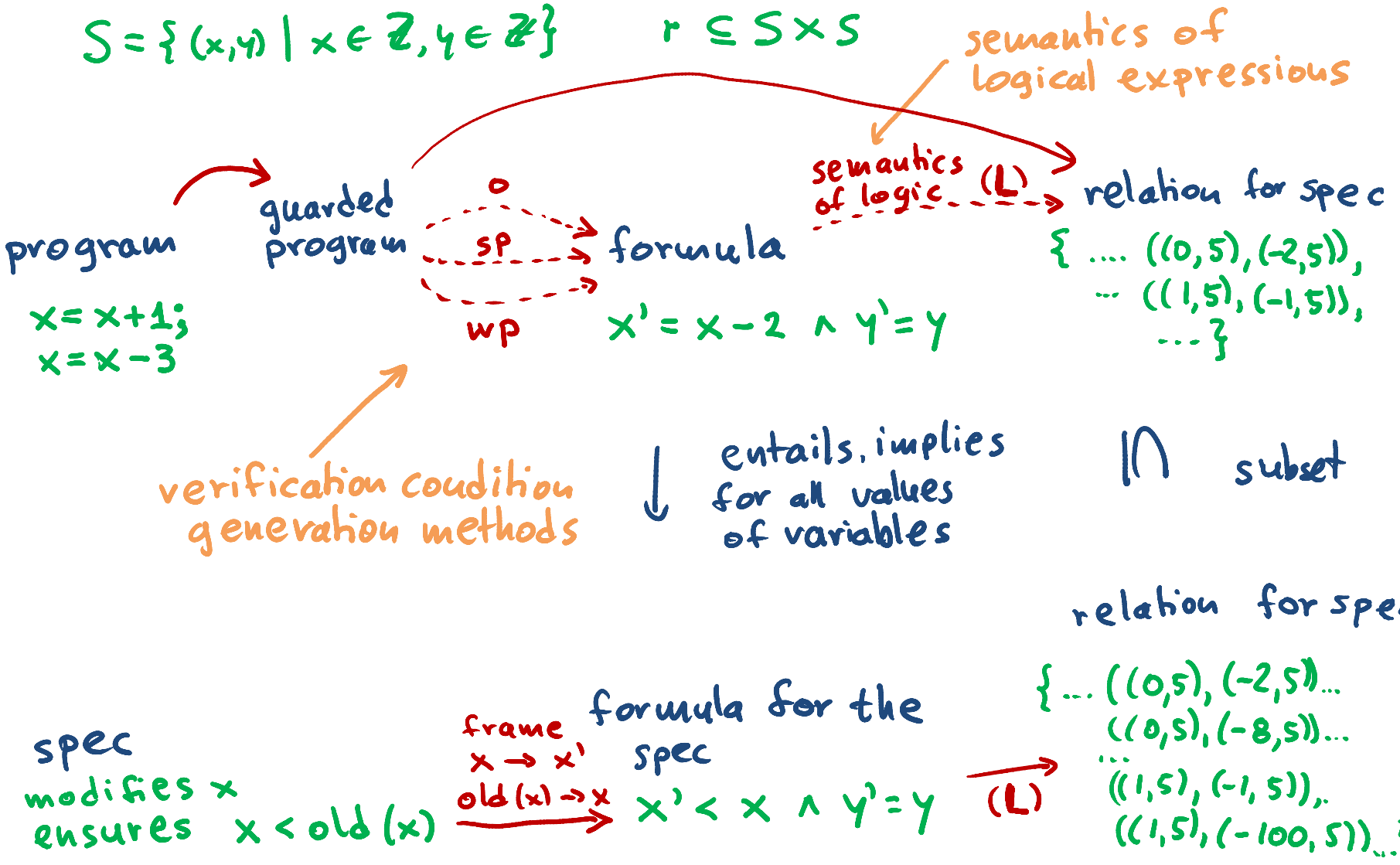
$\forall k \geq 0 \quad \{P\}\ r^k\ \{Q\}$

$\{P\}\ r^*\ \{Q\}$

More on Hoare Logic

- see wiki

Programs, Relations, Formulas



Programs to Formulas (VCG)

Three methods

- compositionally compute formulas for relations
  - then compare them to spec
- forward propagation – compute sp of pre
- backward propagation – compute wp of post

From Programs to Formulas (compositional way)

Given

- guarded program p with set of variables V,

Compute

- formula F
- whose free variables can be x and x' , for all x in V

such that F holds iff

program starting in state given by unprimed variables can end up in state given by primed variables

we should already know the answer

Construct formulas recursively

V – variables

Guarded program given by tree

Leaves: x=E, assume(P)

relation was $\Delta_{[P]} = \{(s,s') \mid P(s) \wedge s' = s\}$ $s \models P$

assume(P) → $P \wedge \bigwedge_{x \in V} x' = x$

x=E → $x' = E \wedge \bigwedge_{y \in V \setminus \{x\}} y' = y$

Tree nodes (recursion)

Non-deterministic choice **[]**

$C_1 \;[]\; C_2 \rightsquigarrow F_1 \vee F_2$

$\mathcal{F}_{\mathcal{C}}(C_1 [] C_2) = \mathcal{F}_{\mathcal{C}}(C_1) \vee \mathcal{F}_{\mathcal{C}}(C_2)$

Sequential composition **;**

$C_1 ; C_2 \rightsquigarrow \exists_{x \in V} x_0 .\ \mathcal{F}_{\mathcal{C}}(C_1)[x' \mapsto x_0]_{x \in V} \wedge \mathcal{F}_{\mathcal{C}}(C_2)[x \mapsto x_0]_{x \in V}$

$x' = x + 1 \qquad x' \leq x + 10$

$y' = y - 2 \qquad y' = y - 100$

$\{((x,y),(x',y')) \mid F_1\} \circ \{((x,y),(x',y')) \mid F_2\} =$

Consequences

$\mathcal{F}_C(\text{assume}(P);\ c) = P \wedge \mathcal{F}_C(c)$

$\exists \vec{x}_0 \in V .\ (P \wedge \bigwedge_{x \in V} x' = x)[x' \mapsto x_0]_{x \in V} \wedge \mathcal{F}_C(c)[x \mapsto x_0]_{x \in V}$

$\exists x_0 \in V .\ P \wedge \bigwedge_{x \in V} x_0 = x \wedge \mathcal{F}_C(c)[x \mapsto x_0]_{x \in V}$

use one-point rule to simplify

$c;\ \text{assume}(P) = \mathcal{F}_C(c) \wedge P[x \mapsto x']_{x \in V}$

$\exists x_0 \in V .\ \mathcal{F}_C(c)[x' \mapsto x_0]_{x \in V} \wedge (P \wedge \bigwedge_{x \in V} x' = x)[x \mapsto x_0]_{x \in V}$

$\exists x_0 \in V .\ \mathcal{F}_C(c)[x' \mapsto x_0]_{x \in V} \wedge P[x \mapsto x_0]_{x \in V} \wedge \bigwedge_{x \in V} x' = x_0$

use one-point rule

About One-Point Rules

Which formula simplifications are correct?

$\exists x.\ x = t \wedge F \quad \rightsquigarrow \quad F[x := t]$

$\exists x.\ x = t \rightarrow F \quad \rightsquigarrow \quad F[x := t]$?

$\forall x.\ x = t \wedge F \quad \rightsquigarrow \quad F[x := t]$?

$\forall x.\ x = t \rightarrow F \quad \rightsquigarrow \quad F[x := t]$

For each either

- find counterexample, or
- prove equivalence (how?)

$\exists$ one-point rule implies the $\forall$ version

$$\forall x.\ x = t \to F \quad \overset{?}{\equiv} \quad F[x := t]$$

$$\neg\, \exists x. \neg\, (x = t \to F)$$

$$\neg\, \underbrace{\exists x.\ x = t \wedge \neg F}, \quad \text{by one-point rule for } \exists$$

$$\neg\ \ (\neg F)[x \mapsto t]$$

$$\neg\ \ \neg\ F[x \mapsto t]$$

$$F[x \mapsto t]$$

Bounded Quantification

$$\forall x \in S.\ F \leftrightarrow \forall x.(x \in S \to F)$$

$$\exists x \in S.\ F \leftrightarrow \exists x.(x \in S \wedge F)$$

Definition of Formulas

$$F ::= F \wedge F \mid F \vee F \mid \neg F \mid \exists y.F \mid \forall y.F \mid A(t_1, \ldots, t_n)$$

$$t ::= c \mid y \mid f(t_1, \ldots, t_m)$$

Definition of Substitution

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

Semantics: Formula → Set of states

$[\![F]\!] = \{ s \mid F \text{ is true is satisfying assignment } s \}$

$[\![F_1 \wedge F_2]\!] = [\![F_1]\!] \cap [\![F_2]\!]$

alternative style:

$s \models F_1 \wedge F_2$ iff $s \models F_1$ and $s \models F_2$

formula semantics

Formula(‘) → Set of Pairs of States

Formulas with primed and unprimed variables

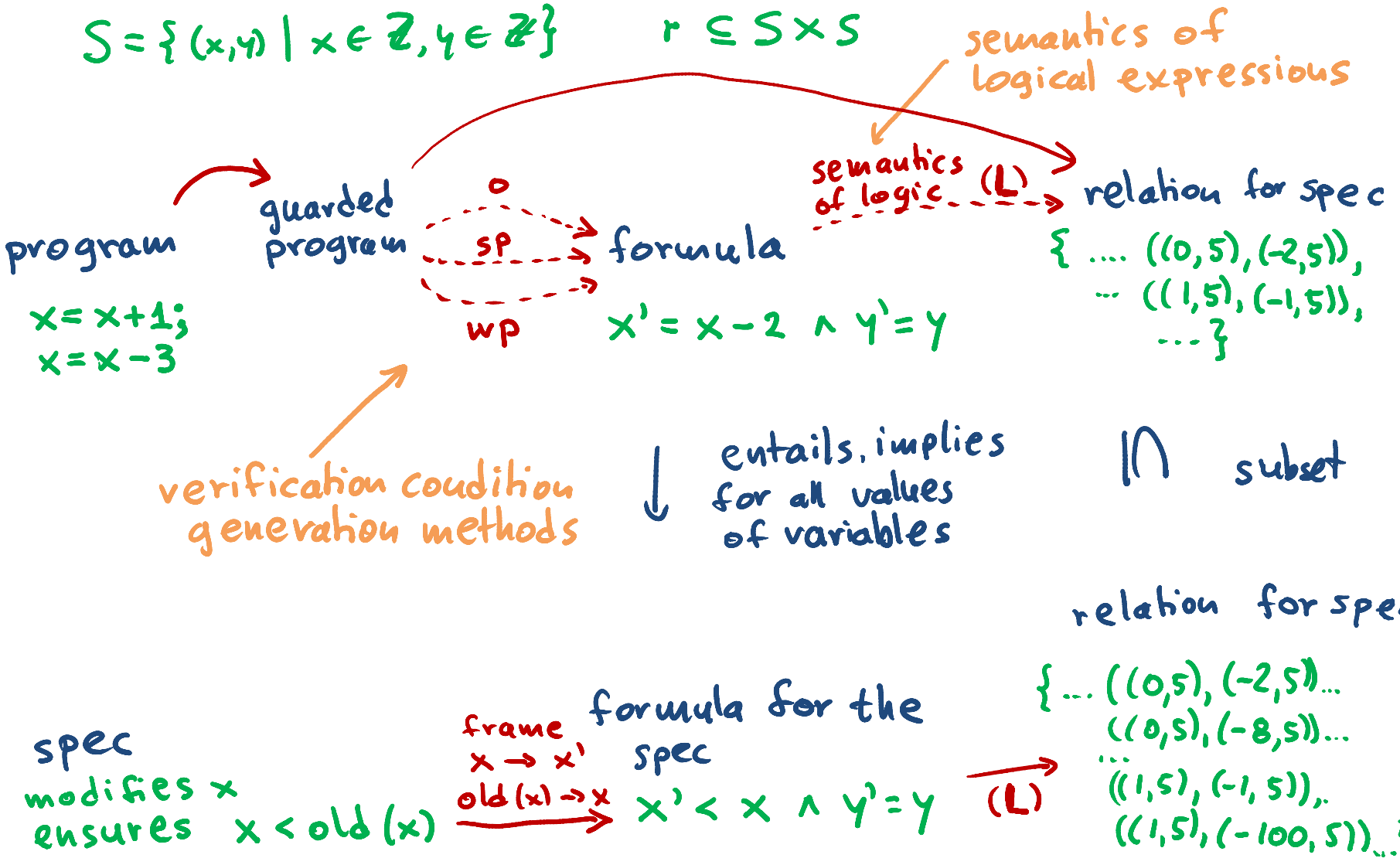
Pairs of Disjoint Functions

Let f_1 , f_2 - partial functions with disjoint domain

Then (f_1 , f_2) can be represented with (f_1 U f_2)

Given semantics for sets of partial functions, we also know how to give semantics for relations on such states

Programs, Relations, Formulas



Lemma for One-Point Rule

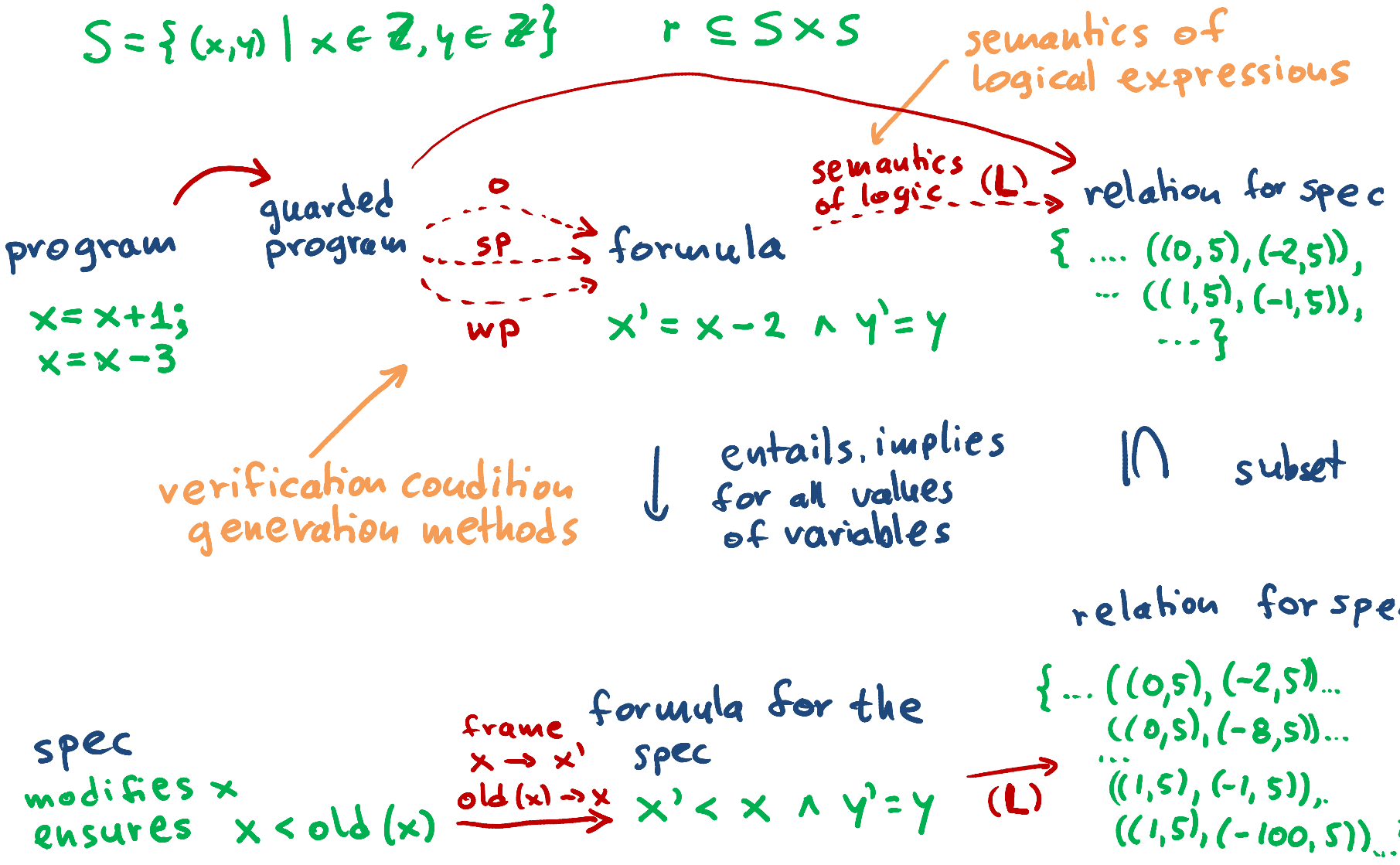
$$[\![F[x:=t]]\!] = \{ s \mid s(x \mapsto [\![t]\!]) \in [\![F]\!] \}$$

where

$$(s(x \mapsto v))(y) = \begin{cases} s(y) & , \text{if } y \neq x \\ v & , \text{if } y = x \end{cases}$$

One Point Rule Proved

Programs to Formulas (VCG)



Further Reading

- C A R Hoare and He Jifeng. Unifying Theories of Programming. Prentice Hall, 1998

- Semantics-based Program Analysis via Symbolic Composition of Transfer Relations, PhD dissertation by Christopher Colby, 1996