$[\![e]\!]_\rho = \sigma$!

$\hat{\rho} : V_{n2} \longrightarrow \hat{V}_{n1}$

$\hat{v} \in \hat{V}_{n1} ::= v \mid \hat{x} \mid \hat{v} \oplus \hat{v} \mid \ominus \hat{v}$ ← symbols

$e \in Exp ::= v \mid x \mid e \oplus e \mid \ominus e$ ← vcbes

↳ prog. variables

$$\frac{}{\langle skip, \hat{\rho} \rangle \rightsquigarrow \langle Cont\langle\cdot\rangle, \hat{\rho} \rangle} \checkmark$$

$$\frac{}{\langle x := e, \hat{\rho} \rangle \rightsquigarrow \langle Cont\langle\cdot\rangle, \hat{\rho}[\hat{x} := [\![e]\!]_\rho] \rangle} \checkmark$$

$$\frac{\langle s_1, \hat{\rho} \rangle \rightsquigarrow \langle Cont\langle\cdot\rangle, \hat{\rho}' \rangle}{\langle s_1 ; s_2, \hat{\rho} \rangle \rightsquigarrow \langle Cont\langle s_2 \rangle, \hat{\rho}' \rangle} \checkmark$$

$$\frac{\langle s_1, \hat{\rho} \rangle \rightsquigarrow \langle Cont\langle s_1' \rangle, \hat{\rho}' \rangle}{\langle s_1 ; s_2, \hat{\rho} \rangle \rightsquigarrow \langle Cont\langle s_1' ; s_2 \rangle, \hat{\rho}' \rangle} \checkmark$$

$$\frac{\langle s_1, \hat{\rho} \rangle \rightsquigarrow \langle \sigma, \hat{\rho}' \rangle \quad \text{if } Absdt(\sigma)}{\langle s_1 ; s_2, \hat{\rho} \rangle \rightsquigarrow \langle \sigma, \hat{\rho}' \rangle} \checkmark$$

✓

$$\frac{[\![e]\!]_\rho = \hat{v} \qquad \Pi \wedge \hat{v} \ \ SAT}{\langle \text{if}(e)\{s_1\}\text{else}\{s_2\}, \hat{\rho}, \Pi \rangle \rightsquigarrow \langle \text{Cont}(s_1), \hat{\rho}, \Pi \wedge \hat{v} \neq 0 \rangle}$$

$\hat{v} \neq 0$ — $\hat{e}$

$$\frac{[\![e]\!]_\rho = \hat{v} \qquad \Pi \wedge \neg \hat{v} \ \ SAT}{\langle \text{if}(e)\{s_1\}\text{else}\{s_2\}, \hat{\rho}, \Pi \rangle \rightsquigarrow \langle \text{Cont}(s_2), \hat{\rho}, \Pi \wedge \hat{v} = 0 \rangle}$$

$\hat{v} = 0$ — $\neg \hat{e}$

$$\langle \text{while}(e)\{s\}, \hat{\rho} \rangle \rightarrow \langle \text{Cont} \langle \text{if}(e)\{s; \text{while}(e)\{s\}\}\text{else}\{\text{skip}\} \rangle, \hat{\rho} \rangle$$

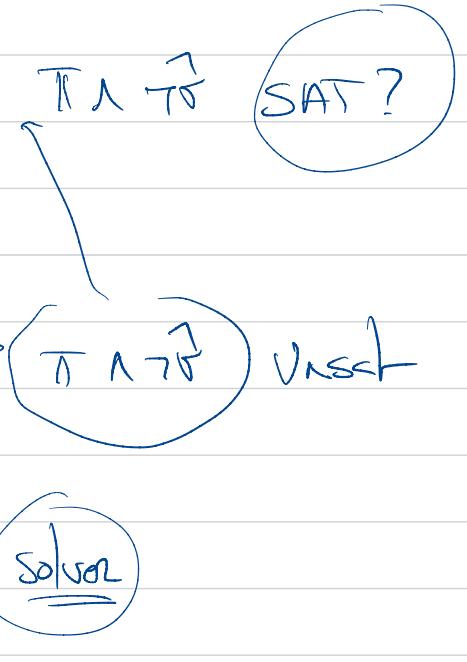
↳ temos $\exists$ fazer um bound à profundidade das execuções

$s ::= \ldots \mid x := \text{symb}() \mid \text{assume}(e) \mid \text{assert}(e)$

$$\frac{\hat{x} \text{ fresh}(\text{prefix})}{x := \text{symb}(), \rho \rightsquigarrow \text{skip}, \rho[x \mapsto \hat{x}]}$$

$$\frac{[\![e]\!]\rho = \hat{r} \qquad \Pi \wedge \hat{v} \text{ SAT}}{\text{assume}(e), \rho, \Pi \rightsquigarrow \text{skip}, \rho, \Pi \wedge \hat{v}}$$

$$\frac{[\![e]\!]\rho = \hat{v} \qquad \Pi \Rightarrow \hat{v} \text{ valid}}{\text{assert}(e), \rho, \Pi \rightsquigarrow \text{skip}, \rho, \Pi}$$

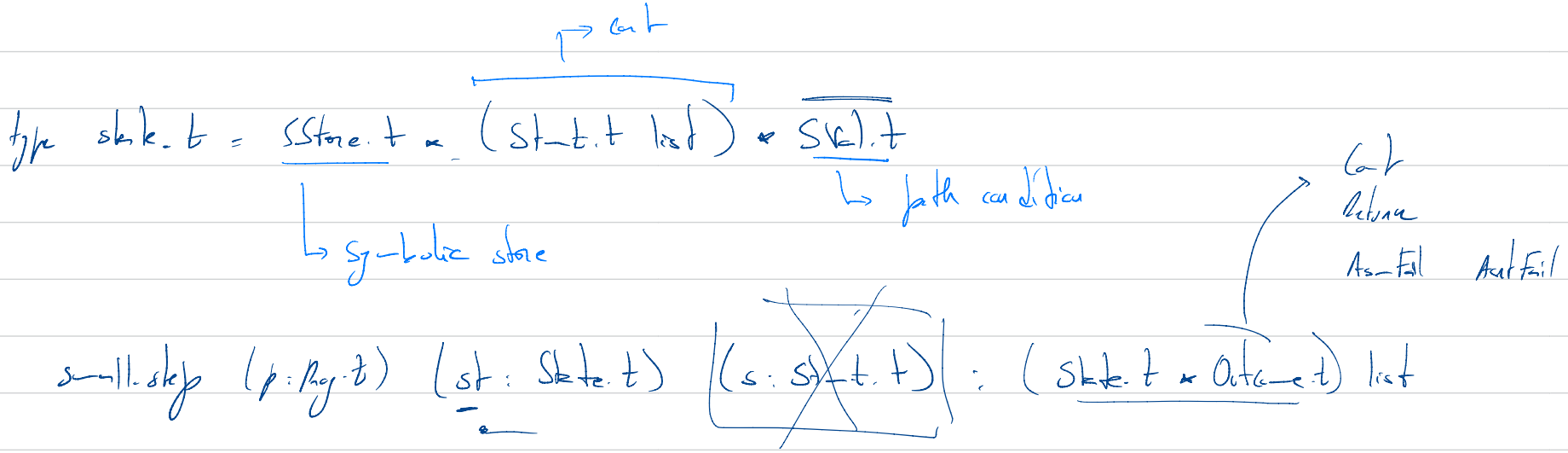


$$s ::= \ldots \mid x := \text{symb}() \mid \text{assume}(e) \mid \text{assert}(e)$$

$$\frac{\hat{x} \text{ fresh}(\text{prefix})}{x := \text{symb}(), \rho \rightsquigarrow \text{skip}, \rho[x \mapsto \hat{x}]}$$

$$\frac{[\![e]\!]\rho = \hat{r} \qquad \Pi \wedge \hat{v} \quad \text{UNSAT}}{\text{assume}(e), \rho, \Pi \rightsquigarrow \text{skip}, \rho, \text{As\_Fail}}$$

$$\frac{[\![e]\!]\rho = \hat{v} \qquad \Pi \wedge \neg \hat{v} \quad \text{SAT}}{\text{assert}(e), \rho, \Pi \rightsquigarrow \text{skip}, \rho, \text{Asrt\_Fail}}$$


cont
type sskt_t = SStore.t * (St_t.t list) * S(c).t
symbolic store
path condition
small_step (p : Prog.t) (st : Sklet.t) : (Sklet.t * Outcome.t) list
Cont
Return
Ass_Fail
Aut Fail
let p, k, π = st in
match s with
| If (e, s1, s2) →
let v = sval_expr p e in
let b1 = is Sat (v ∧ π) in
let b2 = is Sat (¬v ∧ π) in
if b1 && b2
then (
let p' = SStore.copy p in
[((p, s1 ; k, v ∧ π), Cont) ; ((p', s2 ; k, ¬v ∧ π), Cont)]
)

active states

```
type return_t = (Val.t * Out_e.t)

multi_step (p : Prog.t) (sts : SSkte.t list) (acto : return_t list) : return_t list

if sts = [] then acto
else
  (1) let st, sts' = pick sts in
  (2) let next_sts = small_step p st in
  (3) let next_sts_cont, next_sts_final = List.split is_final next_sts in
  (4) multistep
        (join (first (next_sts_cont))
              sts')
        ((finalize next_sts_final) @ acto)
```