CS 520

Theory of Programming Language

03/10 – 03/17, 2021

1. Motivation.

① Predicate logic. ····· boolean expressions in a PL.

+ quantifications.

(a) $\forall x \forall y \exists m \exists n.\ x * m + y * n = 1$

(b) $\forall x \exists y.\ y > x$

$\forall$ $\exists$ "range" over only integers.

binders.

② abstract syntax, den. semantics, inference rules, binders.

③ plays a role in program logic (axiomatic semantics)

2. Abstract syntax. ··· abstract grammar.

① $$\langle \text{intexp} \rangle ::= \underline{0 \mid 1 \mid \ldots} \mid \underline{\langle \text{var} \rangle} \mid -\langle \text{intexp} \rangle \mid \langle \text{intexp} \rangle \begin{smallmatrix} + \\ * \\ (\div \\ \text{rem}, -) \end{smallmatrix} \langle \text{intexp} \rangle$$

$$\langle \text{assert} \rangle ::= \text{true} \mid \text{false} \mid \langle \text{intexp} \rangle \begin{smallmatrix} = \\ > \\ (\geq \\ \leq \\ < \\ \neq) \end{smallmatrix} \langle \text{intexp} \rangle$$

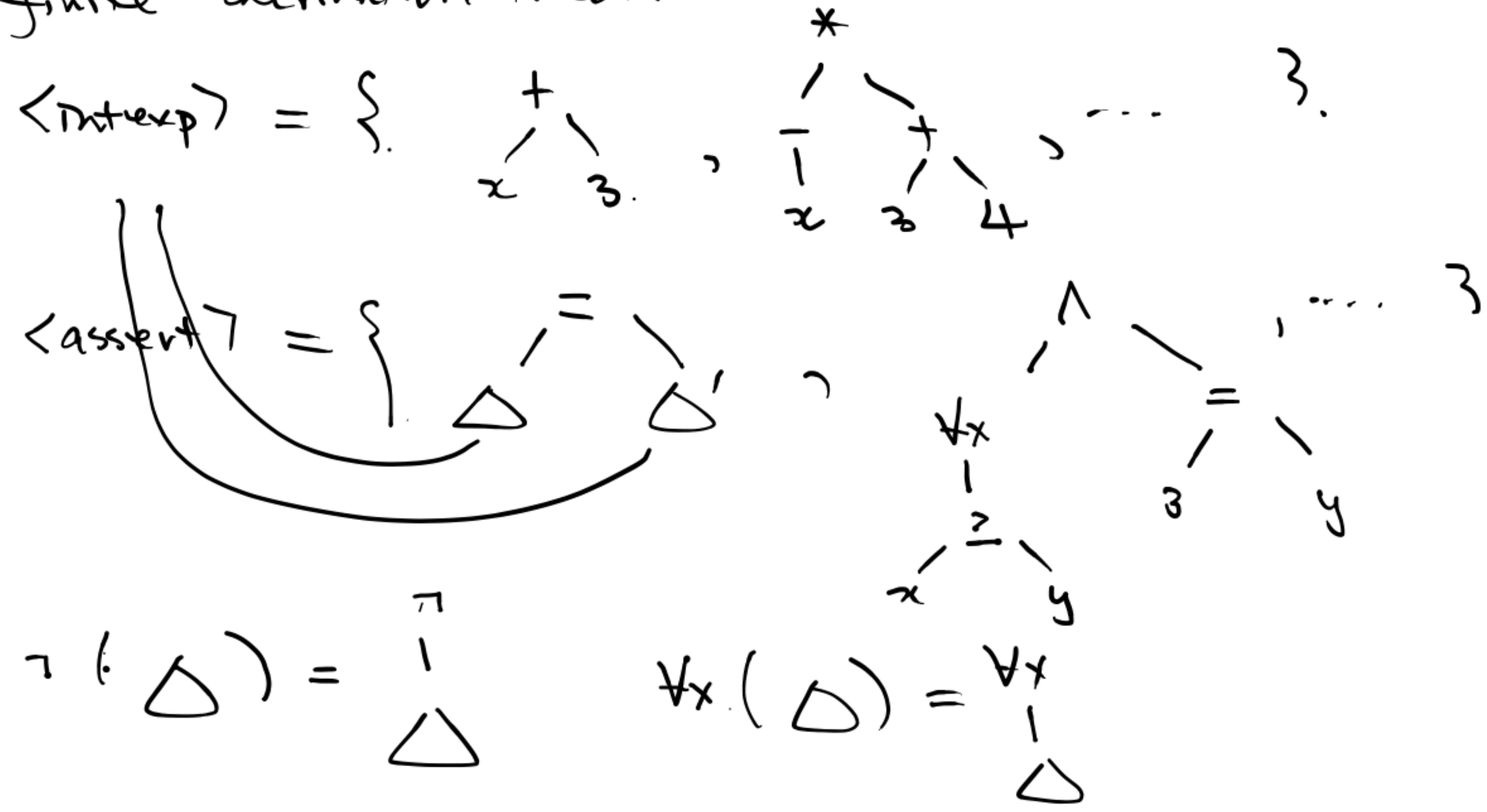
$$\mid \neg \langle \text{assert} \rangle \mid \langle \text{assert} \rangle \begin{smallmatrix} \vee \\ \wedge \\ \vee \end{smallmatrix} \langle \text{assert} \rangle \mid \forall \langle \text{var} \rangle . \langle \text{assert} \rangle \quad (\exists \langle \text{var} \rangle . \langle \text{assert} \rangle).$$

$\forall x. \langle \text{assert} \rangle$, $\forall y. \langle \text{assert} \rangle$, ...

abstract grammar.

② abstract syntax parse trees.

... finite derivation trees.



③ initial algebra.

- signature ···· Java interface.
- algebra ··· Java class.
- initial algebra.

abstract syntax for PL.

abstract grammar for pred. logic

$$(\underset{\text{intexp}}{u},\ \underset{\text{assert}}{t},\ \ldots,\ =\ : u \times u \to t,\ \ldots,\ \forall x : t \to t,\ \ldots)$$

$$A_1 = (S_u, S_t, \ldots, =^0 : S_u \times S_u \to S_t, \ldots, \forall x : S_t \to S_t, \ldots)$$

$$A_2 = (S'_u, S'_t, \ldots, =' : S'_u \times S'_u \to S'_t, \ldots, \forall x : S'_t \to S'_t, \ldots)$$

homomorphism from A_1 to A_2 consists of 2 functions.

$$h : S_u \to S'_u,\quad k : S_t \to S'_t \quad \text{s.t.}$$

h & k together preserve all consts & ops.

1) $S'_t \ni k(\underbrace{\boxed{=^0}(\underset{S_u}{a}, \underset{S_u}{b})}_{\in S_t}) = \underbrace{\boxed{=^1}(\underset{S'_u}{h(a)}, \underset{S'_u}{h(b)})}_{\in S'_t}$ for all $a, b \in S_u$.

④ Consequences.

(1) Syntax-directed def.
(case analysis, induction)

(2) Structural induction. .....

3. Denotational Semantics. .... syntactic entities to mathematical entities.
in a syntax-directed way (compositional way)

omit them often.

$[\![-]\!]_{\underline{intexp}} : \langle intexp \rangle \to [\Sigma \to \mathbb{Z}]$. $\quad (\Sigma = [\langle var \rangle \to \mathbb{Z}])$

$[\![-]\!]_{\underline{assert}} : \langle assert \rangle \to [\Sigma \to \mathbb{B}]$ $\quad (\mathbb{B} = \{tt, ff\})$.

$[\![e]\!]_{intexp}$ ... same as what we did before.

$[\![x]\!]_{intexp} = \lambda\sigma.\ \sigma(x)$.

$[\![x]\!]\sigma = \sigma(x)$

$[\![true]\!]\sigma = tt$

$[\![false]\!]\sigma = ff$. mathematical equality.

syntactic operation.

$[\![e_1 = e_2]\!]\sigma = ([\![e_1]\!]\sigma = [\![e_2]\!]\sigma)$

$[\![e_1 < e_2]\!]\sigma = ([\![e_1]\!]\sigma < [\![e_2]\!]\sigma)$

$[\![\neg p]\!]\sigma = (\neg [\![p]\!]\sigma)$

$[\![p_1 \wedge p_2]\!]\sigma = ([\![p_1]\!]\sigma \wedge [\![p_2]\!]\sigma)$

$[\![\forall x.p]\!]\sigma = (\forall n \in \mathbb{Z}.\ [\![p]\!][\sigma \mid x : n])$. ✓

4. inference rules.

$\forall x \forall y \exists m \exists n \; x*m+y*n=1$

p ... assertion.

p is valid if $[\![p]\!]\sigma = tt$ for all $\sigma \in \Sigma$. (e.g. $\forall x \exists y.\ x>y$.)

premise.

inf. rule.

$$\frac{p_1 \quad p_2 \quad \cdots \quad p_n}{p}$$

conclusion.

if p_1 is valid, p_2 is valid, ..., p_n is valid
then p is valid.

$$\frac{}{e_0 = e_1 \Rightarrow e_1 = e_0}$$

$(\neg(e_0=e_1) \vee (e_1=e_0))$.

$$\frac{p_0 \quad p_0 \Rightarrow p_1}{p_1}$$

ex. ① $\dfrac{p}{\forall x.p}$ ② $\dfrac{}{p \Rightarrow \forall x.p}$ which one is correct? (sound)

V sound

X not sound

if p is valid then $\forall x.p$ is valid.

$p \Rightarrow \forall x.p$ is valid.

$x>1 \Rightarrow \forall x.\ x>1$.

$[\![\forall x.p]\!]\sigma = tt$ for all $\sigma \in \Sigma$.

iff.

$\forall n \in \mathbb{Z}.\ [\![p]\!][\sigma[x:n]] = tt$.

always the case.

5. Binders. .... $\forall x . p$. $\exists x . p$

1) renaming shouldn't make any diff.

2). $(\forall x . (\dots x \dots x \dots)) \,/\, x \to y + z$

should not apply. free.

① Intuition:

ⓐ introduces a new variable.

ⓑ fixes the name x of the variable. y

ⓒ makes the variable available in the scope. "p" $/\, x \to y$.

② free, bound occurrences of a variable, free variable.

$$p = \forall x. \left(x \neq y \lor \forall y \left(x = y \lor \forall x. (x + y \neq x) \right) \right)$$

bound occurrence.   free occurrence.

y is a free variable of p ($\because$ y has a free occurrence in p.)

✓ $FV_{intexp} : \langle intexp \rangle \to 2^{\langle var \rangle}$ .... already did.

✓ $FV_{assert} : \langle assert \rangle \to 2^{\langle var \rangle}$ ... define it in a syntax directed manner.

$FV(true) = FV(false) = \phi$

$FV(e_1 = e_2) = FV(e_1)\ e_2) = FV(e_1) \cup FV(e_2)$ (intexp)

$FV(\neg p) = FV(p)$

$FV(p_1 \land p_2) = FV(p_1) \cup FV(p_2)$

$FV(\forall x. p) = FV(p) \setminus \{x\}$.

(3) substitution ... $\delta \in [\langle var \rangle \to \langle intexp \rangle]$

$x \to x+y, \ y \to z.$

e/δ .... application of δ to expression e.

$x+y \,/\, x \to x+y, y \to z \quad = (x+y)+z.$

Ex: Define p/δ for assertions p in a syntax-directed way.

(a) $(\forall x.\ x>y)/_{x\to y} = \forall x.\ y>y$

wrong def.

(b) $(\forall x.\ x>y)/_{y\to x+1} = \forall x.\ x>x+1$ . . . variable capturing.

different.

update.

$$(\forall x.p)/\delta = \forall x_{new}.\ p/[\delta \mid x : x_{new}]$$

where $x_{new} \notin \bigcup_{y\in FV(p)-\{x\}} FV(\delta(y))$]