1. はじめに

構文論と意味論

第1章

While言語：構文領域

- 数字 n ∈ Num
  n ::= **0** | **1** | n **0** | n **1**　　※二進表現
- 変数 x ∈ Var
- 算術式 a ∈ Aexp
  a ::= n | x | a_1 **+** a_2 | a_1 ***** a_2 | a_1 **−** a_2
- 論理式 b ∈ Bexp
  b ::= **true** | **false** | a_1 **=** a_2 | a_1 **≤** a_2 | **¬**b | b_1 **∧** b_2
- 文 S ∈ Stm
  S ::= x **:=** a | **skip** | S_1**;** S_2
  | **if** b **then** S_1 **else** S_2 | **while** b **do** S

理解確認

While言語：構文領域

○○変数　○○領域

- 数字 n ∈ Num
  n ::= **0** | **1** | n **0** | n **1**　　※二進表現
- 変数 x ∈ Var　　基底要素 or 複合要素
- 算術式 a ∈ Aexp
  a ::= n | x | a_1 **+** a_2 | a_1 ***** a_2 | a_1 **−** a_2
- 論理式 b ∈ Bexp
  b ::= **true** | **false** | a_1 **=** a_2 | a_1 **≤** a_2 | **¬**b | b_1 **∧** b_2
- 文 S ∈ Stm
  S ::= x **:=** a | **skip** | S_1**;** S_2
  | **if** b **then** S_1 **else** S_2 | **while** b **do** S

While言語：構文領域

メタ変数　構文領域

- 数字 n ∈ Num
  n ::= **0** | **1** | n **0** | n **1**　　※二進表現
  （複合要素）
- 変数 x ∈ Var
  （基底要素）
- 算術式 a ∈ Aexp
  a ::= n | x | a_1 **+** a_2 | a_1 ***** a_2 | a_1 **−** a_2
- 論理式 b ∈ Bexp
  b ::= **true** | **false** | a_1 **=** a_2 | a_1 **≤** a_2 | **¬**b | b_1 **∧** b_2
- 文 S ∈ Stm
  S ::= x **:=** a | **skip** | S_1 **;** S_2
  | **if** b **then** S_1 **else** S_2 | **while** b **do** S

プログラム例

- 階乗
  - もし $x = n > 0$ なら，プログラムの停止後に $y = n!$
    y **:= 1; while ¬**(x **= 1**) **do** (y **:=** y ***** x**;** x **:=** x **− 1**)

結合的定義

- 構文の種類は抽象構文によって定まる
- 抽象構文は
  - 基底要素
  - 複合要素：部分に分解される要素

$$a ::= n \mid x \mid a_1 + a_2 \mid a_1 * a_2 \mid a_1 - a_2$$

抽象構文 vs 具象構文

- 抽象構文は式と文の構造に焦点をあてる
  - 字句解析と構文解析は無視できる

- 具象構文は字句解析と構文解析に用いる
  - 構文木を構築するのに十分な情報をもつ

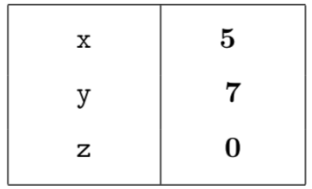
抽象構文 vs 具象構文

- 具象構文については考えない

- 曖昧さをなくすために必要ならば括弧を使う
  while x **<** y **do** (x **:=** x **+ 1;** y **:=** 0)
  又は
  (**while** x **<** y **do** x **:=** x **+ 1**)**;** y **:=** 0
- 算術演算子や論理演算子の通常の優先順位を仮定する
  1 + x ***** 2 は **1 +** (x *** 2**) であり (**1 +** x) *** 2** でない

算術式と論理式の意味論

状態

- 状態は集合 State = Var → Z の要素
  - Zは整数集合
- 別表現：表 

や リスト [x↦5, y↦7,z↦0]

- 検索：各状態sは各x∈ Varの値s xを定める． ※ s(x)とも記す.
- 更新：

$$(s[y \mapsto n])(x) = \begin{cases} n & \text{if } x = y \\ s(x) & \text{if } x \neq y \end{cases}$$

意味関数

- プログラムの構文（文法）と意味ははっきりと異なるもの

- 意味関数は構文要素をその意味を表す数学的対象へ写す全域関数

数字の意味

数字

- 二進表現の数字 n ∈ Num
- 関数 N : Num → Z は数字を整数に写す
  N ⟦**0**⟧　　＝ 0
  N ⟦**1**⟧　　＝ 1
  N ⟦n **0**⟧ ＝ ?
  N ⟦n **1**⟧ ＝ ?

- 記法：⟦ ⟧ は構文要素を囲む括弧

数字の意味

数字

- 二進表現の数字 n ∈ Num
- 関数 N : Num → Z は数字を整数Zに写す
  N ⟦**0**⟧ = 0
  N ⟦**1**⟧ = 1
  N ⟦n **0**⟧ = 2 · N ⟦n⟧
  N ⟦n **1**⟧ = 2 · N ⟦n⟧ + 1

※数字のフォントは
両辺で異なる

- 記法：⟦ ⟧ は構文要素を囲む括弧

算術式の意味論

- **2*x+y**といった式の値は変数x, yの値による.
  - 変数x, yの値は現在の状態で決まる
- 現在の状態sの算術式aの意味はA〚a〛sであり,
  次の意味関数で定まる：
  A : Aexp → (State → Z)
- 関数 A は一度に1つの引数をとる
  - A〚x **+ 1**〛 は State → Z という型の関数である
  - s x = 2 となる状態 s が与えられると A〚x **+ 1**〛s = 3 となる

算術式の意味論

$A\ [\![n]\!]\ s =$

$A\ [\![x]\!]\ s =$

$A\ [\![a_1 + a_2]\!]\ s =$

$A\ [\![a_1 * a_2]\!]\ s =$

$A\ [\![a_1 - a_2]\!]\ s =$

算術式の意味論

$$
\begin{aligned}
A\ [\![n]\!]\ s &= N\ [\![n]\!] \\
A\ [\![x]\!]\ s &= s\ x \\
A\ [\![a_1 \mathbf{+} a_2]\!]\ s &= A\ [\![a_1]\!]\ s + A\ [\![a_2]\!]\ s \\
A\ [\![a_1 \mathbf{*} a_2]\!]\ s &= A\ [\![a_1]\!]\ s \cdot A\ [\![a_2]\!]\ s \\
A\ [\![a_1 \mathbf{-} a_2]\!]\ s &= A\ [\![a_1]\!]\ s - A\ [\![a_2]\!]\ s
\end{aligned}
$$

※演算子のフォントは
両辺で異なる

算術式の意味論

- 左辺の**+**, ***, **–** と右辺の+,·,-は異なるもの
  - 右辺のは通常の算術演算子（数学的対象）
  - 左辺のは構文要素
- 算術式の形によって意味が定まっている

- s x = 3, s y = 1となる状態sが与えられたとき

A〚**10***x**+**y〛s　＝

＝

＝

算術式の意味論

- 左辺の**+**, *****, **–** と右辺の+,·,-は異なるもの
  - 右辺のは通常の算術演算子（数学的対象）
  - 左辺のは構文要素
- 算術式の形によって意味が定まっている

- s x = 3, s y = 1となる状態sが与えられたとき

$$
\begin{aligned}
A\llbracket \mathbf{10*}x\mathbf{+}y\rrbracket s &= A\llbracket \mathbf{10*}x\rrbracket s + A\llbracket y\rrbracket s \\
&= A\llbracket \mathbf{10}\rrbracket s \cdot A\llbracket x\rrbracket s + 1 \\
&=
\end{aligned}
$$

算術式の意味論

- 左辺の**+**, *****, **–** と右辺の+,·,-は異なるもの
  - 右辺のは通常の算術演算子（数学的対象）
  - 左辺のは構文要素
- 算術式の形によって意味が定まっている

- s x = 3, s y = 1となる状態sが与えられたとき

$$
\begin{aligned}
A\llbracket \mathbf{10*}x\mathbf{+}y\rrbracket s &= A\llbracket \mathbf{10*}x\rrbracket s + A\llbracket y\rrbracket s \\
&= A\llbracket \mathbf{10}\rrbracket s \cdot A\llbracket x\rrbracket s + 1 \\
&= A\llbracket \mathbf{10}\rrbracket s \cdot 3 + 1 = 2\cdot 3 + 1 = 7
\end{aligned}
$$

結合的定義

- 意味論は関数の結合的な定義で定まる.
  - それぞれの基本要素に対して意味が定まる
  - それぞれの複合要素を構成する方法に対して意味が構成される
    - 複合要素の各部分に対する意味から構成される

論理式の意味論

- 算術式の場合と同様
- 論理式の意味関数
  $B : Bexp \to (State \to T)$
- 真偽値の集合Tはtt(真)とff(偽)からなる

論理式の意味論

$B\ [\![\mathbf{true}]\!]\ s = $

$B\ [\![\mathbf{false}]\!]\ s = $

$B\ [\![a_1 = a_2]\!]\ s = $

$B\ [\![a_1 \leqq a_2]\!]\ s = $

$B\ [\![\neg b]\!]\ s = $

$B\ [\![b_1 \wedge b_2]\!]\ s = $

論理式の意味論

$$
\begin{aligned}
&\mathrm{B}\,[\![\mathbf{true}]\!]\,s = \mathrm{tt}\\
&\mathrm{B}\,[\![\mathbf{false}]\!]\,s = \mathrm{ff}\\
&\mathrm{B}\,[\![a_1 = a_2]\!]\,s = \begin{cases} \mathrm{tt} & \text{if } \mathrm{A}\,[\![a_1]\!]\,s = \mathrm{A}\,[\![a_2]\!]\,s \\ \mathrm{ff} & \text{if } \mathrm{A}\,[\![a_1]\!]\,s \neq \mathrm{A}\,[\![a_2]\!]\,s \end{cases}\\
&\mathrm{B}\,[\![a_1 \leqq a_2]\!]\,s = \begin{cases} \mathrm{tt} & \text{if } \mathrm{A}\,[\![a_1]\!]\,s \leqq \mathrm{A}\,[\![a_2]\!]\,s \\ \mathrm{ff} & \text{if } \mathrm{A}\,[\![a_1]\!]\,s > \mathrm{A}\,[\![a_2]\!]\,s \end{cases}\\
&\mathrm{B}\,[\![\neg b]\!]\,s = \begin{cases} \mathrm{tt} & \text{if } \mathrm{B}\,[\![b]\!]\,s = \mathrm{ff} \\ \mathrm{ff} & \text{if } \mathrm{B}\,[\![b]\!]\,s = \mathrm{tt} \end{cases}\\
&\mathrm{B}\,[\![b_1 \wedge b_2]\!]\,s = \begin{cases} \mathrm{tt} & \text{if } \mathrm{B}\,[\![b_1]\!]\,s = \mathrm{tt} \text{ かつ } \mathrm{B}\,[\![b_2]\!]\,s = \mathrm{tt} \\ \mathrm{ff} & \text{if } \mathrm{B}\,[\![b_1]\!]\,s = \mathrm{ff} \text{ 又は } \mathrm{B}\,[\![b_2]\!]\,s = \mathrm{ff} \end{cases}
\end{aligned}
$$

意味論の性質

数学的帰納法

- 数学的帰納法は自然数(0又は正の整数)で
  サイズが与えられるものの性質を証明するのに使われる

- $2^n \leq (n+1)! \leq (n+1)^n$ を証明せよ
- 基底段階：n=0のとき$1 \leq 1 \leq 1$であることから明らか.
- 帰納段階：n=m+1>0のとき,
  $2^m \leq (m+1)! \leq (m+1)^m$ を仮定して
  $2^{m+1} \leq (m+1+1)! \leq (m+1+1)^{m+1}$ を証明する.

構造帰納法

- 構造帰納法は意味論の性質を証明するのに使われる.
  数学的帰納法に似ている.

- 基底段階：全ての基本要素で性質が成り立つことを証明する
- 帰納段階：全ての複合要素で性質が成り立つことを証明する
  複合要素を構成する全ての要素で性質が成り立つことを仮定して複合要素自身で性質が成り立つことを証明する

- （構造帰納法は要素のサイズを上界とする数学的帰納法として説明できる）

数字の意味論

- 関数 N ⟦ ⟧ は全域か？
  すなわち任意の n ∈ Num に対して
  ただ1つのi ∈ Z が存在してN ⟦n⟧ = i か？
- 答えはYes．構造帰納法で示せる．

数字の意味論

- 関数 N ⟦ ⟧ は全域か？
  すなわち任意の n ∈ Num に対して
  ただ1つのi ∈ Z が存在してN ⟦n⟧ = i か？
- 答えはYes．構造帰納法で示せる．

- n=**0**のとき：？
- n=n’ **0**のとき：？
- n=**1**やn=n’ **1**のときも同様

自由変数

- 算術式の値がその中で出現する変数の値にのみで定まることを形式化したい
- 式a中の自由変数の集合FV(a)：
  $FV(n) = \emptyset$
  $FV(x) = \{x\}$
  $FV(a_1 + a_2) = FV(a_1) \cup FV(a_2)$
  $FV(a_1 * a_2) = FV(a_1) \cup FV(a_2)$
  $FV(a_1 - a_2) = FV(a_1) \cup FV(a_2)$
- 補題：状態sとs'が任意の$x \in FV(a)$に対して
  $s\ x = s'\ x$ならば，$A\ [\![a]\!]\ s = A\ [\![a']\!]\ s'$.
  - 構造帰納法で証明できる.

理解確認

自由変数

- FV(x**+1**)　　＝？
- FV(x**+**y*****x)　　＝？

理解確認

自由変数

- FV(x**+1**)　　＝{ x }
- FV(x**+**y*****x)　　＝{ x, y }

代入

- 状態は変数に適用し値が返る　→　代入は式に適用し式が返る
- 式a中の変数yをa_0にする代入：

$$
\begin{aligned}
n[y \mapsto a_0] &= n \\
x[y \mapsto a_0] &= \begin{cases} a_0 & \text{if } x = y \\ x & \text{if } x \neq y \end{cases} \\
(a_1 + a_2)[y \mapsto a_0] &= (a_1[y \mapsto a_0]) + (a_2[y \mapsto a_0]) \\
(a_1 * a_2)[y \mapsto a_0] &= (a_1[y \mapsto a_0]) * (a_2[y \mapsto a_0]) \\
(a_1 - a_2)[y \mapsto a_0] &= (a_1[y \mapsto a_0]) - (a_2[y \mapsto a_0])
\end{aligned}
$$

- 補題：任意の状態sに対して

$$\mathcal{A}\llbracket a[y \mapsto a_0] \rrbracket\, s = \mathcal{A}\llbracket a \rrbracket\, (s[y \mapsto \mathcal{A}\llbracket a_0 \rrbracket\, s])$$

代入の例

$$(x\mathbf{+1})[x \mapsto \mathbf{3}] = \mathbf{3+1}$$
$$(x\mathbf{+}y\mathbf{*}x)[x \mapsto y\mathbf{-5}] = (y\mathbf{-5})\mathbf{+}y*(y\mathbf{-5})$$

$$((x\mathbf{+1})[x \mapsto \mathbf{3}])[x \mapsto \mathbf{4}] = \mathbf{3+1}$$

前置の代入

- 状態は変数に適用し値が返る　→　代入は式に適用し式が返る
- 式a中の変数yをa_0にする代入：

$$(s[y \mapsto a_0])n = n$$

$$(s[y \mapsto a_0])x = \begin{cases} a_0 & \text{if } x = y \\ s\ x & \text{if } x \neq y \end{cases}$$

$$(s[y \mapsto a_0])(a_1 + a_2) = (s[y \mapsto a_0])a_1 + (s[y \mapsto a_0])a_2$$

$$(s[y \mapsto a_0])(a_1 * a_2) = (s[y \mapsto a_0])a_1 * (s[y \mapsto a_0])a_2$$

$$(s[y \mapsto a_0])(a_1 - a_2) = (s[y \mapsto a_0])a_1 - (s[y \mapsto a_0])a_2$$

まとめ

- While言語：構文と意味論
- 証明の技術：構造帰納法