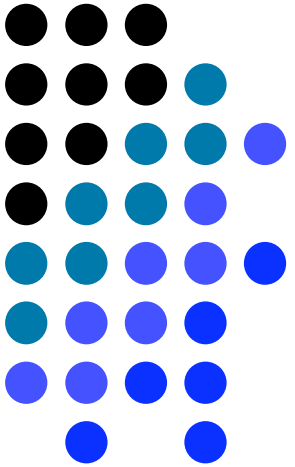
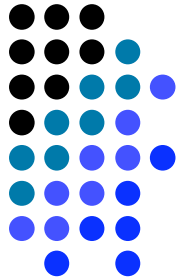
Semantics with Applications

(2章4節)



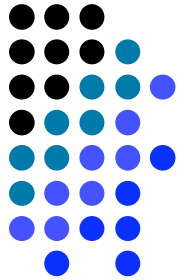
2.4 Whileの拡張



２つの操作的意味論のそれぞれの利点・欠点を示すために、Whileにいろいろな拡張を加えていく。

Abortion
Non-determinism(非決定性orの導入)
Parallelism(並列処理)
の3つ

Abortion



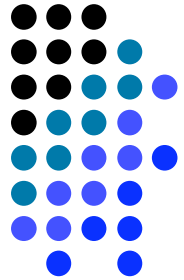
- abort:プログラムの実行を完全に止めるというもの
- つまりabortと while true do skip は違う

  → abortは実行を止める（プログラムは完全に止まる）が、whileの方は永遠にループし続ける

- また、abortと skip も違う

  → abortは実行を止める（プログラムは完全に止まる）が、skipはskipに続くものは実行される

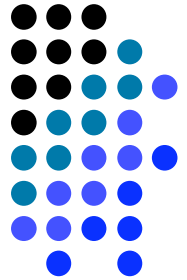
abort を含めた新しい構文



$S ::= x := a \mid skip \mid S_1;S_2$
$\mid$ if b then S_1 else $S_2 \mid$ while b do S
$\mid$ abort

集合の定義は繰り返さないが、それは拡張された言語に合うよう修正されてると仮定する。
よって残っている仕事は新しい推移関係→と⇒を定義すること。しかし、abortはスタックなので新たに定義する必要はない

abortとskip(sos)

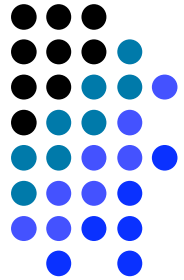


構造的操作的意味論ではabortとskipが同値でないのは明らか

＜skip , s＞⇒ s (状態sからのskip導出列)

＜abort , s＞ (状態sからのabort導出列)

abortとskip(sos)



同様にabortと while true do skip も同値ではない

＜while true do skip , s＞

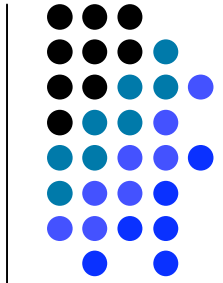
⇒＜ if true then (skip;while true do skip)
else skip , s＞

⇒＜ skip; while true do skip , s＞

⇒＜ while true do skip , s＞⇒・・・・

となり、while true do skip では無限の導出列だが、abortには導出列はない。

abortとskip(ns)



自然意味論でも skip と abort は意味論的に
等価になれないことは明らか。

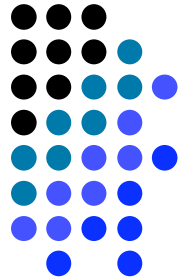
abortとskip(ns)

しかし abort と while true do skip は同値であるという結果になる！

なぜなら自然意味論では実行が正確に終わるものしか考えていなかったから。

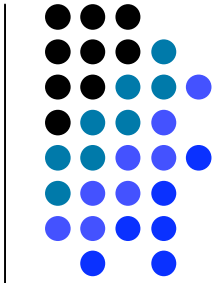
もし、＜S , s＞→s’ が導出木を持たないとき、それが abort のスタック構造によるものかループの実行によるものか区別がつかない。

NS vs SOS (abort)



- 自然意味論ではループと基準を外れた終了(abort)の区別が出来ない
- 構造的操作的意味論ではループは無限の導出列によって、基準を外れた終了(abort)はスタック構造の有限な導出列で示される

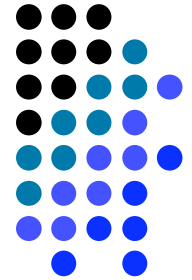
  ⇒つまり abort においては区別できる分、SOS の方が良い



演習2.31

定理2.26はWhileの自然意味論と構造的操作的意味論が同値であると言っている。

abort で拡張されたWhileは同様の結果を持つか議論せよ。



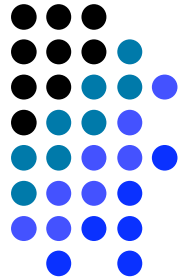
演習2.32

Whileを命令文 assert b before S で拡張せよ。これはもし b が true に評価されたならSを実行し、されなければプログラムの実行をabort するというものである。

これを表現するように表2.2の構造的操作的意味論を拡張せよ。(While が abort命令を持たないと仮定する)

assert true before S と S は意味論的に同値だが、assert false before S は while true do skip や skip と同値でないことを示せ。

Non-determisim (非決定論)

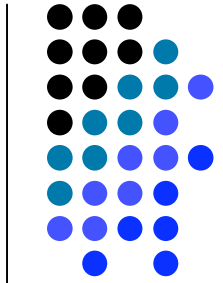


- S_1 or S_2 :実行するのをS_1かS_2か非決定的に選ぶことが出来る

  新しい構文は次の通り

  S ::= x := a | skip | S_1; S_2
  | if b then S_1 else S_2 | while b do S
  | S_1 or S_2

非決定論の実行例



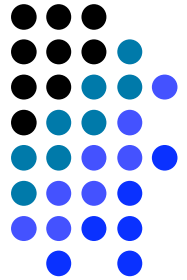
x := 1 or (x := 2; x := x + 2)

→ 左を選んで
x := 1

→ 右を選んで
x := 4

の二通りが考えられる

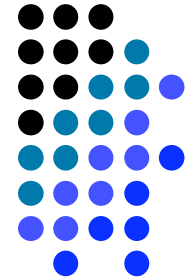
自然意味論への非決定論の追加



- 自然意味論では、図2.1 に下の2つの規則を追加することで拡張が実現可能に

$$[\mathrm{or}^1_{\mathrm{ns}}] \quad \frac{\langle S_1,\ s\rangle \to s'}{\langle S_1\ \mathtt{or}\ S_2,\ s\rangle \to s'}$$

$$[\mathrm{or}^2_{\mathrm{ns}}] \quad \frac{\langle S_2,\ s\rangle \to s'}{\langle S_1\ \mathtt{or}\ S_2,\ s\rangle \to s'}$$



（非決定論で拡張した自然意味論）

< x := 1 or (x := 2; x := x + 2) , s >

を考えてみる

- < x := 1 or (x := 2; x := x + 2) , s >→s[x→1]
- < x := 1 or (x := 2; x := x + 2) , s >→s[x→4]

の2つの導出木が得られる。

しかし、x := 1 を while true do skip に変えてみると

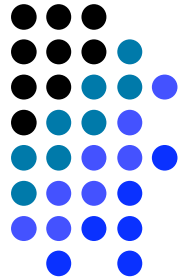
…

- < (while true do skip) or (x := 2; x := x + 2)>

→s[x→4] の1つの導出木のみしか得られない

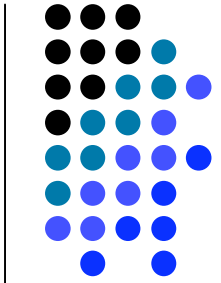
（前ページの上の仮定が存在しないため）

SOSへの非決定論の追加



- 構造的操作的意味論では、表2.2 に下の2つの規則を追加することで拡張が実現可能に

$$[\mathrm{or}^1_{\mathrm{sos}}] \quad \langle S_1 \ \mathtt{or}\ S_2,\ s\rangle \Rightarrow \langle S_1,\ s\rangle$$
$$[\mathrm{or}^2_{\mathrm{sos}}] \quad \langle S_1 \ \mathtt{or}\ S_2,\ s\rangle \Rightarrow \langle S_2,\ s\rangle$$



（非決定論で拡張した構造的操作的意味論）

< x := 1 or (x := 2; x := x + 2) , s >

　を考えてみる

・< x := 1 or (x := 2; x := x + 2) , s >⇒s[x|→1]

・< x := 1 or (x := 2; x := x + 2) , s >⇒s[x|→4]

の2つの導出列が得られる。

x := 1 を while true do skip に変えてみると…

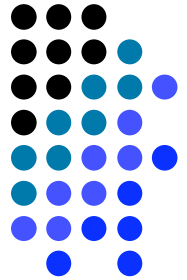
・<(while true do skip) or (x := 2; x := x + 2)>

　　⇒＜while true do skip , s＞

　　⇒・・・・

または有限で ⇒ s[x|→4] の2つの導出列が得られる

NS vs SOS(非決定性)



- 自然意味論では非決定性は可能ならループを抑制出来る

- 構造的操作的意味論では非決定性はループを抑制できない

⇒後者は”wrong”分岐を選べるが、
前者は”right”分岐しか常に選ばない

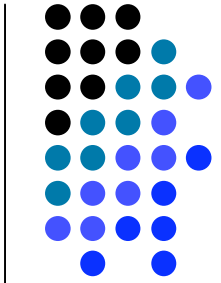
演習2.33

命令文 x := -1; while x <= 0 do (x := x - 1 or x:= (-1) * x)を考えなさい。

状態 s を与えられた時、自然意味論によると結果として起きるかもしれない最終状態の集合を記述せよ。

その上で構造的操作的意味論によって詳細に述べられる導出列の集合を記述しなさい。

この議論に基づいて、あなたが自然意味論と構造的操作的意味論が等価であると見なすかどうかについて議論しなさい。



演習2.34

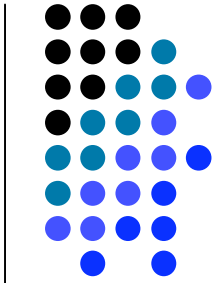
Whileを命令文 random(x) で拡張しよう。

それは x の値を何か正の自然数に変えるというものである。

これを表現するために構造的操作的意味論と同様に、自然意味論も拡張しなさい。

While を or 構造で拡張している場合 、random(x)が余分な構造であるかどうか議論しなさい。

Parallelism(並列処理)

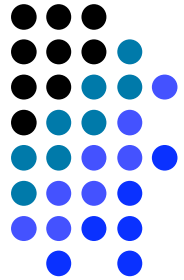


- S_1 par S_2: S_1 と S_2 の両方の命令文が実行されなくてはならないが、交互的な実行が可能である。(どうやって実行されるかは次ページ)

新しい構文は次の通り

S ::= x := a | skip | S_1;S_2
| if b then S_1 else S_2 | while b do S
| S_1 par S_2

並列処理の実行例



x := 1 par (x := 2; x := x + 2)

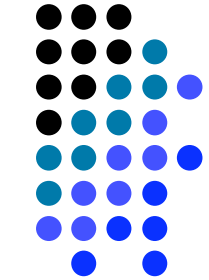
→ x := 1 , x := 2 , x := x+2 の順番で処理してx := 4

→ x := 2 , x := 1 , x := x+2 の順番で処理してx := 3

→ x := 2 , x := x+2 , x := 1 の順番で処理してx := 1

の三通りが考えられる

SOSへの並列処理の追加



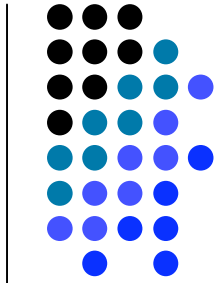
- 構造的操作的意味論では、図2.2 に次の4つの規則を追加することで拡張が実現可能に

$$[\mathrm{par}^1_{\mathrm{sos}}]\ \frac{\langle S_1,\ s\rangle \Rightarrow \langle S_1',\ s'\rangle}{\langle S_1\ \mathbf{par}\ S_2,\ s\rangle \Rightarrow \langle S_1'\ \mathbf{par}\ S_2,\ s'\rangle}$$

$$[\mathrm{par}^2_{\mathrm{sos}}]\ \frac{\langle S_1,\ s\rangle \Rightarrow s'}{\langle S_1\ \mathbf{par}\ S_2,\ s\rangle \Rightarrow \langle S_2,\ s'\rangle}$$

$$[\mathrm{par}^3_{\mathrm{sos}}]\ \frac{\langle S_2,\ s\rangle \Rightarrow \langle S_2',\ s'\rangle}{\langle S_1\ \mathbf{par}\ S_2,\ s\rangle \Rightarrow \langle S_1\ \mathbf{par}\ S_2',\ s'\rangle}$$

$$[\mathrm{par}^4_{\mathrm{sos}}]\ \frac{\langle S_2,\ s\rangle \Rightarrow s'}{\langle S_1\ \mathbf{par}\ S_2,\ s\rangle \Rightarrow \langle S_1,\ s'\rangle}$$



（並列処理で拡張した構造的操作的意味論）

< x := 1 par (x := 2; x := x + 2) , s >

を考えてみる

・< x := 1 par (x := 2; x := x + 2) , s >

⇒ < x := 2; x := x + 2 , s > s[x|→1]

⇒ < x := x + 2 , s > s[x|→2] ⇒ s[x|→4]

・< x := 1 par (x := 2; x := x + 2) , s >

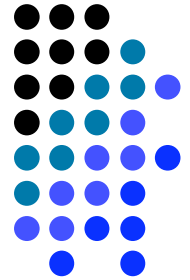
⇒ < x := 1 par x := x + 2 , s > s[x|→2](*)

⇒ < x := 1 , s > s[x|→4] ⇒ s[x|→1]

・(*) ⇒ < x := x + 2 , s > s[x|→1] ⇒ s[x|→3]

の三通りの導出列が得られる。

自然意味論への並列処理の追加



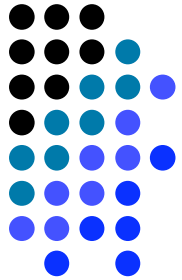
- 構造的操作的意味論では、図2.2 に次の2つの規則を追加することで拡張したい・・

$$[\mathrm{par}^1_{\mathrm{ns}}] \quad \frac{\langle S_1,\ s\rangle \to s',\langle S_2,\ s'\rangle \to s''}{\langle S_1\ \mathbf{par}\ S_2,\ s\rangle \to s''}$$

$$[\mathrm{par}^2_{\mathrm{ns}}] \quad \frac{\langle S_2,\ s\rangle \to s',\langle S_1,\ s'\rangle \to s''}{\langle S_1\ \mathbf{par}\ S_2,\ s\rangle \to s''}$$

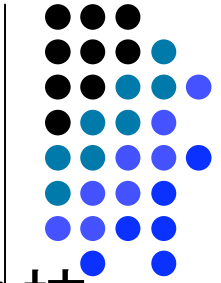
しかし、この規則ではS_2の前にS_1を実行しているだけ（もしくは逆）なので、これでは並列処理になっていない。

NS vs SOS (並列処理)



- 自然意味論では交互的な実行（並列処理）は表現できない

- 構造的操作的意味論では計算の小さいステップに集中しているため、簡単に交互的な実行（並列処理）が表現できる



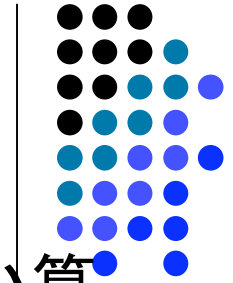
演習2.35

While の par 構造に加えて protect S end を含む拡張を考えなさい。
これは命令文 S が極小の構成要素として実行されるということで、たとえば

x := 1 par protect (x:= 2; x := x + 2) end

では 1 と 4 の2つの結果しか可能でないようにするものである。

構造的操作的意味論をこれを表現するように拡張せよ。自然意味論では拡張された言語を明確に述べられるか？



演習2.36

式の個々の部分が並列に計算されるかもしれない算術式の構造的操作的意味論を明確に述べよ。

*A*によって明確に述べられた結果をまだ得ることを示そうと試せ。