Procedury

Rozszerzamy składnię języka o procedury. Musimy rozszerzyć definicję deklaracji i instrukcji.

- $d ::=$ var $v \mid d_1; d_2 \mid$ proc $P; c$

- $c ::=$ skip $\mid v := e \mid c; c' \mid$ if b then c else $c' \mid$ while b do $c \mid$ begin $d; c$ end $\mid$ call P

Sposoby przekazywania parametrów:

1. przez zmienną
2. przez wartość
3. przez nazwę

Na początek przyjmujemy następujące założenia:

1. brak rekursji
2. procedury bezparametrowe

Przykład 1

```
program proc;
   var x:integer;
   procedure druk;
     begin
       writeln(x);
     end;
   procedure stat;
     begin var x:integer;
           x:=2;
           druk
     end;
begin
  x:=1;
  stat
end.
```

Pytanie: co wydrukuje procedura **druk** ?
Możliwości są dwie:

1. Pascal, Algol-60 – 1 (*semantyka statyczna*) – zmienne globalne pochodzą ze środowiska deklaracji.

2. Lisp, POP-2 – 2 (*semantyka dynamiczna*) – zmienne globalne pochodzą ze środowiska wywołania.

Semantyka statyczna

Kategorie syntaktyczne jak poprzednio.

Dziedziny semantyczne jak poprzednio oraz

- $Proc = [S \to S_\perp]$
- $Dv = Loc + Proc + \{ \underline{\text{puste}} \}$
- $Env = [Ide \to Dv]$

Równania semantyczne

Aexp i **Bexp** jak poprzednio.

Decl

$$\mathcal{D}[[\ \underline{\text{var}}\ v]] = \lambda\rho \in Env.\lambda s \in S.(\rho[(new(s))_1/v], (new(s))_2)$$

$$\mathcal{D}[[d_1; d_2]] = \lambda\rho \in Env.\lambda s \in S.\mathcal{D}[[d_2]](\mathcal{D}[[d_1]]\rho s)_1(\mathcal{D}[[d_1]]\rho s)_2$$

$$\mathcal{D}[[\ \underline{\text{proc}}\ P; c]] = \lambda\rho \in Env.\lambda s \in S.(\rho[f/P], s)$$

gdzie $f \in Proc$ jest zdefiniowana następująco:

$$f = \lambda s' \in S.\mathcal{C}[[c]]\rho s'$$

Com jak poprzednio oraz

$$\mathcal{C}[\![\ \underline{\text{call}}\ P]\!] = \lambda\rho \in Env.\lambda s \in S.\textbf{if}\ \rho P \notin Proc$$
$$\textbf{then}\ \perp_S\ \textbf{else}\ (\rho P)s.$$

Semantyka dynamiczna

Dziedziny semantyczne jak poprzednio z wyjątkiem dziedziny Proc. Nowa definicja

- $Proc = [Env \to [S \to S_\perp]]$

Decl

$$\mathcal{D}[[\ \underline{\text{proc}}\ P; c]] = \lambda\rho \in Env.\lambda s \in S.(\rho[f/P], s)$$

gdzie $f \in Proc$ jest zdefiniowana następująco:

$$f = \lambda\rho' \in Env.\lambda s' \in S.\mathcal{C}[[c]]\rho' s'$$

Com

$$\mathcal{C}[\![\ \underline{\text{call}}\ ]\!] = \lambda\rho \in Env.\lambda s \in S.\textbf{if}\ \rho P \notin Proc$$
$$\textbf{then}\ \perp_S\ \textbf{else}\ (\rho P)\rho s.$$

Semantyka statyczna – rekurencja

Dziedziny semantyczne

Wracamy do poprzedniej definicji dziedziny Proc.

- $Proc = [S \to S_\perp]$

Decl

$\mathcal{D}[[\ \underline{\text{proc}}\ P; c]] = \lambda\rho \in Env.\lambda s \in S.(\rho[f/P], s)$

operator $F : Proc \to Proc$ jest zdefiniowany następująco:

$F = \lambda g \in Proc.\lambda s' \in S.\mathcal{C}[[c]](\rho[g/P])s'$

$f = fix(F)$

Com

wywołanie procedury jak w przypadku semantyki statycznej bez rekurencji.

Procedury z parametrem

Składnia

- $d ::=$ var $v \mid d_1; d_2 \mid$ proc $P(v); c$

- $c ::=$ skip $\mid v := e \mid c; c' \mid$ if b then c else $c' \mid$ while b do $c \mid$ begin $d; c$ end $\mid$ call $P(v)$

Przekazywanie parametrów przez zmienną

Semantyka statyczna

Kategorie syntaktyczne jak poprzednio.

Dziedziny semantyczne jak poprzednio oraz

- $Proc1 = [Loc \to Proc]$
- $Dv = Loc + Proc1 + \{ \underline{\text{puste}} \}$
- $Env = [Ide \to Dv]$

Równania semantyczne

Aexp i **Bexp** jak poprzednio.

Decl

$$\mathcal{D}[[\ \underline{\text{var}}\ v]] = \lambda\rho \in Env.\lambda s \in S.(\rho[(new(s))_1/v], (new(s))_2)$$

$$\mathcal{D}[[d_1; d_2]] = \lambda\rho \in Env.\lambda s \in S.\mathcal{D}[[d_2]](\mathcal{D}[[d_1]]\rho s)_1(\mathcal{D}[[d_1]]\rho s)_2$$

$$\mathcal{D}[[\ \underline{\text{proc}}\ P(v); c]] = \lambda\rho \in Env.\lambda s \in S.(\rho[f/P], s)$$

gdzie $f \in Proc1$ jest zdefiniowana następująco:

$$f = \lambda l \in Loc.\lambda s' \in S.\mathcal{C}[[c]](\rho[l/v])s'$$

Com jak poprzednio oraz

$$\mathcal{C}[\![\ \underline{\text{call}}\ P(v)]\!] = \lambda\rho \in Env.\lambda s \in S.\textbf{if}\ \rho P \notin Proc1$$
$$\textbf{then}\ \perp_S\ \textbf{else if}\ \rho v \notin Loc\ \textbf{then}\ \perp_S\ \textbf{else}\ (\rho P)(\rho v)s.$$

Przekazywanie parametrów przez wartość

Proces przekazywania parametru przez wartość składa się z dwóch etapów:

1. Obliczenie wartości parametru aktualnego.

2. Przydział lokacji do zapamiętania tej wartości.

Proces wykonania procedury składa się także z dwóch etapów:

1. Wykonanie instrukcji wywołania procedury.

2. Wykonanie treści procedury.

W związku z tym przekazywanie parametru przez wartość możemy zorganizować na kilka różnych sposobów.

1. **Obliczenie** wartości parametru aktualnego i **przydział lokacji** do zapamiętania tej wartości wykonuje **instrukcja wywołania** procedury. Funkcja reprezentująca procedurę "pobiera" wartość parametru i lokację.

   Nowa definicja $Proc1$:

   - $Proc1 = [Sv \to [Loc \to [S \to S_\perp]]]$

$\mathcal{D}$[[proc $P(v); c$]] $= \lambda\rho \in Env.\lambda s \in S.(\rho[f/P], s)$

gdzie $f = \lambda w \in Sv.\lambda l \in Loc.\lambda s' \in S.\mathcal{C}[[c]](\rho[l/v])(s'[w/l])$

$\mathcal{C}$[[call $P(e)$]] $= \lambda\rho \in Env.\lambda s \in S.$ **if** $\rho P \notin Proc1$ **then** $\perp_S$ **else**

if $\mathcal{A}[[e]]\rho s \notin Sv$ **then** $\perp_S$ **else** $(\rho P)(\mathcal{A}[[e]]\rho s)(new(s))_1(new(s))_2$.

2. **Obliczenie** wartości parametru aktualnego wykonuje **instrukcja wywołania** procedury, a **przydział lokacji** do jej zapamiętania wykonuje **funkcja reprezentująca procedurę**.

   - $Proc1 = [Sv \to [S \to S_\perp]]$

$\mathcal{D}[[\ \underline{\text{proc}}\ P(v); c]] = \lambda\rho \in Env.\lambda s \in S.(\rho[f/P], s)$

gdzie $f = \lambda w \in Sv.\lambda s' \in S.$ **let** $(l'', s'') = new(s')$ **in**

$\mathcal{C}[[c]](\rho[l''/v])(s''[w/l''])$

$\mathcal{C}[[\ \underline{\text{call}}\ P(e)]] = \lambda\rho \in Env.\lambda s \in S.$**if** $\rho P \notin Proc1$ **then** $\perp_S$ **else**

if $\mathcal{A}[[e]]\rho s \notin Sv$ **then** $\perp_S$ **else** $(\rho P)(\mathcal{A}[[e]]\rho s)s.$

3. **Instrukcja wywołania** procedury przekazuje do procedury **nieobliczony** parametr aktualny. **Funkcja reprezentująca procedurę oblicza** wartość parametru i **przydziela lokację**.

   Aby sformalizować pojęcie nieobliczonego parametru aktualnego wprowadzamy pojęcie domknięcia wyrażeń:

   - $Cl = [S \to Sv]$
   - $Proc1 = [Cl \to [S \to S_\perp]]$

$$\mathcal{D}[\![\underline{\text{proc}}\ P(v); c]\!] = \lambda\rho \in Env.\lambda s \in S.(\rho[f/P], s)$$

gdzie $f = \lambda\xi \in Cl.\lambda s' \in S.$**if** $\xi s' \notin Sv$ **then** $\perp_S$ **else**
let $(l'', s'') = new(s')$ **in** $\mathcal{C}[\![c]\!](\rho[l''/v])(s''[\xi s'/l''])$

$$\mathcal{C}[\![\underline{\text{call}}\ P(e)]\!] = \lambda\rho \in Env.\lambda s \in S.\textbf{if}\ \rho P \notin Proc1\ \textbf{then}\ \perp_S$$
else $(\rho P)(\mathcal{A}[\![e]\!]\rho)s.$