SpecVer & LEM'12 projects XX (2015) 1–3
Dombrova Research

Brulion – this version for your eyes only
Ćwiczenie pt. swap

Andrzej Salwicki
salwicki@mimuw.edu.pl
Dąbrowa Leśna

Streszczenie. Udowodnimy poprawność algorytmu *swap* czyli zamiana.

Znacie?
Znamy!
No to posłuchajcie ...

1. Zamiana wartości zmiennych

Niniejszy drobiazg stawia sobie kilka celów:

- udowodnić poprawność króciutkiego programu,
- zastanowić się nad pojęciem programu generycznego (w terminologii Ady) lub wzorca (w terminologii C++).

Niech T będzie typem, pierwotnym jak char, string, integer, boolean lub real, albo złożonym, zdefiniowanym przy pomocy deklaracji klasy T. Niech x, y, z będą zmiennymi typu T. Zamierzamy udowodnić następującą formułę algorytmiczną

$$(x = a \wedge y = b) \implies \{z := x;\ x := y;\ y := z\}(x = b \wedge y = a)$$

Każdy powie: ależ to oczywiste, ponieważ dowolne obliczenie wyznaczone przez ten program zamieni wartości zmiennych x i y. My chcemy pokazać, że wnioskowanie o semantycznych własnościach programów może być wolne od pojęcia obliczenia.

2. Rachunek

Będziemy po kolei eliminować instrukcje przypisania. Posługiwać się przy tym będziemy dwoma aksjomatami logiki algorytmicznej:

$$\{v := \tau\}\,\alpha(v) \equiv \alpha(v/\tau) \tag{Ax:=}$$

oraz

$$\{\,K;\,M\}\alpha \Leftrightarrow K\,M\,\alpha \tag{Ax;}$$

a także prawami rachunku zdań, w szczególności wykorzystujemy ekstensjonalność operacji logicznych. Np. takie dwie reguły wnioskowania

$$\frac{\alpha \Rightarrow \beta, \beta \equiv \gamma}{\alpha \Rightarrow \gamma} \qquad\qquad \frac{\alpha \Rightarrow \beta}{K\,\alpha \Rightarrow K\,\beta}$$

Zaczynamy

$$(6)\quad (x = a \wedge y = b) \Rightarrow \{\underbrace{z := x;\, x := y}_{K};\, \underbrace{y := z}_{M}\}\underbrace{(x = b \wedge y = a)}_{\alpha}$$

(6) is equivalent to (5) by (Ax;) and propositional calculus

$$(5)\quad (x = a \wedge y = b) \Rightarrow \overbrace{\{z := x;\, x := y\}}^{K}\overbrace{\{y := z\}}^{M}\underbrace{(x = b \wedge y = a)}_{\alpha(y)}$$

(5) is equivalent to (4) by (Ax:=) and propositional calculus

$$(4)\quad (x = a \wedge y = b) \Rightarrow \{z := x;\, x := y\}\underbrace{(x = b \wedge z = a)}_{\alpha(y/z)}$$

(4) is equivalent to (3) by (Ax;) and propositional calculus

$$(3)\quad (x = a \wedge y = b) \Rightarrow \{z := x\}\{x := y\}(x = b \wedge z = a)$$

(3) is equivalent to (2) by (Ax:=) and propositional calculus

$$(2)\quad (x = a \wedge y = b) \Rightarrow \{z := x\}(y = b \wedge z = a)$$

(2) is equivalent to (1) by (Ax:=) and propositional calculus

$$(1)\quad (x = a \wedge y = b) \Rightarrow (y = b \wedge x = a)$$ Tautology

3. Dowód

I to już jest dowód. Przypomnijmy, każda z formuł (1), (2), (3), (4), (5), (6) jest równowazna pozostałym formułom. Formuła (1) jest tautologią - jest więc prawdziwa. Wynika stąd, że prawdziwa jest formuła (6).

Przeczytajmy formułę (6) jeszcze raz, po polsku: *jeżeli* $(x = a \wedge y = b)$ *to po wykonaniu programu* $\{z := x;\, x := y;\, y := z\}$ *zachodzi warunek końcowy* $(x = b \wedge y = a)$. I o to nam chodziło.

4. instrukcja swap

Wyprzedzając dalszy ciąg rozważań wprowadzimy nowe polecenie swap zdefiniowane przez następującą deklarację procedury

```
unit swap: procedure(type T; inout x,y:T);
  var z: T
begin
  z:=x; x:=y; y:=z
end swap;
```

W zasięgu tej definicji polecenie **call** swap(A, u, v); gdzie A jest typem zdefiniowanym przez pewna klasę, natomiast u i v są zmiennymi typu A spowoduje zamianę wartości zmiennych u i v, ponieważ

$$(u = k \wedge v = l) \Rightarrow \{\textbf{call}\ swap(A, u, v)\}(u = l \wedge v = k).$$

5. Luźne uwagi końcowe

1. A może definicję instrukcji swap wstawić jako metodę do klasy Object?
2. W każdym razie polecenie takie może okazać się przydatne w algorytmach sortowania gdzie wystąpią testy x<y oraz zamiany call swap(x,y).

   Dokładniej, możemy przyjąć, że w programie sortującym występują relacja $<$ np. **if** A[i] < A[j] ... oraz polecenie
   call swap(A[k], A[l])
   Jeśli tak jest to program na pewno wyznacza pewną permutację ciagu A[1], A[2], ..., A[n]. (Nie musimy juz tego dowodzić!)