ς-kalkul rychle a pochopitelně

Petr "s3rvac" Zemek, s3rvac@seznam.cz
15. května 2007
Fakulta Informačních Technologií, Brno

Abstrakt

Tento článek navazuje na předchozí článek o λ-kalkulu a opět se snaží co nejpochopitelněji a formou analogií prezentovat základy formálního systému ς-kalkulu, především z pohledu předmětu Principy programovacích jazyků. Výklad je neformální, pro formálnější popis se můžete podívat do opory IPP. Tento článek si neklade za cíl plně nahradit studijní oporu, měl by sloužit především pro rychlejší pochopení ς-kalkulu. Proto se omlouvám za některá případná slovní spojení a zjednodušení.

1 O čem bude řeč aneb co to ten ς-kalkul je ...

ς-kalkul je příklad formálního popisu objektově orientovaného jazyka (dále jen OOP). Nám bude sloužit jako ukázka popisu OOP konstrukcí tak, jak je znáte z různých objektově orientovaných jazyků, ve formálním jazyce. Zde bych mohl opět použít moji oblíbenou frázi o programování v matematice :).

Budu se snažit ukazovat analogie mezi objektově orientovaným jazykem a mezi ς-kalkulem, nicméně jsem záměrně nezvolil žádný konkrétní jazyk (C++, Java, ...). Vše se budu ale snažit popsat tak, aby to bylo srozumitelné všem.

2 Základní konstrukce

Zde nastává menší problém, jelikož ς-kalkul se podle mě jako takový se hodí spíše pro popis prototypově orientovaných jazyků, ale já to budu z části prezentovat jakoby byl třídně orientovaný (v zájmu názornosti), takže se nesmíte divit, že některé z níže uvedených operací v jazycích jako je C++ a Java možné :).

Prvky ς-kalkulu se označují jako ς-termy a jsou následující.

2.1 Proměnné

Úplně stejný význam jako v jiných jazycích (tj "normální" proměnné :). Označují se většinou malými písmeny, např. můžeme mít objekt o.

2.2 Vytvoření (popis) objektu

Představte si následující třídu, která má jednu metodu a dva atributy (proměnné):

```
class A {
    int f() { return b; }
    int a;
    int b;
}
```

Úplně stejnou třídu (resp. už objekt, protože v ς-kalkulu třídy nejsou) vytvoříme v ς-kalkulu takto (nemá jméno):

$$[f = \varsigma(x)b]$$

Hranaté závorky znázorňují objekt, f je jméno metody, která má jeden parametr x a tento parametr bude při volání nahrazen objektem, který tuto metodu volá (analogie `this` v C++ a Javě). Bez tohoto parametru bychom nevěděli, který objekt tuto metodu volal. Následuje tělo metody, které v našem případě značí, že se bude vracet proměnná b, stejně jako v našem příkladě.

Konkrétní objekt (proměnnou), např. o, vytvoříme velice jednoduše a to:

$$o = [f = \varsigma(x)b]$$

Poznámka – tento popis je velmi zjednodušený, ve skutečnosti nám tam chybí ještě proměnné a a b, o nich bude řeč až později.

2.3 Volání metody objektu a výběr hodnoty atributu

V ς-kalkulu se metoda objektu volá úplně stejně jako v C++ a Javě, tedy pokud jsme si vytvořili objekt o, který má metodu f, tak ji zavoláme takto:

$$o.f$$

Jediný rozdíl je, že v ς-kalkulu se při volání metody nepíšou kulaté závorky. Hodnotu atributu b bychom získali takto:

$$o.b$$

2.4 Změna těla metody a změna hodnoty atributu

V nám známých jazycích sice nemůžeme měnit tělo metody, ale tady to lze. Provádí se to pomocí "operátoru" $\Leftarrow$. Pokud budeme chtít, aby nám metoda f vracela místo hodnoty (proměnné) b hodnotu (proměnnou) a, tak to zapíšeme následujícím způsobem:

$$o.f \Leftarrow \varsigma(x)a$$

Ve výsledku to znamená, že nám vznikne nový objekt (funkcionální sémantika), který má změněnou metodu f tak, jak jsme chtěli. Hodnota atributu by se měnila podobně, vše bude jasnější, až popíšu, jak to doopravdy v ς-kalkulu funguje.

3 Jak je to tedy s atributy a popisem objektů?

Teď, když už znáte základy popisu objektů a ς-termů (to jsou všechny ty prvky zmíněné v předchozí části) v ς-kalkulu, tak můžu vysvětlit i podrobnosti, které jsem zatím záměrně zamlčel.

V ς-kalkulu objekty ve skutečnosti žádné atributy nemají (tak, jak je známe z jiných jazyků). Existují tam pouze metody, tudíž veškerá práce s atributy by se dala přirovnat k metodám `GetAttribute()` a `SetAttribute()`. Kompletní popis naší třídy a vytvoření objektu o by tedy mohl vypadat takto:

$$o = [f = \varsigma(x_1)b,\ b = \varsigma(x_2)b',\ a = \varsigma(x_3)a']$$

Definice metody f zůstala stejná, ale přibyla nám tam metoda b, která vrací hodnotu (proměnnou) b' (nelze ji pojmenovat jen b, protože to už je název metody). Analogicky to je s atributem a, který je teď metoda, která vrací "svou hodnotu".

Volání metody f objektu o (tedy $o.f$) bude vypadat následovně. Zavolá se metoda f, za její parametr x_1 se dosadí náš objekt[1], který metodu volal (vzpomínáte – musíme vědět, který objekt metodu volá). Tato informace nám ale zatím k ničemu není, protože se s proměnnou x_1 v těle metody nepracuje. Výsledkem tedy bude, že se zavolá metoda b, opět proběhne nahrazení, tentokrát o za x_2 a vrátí se nám proměnná b'.

[1] Podrobnosti viz. sekce *Redukce – sémantika ς-kalkulu*

4 Volné a vázané proměnné

Analogicky jako v λ-kalkulu se zde vyskytují dva typy proměnných – *volné* a *vázané*. Myslím, že bude stačit vědět, že volnost proměnné určuje rozsah platnosti této proměnné a že proměnná může být v daném výrazu buď volná nebo vázaná, ale ne obojí zároveň.

5 Substituce proměnné

Už jsem ji zmiňoval v souvislosti s voláním metody objektu, tak nyní pouze stručně popíšu zápis a význam. Subsituce se zapisuje pomocí dvou složených závorek $\{\!\{ co \leftarrow zaco \}\!\}$. Takže když zavoláme metodu f objektu o, dojde k následující substituci:

$$[f = \varsigma(x)b]\{\!\{ x \leftarrow o \}\!\}$$

Znamená to, že se název objektu (proměnná) substituuje za parametr funkce. Tenhle příklad zřejmě není úplně správný podle λ-kalkulu, ale jako ukázka to stačí (ve skutečnosti by zápis subsituce při volání metody vypadal jinak).

6 Redukce – sémantika ς-kalkulu

Asi si říkáte něco ve smyslu „jéje, zase nějaké redukce“, ale tyhle jsou celkem v pohodě :). V ς-kalkulu jsou celkem dvě redukce a jedná se o vlastní výpočet, neboli to, co můžeme v ς-kalkulu provádět – výpočet libovolného ς-termu (neboli výrazu) se zapisuje formou posloupnoti redukčních kroků. Pokud se tedy b redukuje v jednom kroku na c, tak píšeme $b \rightarrowtail c$.

6.1 Redukce invokace

Volání metody – například pokud máme náš objekt o, tak při invokaci metody f se provede následující redukce: $o.f \rightarrowtail b\{\!\{ x \leftarrow o \}\!\}$, takže výsledkem je b, což je skutečně “výsledek těla” naší původní metody f.

6.2 Redukce modifikace

Změna těla metody – například tímto zápisem: $o.f \Leftarrow \varsigma(x)a \rightarrowtail [f = \varsigma(x)a]$ vznikne nový objekt s upravenou metodou f, která bude “vracet proměnnou a”.

7 Závěr

Tak jo, jsme na konci a já (opět) doufám, že se mi to aspoň trochu podařilo vysvětlit. ς-kalkul je více podobný tomu, co známe z objektových programovacích jazyků než λ-kalkul a proto je třeba si všechny kontrukce představovat tak, jak jsou ve skutečnosti implementované v jazycích jako je C++ a Java (omlouvám se, že pořád dávám jako příklad jen ty dva jazyky :). Toť vše, mějte se.

Reference

[1] Kolář D., *Studijní opora k předmětu IPP, II. část*, 2006

[2] Wikipedia, *Internetová encyklopedie*, http://wikipedia.org/