Jutustus tüübikontrollist

Asko Tiidumaa

7. Detsember 2004

Milleks tüübikontroll?

- Tüüp: väärtuste ja mõistlike operatsioonide hulk
- Semantilised kitsendused:
  - Operaator ja sobimatu tüübiga arg
  - Operaatori ja (coercion) valik
    - eg 3.0+1
  - Polümorfsete funktsioonide tuvastus

Tüübid (1/2)

- Tüübid
  - Sarnaste omadustega väärtused
  - Defineerib keel ja/või programmeerija
- Tüübisüsteem
  - Tüüpide hulk mingi keele jaoks
  - Reeglid, millega tüüpide abil suunata programmi käitumist

Tüübid (2/2)

- Näiteid tüübireeglitest:
  - Kui liitmisoperaatori mõlemad argumendid on täisarvud, siis on ka tulemus täisarv
- Miks see hea võiks olla?
  - Programmi töö käigus ei esine ootamat
  - Väljendusrikkus (overloading, polymorf
  - Lisainformatsioon koodigeneraatorile

Tüübisüsteem

- Igal keele elemendil (operaator, avaldis, lause, ...) on tüüp
  - Baastüübid: int, real, char, ...
  - Kogumid: massiiv, kirje, viit, fn, ...
- Tüübisüsteem koosneb reeglitest mis seob tüübiavaldised keele elementidega
- Vahel ka tüübid void ja typeError

Tüübiavaldised

- Kõik baastüübid on tüübiavaldised
- Lisaks veel tüübikonstruktorid:
  - Massiiv array(1..I,int)
  - Produkt? T_1 x T_2
  - Kirje record((v_1 x t_1) x ...)
  - Viit ptr(T)
  - Funktsioon int x int ? x

Tüübisüsteemi omadused

- Verifitseeritav (*decidably verifiable*)
  - Eksisteerib kontrollialgoritm
- Transparentne (*transparent*)
  - Tüübikontroll ja veaolukorrad on selgelt tõlgendatavad ning ennustatavad
- Rakendatav (*enforceable*)
  - Võimalikult palju staatilist kontrolli

Tüübikontroll

- Tüübikontrollija
  - Tugev/nõrk, staatiline/dünaamiline
- Staatiline tüübikontroll
  - Kompileerimise ajal
  - Leiab kohe vead, programm kiire
- Dünaamiline tüübikontroll
  - Programmi töö ajal

Tugev tüübikontroll

- Tüübid seotud muutujate nimedega
- Kontroll kompileerimise ajal
- Tüübiteisendused keelatud või eksplitsiitsed (eri arvamused)
- Tüübikontrollist mööda hiilida ei saa
- Keeruline, (compound) tüübisüsteem
- Andmeobjekti tüüp ei muutu elu ajal

Nõrk tüübikontroll

- Tüübid seotud muut. Väärtustega
- Tüübikontroll programmi töö ajal või puudub üldse (eri arvamused)
- Tüübiteisendused lubatud või implitsiitsed (eri arvamused)
- Saab kõrvale hiilida (cast & muu)
- Vaid mõned skalaarsed andmetüübid
- Muutuja andmetüüp või muutuda

Tüübikontroll

- Ekvivalents
  - Kahte tüüpi kaks väärtust võrdsed?
- Kompatiiblus
  - Millal saab x tüüpi väärtust kasutada y tüüpi väärtust eeldavas kontekstis?
- Mõju (*type inference*)
  - Mis on sellist tüüpi argumentidega avaldise (väärtuse) tüüp?

Tüübiteisendused

- Mis on "X+I" tüüp, kui
  - X on reaalarvu tüüpi
  - I on täisarvu tüüpi
  - Reaalarvude ja täisarvude jaoks on masinkoodis eraldi (aritmeetika) käsud
- Ühe liidetava tüüp vajab muutmist
  - Teisendused lähevad vahekoodi
  - Implitsiitne teisendus (*coercion*)
- Keel lihtsam aga nõrgem

Tüüpide mõju

- Avaldise või muutuja tüübi tuvastamine minimaalse infoga
- Olgu meil selline funktsioon function Foo(a,b)=a+b
- Kompilaator teab, et “+” tüüp on int ? int ? int
- Ehk + võtab kaks täisarvu ning tagastab täisarvu. Seega Foo samuti

Lihtne tüübikontrollija (1/6)

- Lihtne keel, igal identifikaatoril on oma tüüp. Grammatika:

P → D:E

D → D:D | id:T

T → char|int|array[num] of T|^T

E → literal|num|id|E mod E|E[E]|E^

Lihtne tüübikontrollija (2/6)

D? id:T	addtype(id.e,T.type)
T? char	T.type := char
T? int	T.type := int
T? ^T_1	T.type := ptr(T_1.type)
T? arr[n]of T_1	T.type := arr(l.n.v,T_1.type)

Kõik atribuudid on sünteesitud

Kuna P ? D:E siis tüübid on olemas enne E kontr

Lihtne tüübikontrollija (3/6)

E? literal	E.type:=char
E? num	E.type:=int
E? id	E.type:=lookup(id.e)
E? E_1 mod E_2	E.type:=IF E_1.type=int AND E_2.type=int THEN int ELSE typeError
E? $E_1[E_2]$	E.type:=IF E_2.type=int AND E_1.type=array(s,t) THEN t ELSE typeError
E? E_1^	E.type:=IF E_1.type=ptr(t) THEN t ELSE typeError

Lihtne tüübikontrollija (4/6)

Natuke üldistust ja implitsiitset teisendamist

E? num.num	E.type:=real
E? E_1 op E_2	

E_1.type	E_2.type	E.type
int	int	int
int	real	real
real	int	real
real	real	real
muu	muu	typeError

Lihtne tüübikontrollija (5/6)

S? id? E	S.type:=IF id.type=E.type THEN void ELSE typeError
S? if E then S_1	S.type:=IF E.type=boolean THEN S_1.type ELSE typeError
S? while E_1 do S_1	S.type:=IF E.type=boolean THEN S_1.type ELSE typeError
S? S_1:S_2	S.type:=IF S_1.type=void AND S_2.type=void THEN void ELSE typeError

Lihtne tüübikontrollija (6/6)

T? T "? " T	deklaratsioon
E? E (E)	aplikatsioon

T? T_1 "? " T_2	T.type:=(T_1.type ? T_2.type)
E? E_1 (E_2)	E.type:=IF E2.type=s AND E1.type=s? t THEN t ELSE typeError

Tüüpide ekvivalents

- Strukturaalne ekvivalents
  - (*structural equivalence*)
- Nimeline ekvivalents
  - (*name equivalence*)
- Deklaratiivne ekvivalents
  - (*declarative equivalence*)

Strukturaalne ekvivalents

```
function sequiv(s,t): boolean;
begin
    if (s ja t on samad baastüübid) then
        return true;
    elsif (s=array(s1,s2) and t=array(t1,t2)) then
        return sequiv(s1,t1) and sequiv(s2,t2);
    elsif s=s1 x s2 and t=t1 x t2 then
        return sequiv(s1,t1) and sequiv(s2,t2);
    elsif s=ptr(s1) and t=ptr(t1) then
        return sequiv(s1,t2);
    elsif s=s1? s2 and t=t1 ?  t2 then
        return sequiv(s1,t1) and sequiv(s2,t2);
    else
        return false;
end;
```

Nimeline ekvivalents

- Kaks tüübiavaldist on ekvivalentsed kui nad on sarnased
- Iga nimi esitab eeldatavasti erinevat andmetüüpi
- type link1 = ^A ? type link2 = ^A
- Tugevam kui strukturaalne eq sest nimi võib viidata vaid ühele tüübile

Deklaratiivne ekvivalents

- Mis saab siis kui tüüp ei ole nimi?

```
type link = ^A
var a1: ^A // implitsiitne tüüp type1
var a2,a3: ^A // implitsiitne tüüp type2
var a4: link
var a5: link
```

- Näiteks C kasutab str eq kõikide tüüpide jaoks peale kirjete. Kirjetes on kirje nimi osaks kirje tüübist.

Head ja vead

- Staatiline
  - Vead leitakse varakult
  - Programmi täitmine kiire
  - Liiga range (kasutute kohtade kontroll)
- Dünaamiline
  - Paindlikum
  - Ruumi ja aja kulu
  - Mittetäidetavaid programmi osi ei kontrollita (kiire luua, raskem siluda)

Tüüpida või siis mitte?

- Koodi täitmise kiirus
- Pisimahulise arenduse efektiivsus
- Kompileerimise efektiivsus
- Suuremahulise arenduse efektiivsus
- Keele võimaluste ulatus

Ehk siin kõik räägib tüüpimise kasuks

Täiendavaks lugemiseks...

- Type Systems, Luca Cardelli
  http://citeseer.ist.psu.edu/cardelli97type.html
- Üks tüübinduse Wiki (sealt hargneb)
  http://c2.com/cgi/wiki?TypeChecking
- etc