Tyyppikvanttorit ja alityypitys

Harri Mansikka

Helsinki 30.3.2003

Tyyppiteoria ja ohjelmointikielet -seminaari
esitelmä 3.4.2003

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Aluksi

Nyt kun sekä tyyppikvanttorit että alityypitys ovat tuttuja, herää luonnollisesti kysymys, voidaanko ne yhdistää. Ilmeinen tapa tehdä se olisi yksinkertaisesti yhdistää järjestelmä F ja $\lambda_{<:}$, ja näin voitaisiinkin tehdä. Toisaalta voidaan mennä askeleen verran pidemmällekin, ja sekoittaa näiden ominaisuuksia toisiinsa. Seuraava Benjamin C. Piercen kirjan *Types and Programming Languages* [Pie02] luvun 26 kanssa pikälti käsi kädessä kulkeva esitys kertoo yhdestä (tai puolestatoista) tavasta tehdä niin.

2 Miksi sekoittaa?

Oletetaan tässä esimerkissä, että käytössä on järjestelmien F ja $\lambda_{<:}$ unioni. Määritellään identiteettifunktio `f` tietueelle, jolla on kenttänä `a` luonnollinen luku

```
f = λx:{a:Nat}. x;
```

```
▶ f : {a:Nat} → {a:Nat}
```

sekä kyseisenkaltainen tietue `ra`

```
ra = {a=0};
```

ja nyt voidaan soveltaa

```
f ra;
```

```
▶ {a=0} : {a:Nat}
```

Kun määritellään suurempi tietue `rab`, jossa on toinenkin kenttä `b`

```
rab = {a=0, b=true};
```

niin $\lambda_{<:}$:n sisällyttämissäännön (*rule of subsumption*) nojalla siihenkin voidaan soveltaa funktiota `f`

```
f rab;
```

```
▶ {a=0, b=true} : {a:Nat}
```

mutta tällöin tarkempi tyyppi menetään — identiteettifunktio vei mahdollisuuden koskea kenttään `b`, eikä tätä voi hyväksyä!

Yritetään ratkaista ongelma käyttämällä järjestelmän F tarjoamaa polymorfismia.

```
fpoly = λX. λx:X. x;
▶ fpoly : ∀X. X → X
```

Tämä toimii nyt halutulla tavalla

```
fpoly [{a:Nat, b:Bool}] rab;
▶ {a=0, b=true} : {a:Nat, b:Bool}
```

Nyt kuitenkin menetetään hyödyllistä informaatiota. Tätä valaisee esimerkki, jossa funktion halutaan palauttavan jotain numeerisesta kentästä riippuvaa. Määritellään funktio `f2`, joka palauttaa parina parametrinsa sekä sen numeerisen kentän sisällön seuraajan

```
f2 = λx:{a:Nat}. {orig=x, asucc=succ(x.a)};
▶ f2 : {a:Nat} → {orig:{a:Nat}, asucc:Nat}
```

ja kuten edellä, sen soveltaminen tietueeseen `rab` hukkaa kentän `b`

```
f2 rab;
▶ {orig={a=0,b=true}, asucc=1} : {orig:{a:Nat], asucc:Nat}
```

Tällä kertaa polymorfismi ei sellaisenaan ole ratkaisu. Tyyppikvantifiointi `x`:n tyypin yli mahdollistaisi tyyppirikkomuksen, jossa parametriksi tarjotaan jotain muuta kuin kentän `a` sisältävää tietuetta. Esimerkkikääntäjä ([Pie02]) huomaa tämän

```
f2poly = λX. λx:X. {orig=x, asucc=succ(x.a)};
▶ Error: Expected record type
```

Jotta voitaisiin vaatia oikeanlaista parametria, on kieltä rikastettava sidottavan tyyppimuuttujan alityypitysrajoitteella. Saadaan järjestelmä $F_{<:}$, jonka erikoisominaisuus on *rajoitettu kvantifiointi*. Nyt esimerkki voidaan saattaa halutulla tavalla toimivaan muotoon

```
f2poly = λX<:{a:Nat}. λx:X. {orig=x, asucc=succ(x.a)};
▶ f2poly : ∀X<:{a:Nat}. X → {orig:X, asucc:Nat}
```

3 Määritellään uutta

Järjestelmä $F_{<:}$ saadaan yhdistämällä järjestelmä F ja $\lambda_{<:}$:n alityyppirelaatio sekä lisäämällä tyyppimuuttujien universaalikvantifiointeihin alityypitysrajoitteet. Liitteessä A on saatava järjestelmä kokonaisuudessaan.

3.1 Rajoittamaton kvantifiointi

Vähintäänkin esitysteknisistä syistä halutaan edelleen käyttää myös rajoittamatonta kvantifiointia. Se määritellään ilmeisellä tavalla kaikkien tyyppien ylityypillä `Top` rajoittamisena:

$$\forall \mathtt{X}.\mathtt{T} =^{def} \forall \mathtt{X}<:\mathtt{Top}.\mathtt{T}$$

3.2 Alityypitys

$F_{<:}$:ssä tyyppimuuttujaan liittyy aina rajoite, joista on pidettävä kirjaa alityypityksen ja tyypintarkistuksen aikana. Tämä tehdään lisäämällä kontekstissa esiintyvään tyyppimuuttujan sidontaan alityypitysrajoite. Näillä rajoitteilla voidaan sitten perustella alityypityksessä päättelyt muotoa "tyyppimuuttuja `X` on tyypin `T` alityyppi" käyttämällä uutta sääntöä (S-TVar)

$$\frac{\mathtt{X} <: \mathtt{T} \in \Gamma}{\Gamma \vdash \mathtt{X} <: \mathtt{T}} \ (\mathrm{S-TVar})$$

Tämän säännön lisäys tekee alityypityksestä kolmipaikkaisen relaation — alityypitysväittämät ovat nyt muotoa $\Gamma \vdash \mathtt{S} <: \mathtt{T}$, joka luetaan "`S` on `T`:n alityyppi oletuksilla Γ". Vastaavasti kaikkiin alityypityssääntöihin täytyy lisätä konteksti.

Kvantifioitujen tyyppien vertailemista varten tarvitaan uusi sääntö (S-All)

$$\frac{\Gamma, \mathtt{X} <: \mathtt{U}_1 \vdash \mathtt{S}_2 <: \mathtt{T}_2}{\Gamma \vdash \forall \mathtt{X} <: \mathtt{U}_1.\mathtt{S}_2 <: \forall \mathtt{X} <: \mathtt{U}_1.\mathtt{T}_2} \ (\mathrm{S-All})$$

Esimerkki sääntöjen käytöstä: todistetaan, että $\mathtt{B} \to \mathtt{Y} <: \mathtt{X} \to \mathtt{B}$ kontekstissa $\Gamma = \mathtt{B} <: \mathtt{Top}, \mathtt{X} <: \mathtt{B}, \mathtt{Y} <: \mathtt{X}$.

$$\dfrac{\dfrac{\mathtt{X} <: \mathtt{B} \in \Gamma}{\Gamma \vdash \mathtt{X} <: \mathtt{B}} \ (\mathrm{S-TVar}) \quad \dfrac{\dfrac{\mathtt{Y} <: \mathtt{X} \in \Gamma}{\Gamma \vdash \mathtt{Y} <: \mathtt{X}} \ (\mathrm{S-TVar}) \quad \dfrac{\mathtt{X} <: \mathtt{B} \in \Gamma}{\Gamma \vdash \mathtt{X} <: \mathtt{B}} \ (\mathrm{S-TVar})}{\Gamma \vdash \mathtt{Y} <: \mathtt{B}} \ (\mathrm{S-Trans})}{\Gamma \vdash \mathtt{B} \to \mathtt{Y} <: \mathtt{X} \to \mathtt{B}} \ (\mathrm{S-Arrow})$$

3.3 Näkyvyys

Tyyppimuuttujien näkyvyysalueet ovat huomioimisen arvoinen yksityiskohta. Tyypitysväittämässä $\Gamma \vdash \mathtt{t} : \mathtt{T}$ termin `t` ja tyyppimuuttujan `T` vapaat tyyppimuuttujat on määritelty kontekstissa Γ. Nyt kun kielioppiin on lisätty kontekstin tyyppimuuttujan sidontaan alityypitysrajoite, herää kysymys, missä määritellään kontekstissa Γ esiintyvät vapaat tyyppimuuttujat. Esimerkiksi, mitkä seuraavista konteksteista ovat näkyvyydeltään järkeviä?

Γ_1 = X<:Top,y:X→Nat
Γ_2 = y:X→Nat,X<:Top
Γ_3 = X<:{a:Nat,b:X}

Γ_1 on järkevä: tällaisen kontekstin tyypintarkistuksessaan synnyttävä termi olisi muotoa λX<:Top. λy:X→Nat. t, jossa y:n tyypissä X:n on sitonut ympäröivä λ. Vastaavasti Γ_2 on virheellinen: sen synnyttänyt termi voisi olla esimerkiksi muotoa λy:X→Nat. λX<:Top. t, jolla X:n tarkoitettu näkyvyysalue olisi epäselvä. Γ_3 on moniulotteisempi kysymys: voidaan määrittää, että X:n näkyvyysalue kattaa sen omassa määrityksessä olevan rajoiteosan, jolloin esim. termissä λX<:{a:Nat,b:X}. t jälkimmäinen X:n esiintymä olisi sidottu — tätä kutsutaan *F-rajoitetuksi kvantifioinniksi*, joka on teoreettisesti huomattavasti mutkikkaampi tapaus, ja hyödyllinen vain rekursiivisten tyyppien parissa työskenneltäessä, eikä siten tarpeen tässä esityksessä. Jatkossa hyväksytään vain tyypitykset, joissa kontekstissa esiintyvän tyypin vapaat tyyppimuuttujat on jo sidottu ennen sen esiintymää eli sen vasemmalla puolella kontekstissa.

3.4 Tyypitys

Myös kvantifioitujen tyyppien tyypityssääntöjä täytyy hienosäätää. Rajoitetun kvantifioinnin tuontisäännössä (T-TAbs) saadaan abstraktiosta sen rungon tyypintarkistuksessa rajoite kontekstiin

$$\frac{\Gamma, X <: T \vdash t_2 : T_2}{\Gamma \vdash \lambda X <: T.t_2 : \forall X <: T.T_2} \ (T-TABS)$$

kun taas eliminointisäännössä (T-TApp) tarkistetaan, että tyyppiparametri sopii rajoitteeseen

$$\frac{\Gamma \vdash t_1 : \forall X <: T_{11}.T_{12} \quad \Gamma \vdash T_2 <: T_{11}}{\Gamma \vdash t_1\,[T_2] : [X \mapsto T_2]T_{12}} \ (T-TAPP)$$

4 Käyttöesimerkkejä

Havainnollistetaan $F_{<:}$:n ominaisuuksia muutamalla pienellä esimerkillä.

4.1 Parit

On jo nähty, kuinka pari luonnollisia lukuja koodataan järjestelmässä F. Nyt yleistetään koodaus mielivaltaisille tyypeille. Tyypin

```
Pair T1 T2 = ∀X. (T1 →T2→X) → X;
```

alkiot edustavat tyyppien $\mathtt{T}_1$ ja $\mathtt{T}_2$ alkioiden pareja. Konstruktori `pair` ja destruktorit `fst` sekä `snd` määritellään seuraavasti:

```
pair = λX. λY. λx:X. λy:Y. (λR. λp:X→Y→R. p x y) as Pair X Y;
▶ pair : ∀X. ∀Y. X → Y → Pair X Y
fst = λX. λY. λp: Pair X Y. p [X] (λx:X. λy:Y. x);
▶ fst : ∀X. ∀Y. Pair X Y → X
snd = λX. λY. λp: Pair X Y. p [Y] (λx:X. λy:Y. y);
▶ snd : ∀X. ∀Y. Pair X Y → Y
```

Koodaus käy sellaisenaan myös $\mathrm{F}_{<:}$:ssä. Merkillepantavaa on se, että luonnollinen alityypityssääntö pareille

$$\frac{\Gamma \vdash \mathtt{S}_1 <: \mathtt{T}_1 \quad \Gamma \vdash \mathtt{S}_2 <: \mathtt{T}_2}{\Gamma \vdash \mathtt{Pair}\,\mathtt{S}_1\,\mathtt{S}_2 <: \mathtt{Pair}\,\mathtt{T}_1\,\mathtt{T}_2}$$

voidaan todistaa suoraan koodauksen ominaisuuksista (kaikki käytetyt säännöt ovat (S-...))

$$\dfrac{\dfrac{\Gamma,\mathtt{X} \vdash \mathtt{S}_1 <: \mathtt{T}_1 \quad \dfrac{\dfrac{\Gamma,\mathtt{X} \vdash \mathtt{S}_2 <: \mathtt{T}_2 \quad \overline{\Gamma,\mathtt{X} \vdash \mathtt{X} <: \mathtt{X}}\ (\textsc{Refl})}{\Gamma,\mathtt{X} \vdash \mathtt{T}_2 \to \mathtt{X} <: \mathtt{S}_2 \to \mathtt{X}}\ (\textsc{Arrow})}{\Gamma,\mathtt{X} \vdash \mathtt{T}_1 \to \mathtt{T}_2 \to \mathtt{X} <: \mathtt{S}_1 \to \mathtt{S}_2 \to \mathtt{X}}\ (\textsc{Arrow}) \quad \dfrac{}{\Gamma,\mathtt{X} \vdash \mathtt{X} <: \mathtt{X}}\ (\textsc{Refl})}{\Gamma,\mathtt{X} \vdash (\mathtt{S}_1 \to \mathtt{S}_2 \to \mathtt{X}) \to \mathtt{X} <: (\mathtt{T}_1 \to \mathtt{T}_2 \to \mathtt{X}) \to \mathtt{X}}\ (\textsc{Arrow})}{\Gamma \vdash \forall \mathtt{X}.(\mathtt{S}_1 \to \mathtt{S}_2 \to \mathtt{X}) \to \mathtt{X} <: \forall \mathtt{X}.(\mathtt{T}_1 \to \mathtt{T}_2 \to \mathtt{X}) \to \mathtt{X}}\ (\textsc{All})}{\Gamma \vdash \mathtt{Pair}\,\mathtt{S}_1\,\mathtt{S}_2 <: \mathtt{Pair}\,\mathtt{T}_1\,\mathtt{T}_2}\ (\mathrm{def})$$

4.2 Tietueet

Huomattava tulos on, että $\mathrm{F}_{<:}$ mahdollistaa tietuetyyppien ja tietueiden koodauksen suoraan — ilman tarvetta lisätä kieleen primitiiviä `{}`.

Määritellään ensin *joustavat monikot* (nimitys "joustavat", koska ovat jatkettavissa oikealta alityypityksessä). Kaikilla $n \geq 0$ ja tyypeillä $\mathtt{T}_1$ ... $\mathtt{T}_n$, olkoon

$$\{\mathtt{T}_i{}^{i\in 1..n}\} =^{def} \mathtt{Pair}\ \mathtt{T}_1\ (\mathtt{Pair}\ \mathtt{T}_2\ \ldots\ (\mathtt{Pair}\ \mathtt{T}_n\ \mathtt{Top})\ \ldots).$$

Erityisesti $\mathtt{\{\}} =^{def} \mathtt{Top}$. Termeistä $\mathtt{t}_1 \ldots \mathtt{t}_n$ saadaan nyt monikko asettamalla

$$\{\mathtt{t}_i{}^{i\in 1..n}\} =^{def} \mathtt{pair}\ \mathtt{t}_1\ (\mathtt{pair}\ \mathtt{t}_2\ \ldots\ (\mathtt{pair}\ \mathtt{t}_n\ \mathtt{top})\ \ldots),$$

josta jätettiin tässä tyyppiparametrit kirjoittamatta selkeyden (ja ilmaisun pituuden) nimissä. Termiksi `top` käy mikä tahansa tyypin `Top` edustaja eli mikä

tahansa suljettu ja hyvinmääritelty termi. Projektiot saadaan nyt (taas tyyppiparametrit pudottaen) seuraavasti:

$$\mathtt{t.j} =^{def} \mathtt{fst}\,(\underbrace{\mathtt{snd}\,(\mathtt{snd}\,\ldots\,(\mathtt{snd}}_{\mathtt{j}-1\text{ kpl}}\,\mathtt{t})\ldots))$$

Näistä saadaan suoraan joustaville monikoille alityypitys- ja tyypityssäännöt:

$$\frac{\Gamma \vdash \mathtt{S}_i <: \mathtt{T}_i \quad \text{kaikilla } i \in 1..n}{\Gamma \vdash \{\mathtt{S}_i{}^{i\in 1..n+k}\} <: \{\mathtt{T}_i{}^{i\in 1..n}\}}$$

$$\frac{\Gamma \vdash \mathtt{t}_i : \mathtt{T}_i \quad \text{kaikilla } i \in 1..n}{\Gamma \vdash \{\mathtt{t}_i{}^{i\in 1..n}\} : \{\mathtt{T}_i{}^{i\in 1..n}\}}$$

$$\frac{\Gamma \vdash \mathtt{t} : \{\mathtt{T}_i{}^{i\in 1..n}\}}{\Gamma \vdash \mathtt{t.j} : \mathtt{T_j}}$$

Olkoon sitten $\mathcal{L}$ numeroituva ja bijektion *label-with-index* : $\mathbb{N} \to \mathcal{L}$ täydellisesti järjestämä joukko nimiöitä. Nyt voimme määritellä tietueet seuraavasti:

olkoon $L \subset \mathcal{L}$ äärellinen, ja olkoon $\mathtt{S}_l$ tyyppi kaikilla $l \in L$. Olkoon

$$m = max\{\, i \mid \exists l \in L : label - with - index(i) = l\}$$

ja olkoon

$$\hat{\mathtt{S}}_i = \begin{cases} \mathtt{S}_l & \text{jos } label - with - index(i) = l \in L \\ \mathtt{Top} & \text{jos } label - with - index(i) \notin L \end{cases}$$

Tietuetyyppi $\{l : \mathtt{S}_l{}^{l\in L}\}$ määritellään nyt joustavana monikkona $\{\hat{S}_i{}^{i\in 1..m}\}$. Vastaavasti, kun $\mathtt{t}_l$ on termi kaikilla $l \in L$, niin

$$\hat{\mathtt{t}}_i = \begin{cases} \mathtt{t}_l & \text{jos } label - with - index(i) = l \in L \\ \mathtt{top} & \text{jos } label - with - index(i) \notin L \end{cases}$$

Tietuearvo $\{l = \mathtt{t}_l{}^{l\in L}\}$ on $\{\hat{t}_i{}^{i\in 1..m}\}$. Tietueprojektio `t.l` on yksinkertaisesti monikkoprojektio `t.i`, missä *label-with-index*(i)=l.

Tämä koodaus sopii yhteen tuttujen tietuiden tyypitys- ja alityypityssääntöjen (S-RcdWidth, S-RcdDepth, S-RcdPerm, T-Rcd ja T-Proj) kanssa. Sen merkitys on kuitenkin enemmänkin teoreettinen — käytännössä sen vaatimus nimiöiden globaalista numeroinnista on vakava este: modulaarista ohjelmaa käännettäessä ei vielä voida numeroida nimiöitä modulikohtaisesti, vaan se täytyy tehdä kaikille kerralla (siis linkitysvaiheessa).

4.3 Churchin numeraalit ja alityypitys

Lopuksi tarkastellaan rajoitetun kvantifioinnin lisäämistä Churchin numeraaleihin. Järjestelmässä F Churchin numeraalit olivat tyyppiä

```
CNat = ∀X. (X→X) → X → X;
```

Tyyppi siis ottaa parametrina ensinnäkin tyypin, jonka alkioiksi luvut koodataan, sitten ko. tyypin funktion sekä tyypin alkion nollaa edustamaan, jonka jälkeen luku n koodataan soveltamalla saatua funktiota nollan edustajaan n kertaa.

Rajoitettu kvantifiointi mahdollistaa nyt yleisemmän ratkaisun:

```
SNat = ∀X. ∀S<:X. ∀Z<:X. (X→S) → Z → X;
```

Uusi tyyppi ottaa taas tyypin, jolle koodataan, sekä sille kaksi alityyppiä, joista toiseen nollan edustajan on tarkoitus kuulua, ja toiseen seuraaja-alkioiden.

Tätä hyödynnetään seuraavasti — määritellään hieman erilainen tyyppi

```
SZero = ∀X. ∀S<:X. ∀Z<:X. (X→S) → Z → Z;
```

Muodoltaan tämä ei eroa tyypistä `SNat` paljoakaan, mutta käyttäytyminen on huomattavasti rajoitetumpaa. Palautettavan arvon on oltava tyyppiä `Z`, joka on mahdollista vain palauttamalla parametri `z`. Siis arvo

```
szero = λX. λS<:X. λZ<:X. λs:X→S. λz:Z. z;
```

```
▶ szero : SZero
```

on tyypin `SZero` *ainoa* edustaja (tai tarkalleen ottaen kaikki tyypin `SZero` edustajat ovat käyttäytymiseltään `szero`:n kanssa identtisiä). Koska `SZero <: SNat`, pätee myös `szero : SNat`.

Toisaalta, samansuuntaisella tyypillä

```
SPos = ∀X. ∀S<:X. ∀Z<:X. (X→S) → Z → S;
```

on enemmänkin edustajia, esimerkiksi

```
sone   = λX. λS<:X. λZ<:X. λs:X→S. λz:Z. s z;
stwo   = λX. λS<:X. λZ<:X. λs:X→S. λz:Z. s (s z);
sthree = λX. λS<:X. λZ<:X. λs:X→S. λz:Z. s (s (s z));
```

jne. (Itse asiassa, kaikki `SNat`:n edustajat *paitsi* `szero`.)

Näin voidaan valjastaa tyyppijärjestelmä esimerkiksi tarkistamaan, että seuraajafunktio palauttaa aina positiivisen luvun:

```
ssucc = λn:SNat. λX. λS<:X. λZ<:X. λs:X→S. λz:Z.
          s (n [X] [S] [Z] s z);
▶ ssucc : SNat → SPos
```

Samankaltaisia tarkastuksia voidaan koodata muihinkin laskutoimituksiin. Esimerkiksi

```
spluspp = λn:SPos. λm:SPos. λX. λS<:X. λZ<:X. λs:X→S. λz:Z.
            n [X] [S] [S] s (m [X] [S] [Z] s z);
▶ spluspp : SPos → SPos → SPos
```

ja edelleen `spluszz : SZero → SZero → SZero` jne. Tietenkään ei ole tarkoitus määritellä koko liutaa funktioita, joista olisi valittava aina jokin parametrien tyyppien mukaan, vaan päämääränä on yksi ainoa *kuormitettu* funktio `plus`, jonka tyyppi huomioi kaikki mahdollisuudet — siis:

```
plus :   SZero→SZero→SZero
       ∧ SNat→SPos→SPos
       ∧ SPos→SNat→SNat
       ∧ SNat→SNat→SNat
```

missä `t : S∧T` tarkoittaa, että "termi `t` on sekä tyyppiä `S` että tyyppiä `T`".

5 Oikeellisuusominaisuudet

Tyypinsäilymis- ja etenemisominaisuuksille $F_{<:}$:ssa (apulauseineen) on suoraviivaiset induktiotodistukset. Ne ovat mielenkiintoisimmilta osiltaan luettavissa kirjasta [Pie02].

Lemma 1 (Permutaatio) Oletetaan, että Γ on hyvinmuodostettu konteksti, ja että Δ on Γ:n permutaatio, joka säilyttää tyyppimuuttujien näkyvyyden niiden instansseihin. Tällöin

1. jos $\Gamma \vdash$ `t : T`, niin $\Delta \vdash$ `t : T` ja
2. jos $\Gamma \vdash$ `S <: T`, niin $\Delta \vdash$ `S <: T`. □

Lemma 2 (Heikennys)

1. Jos $\Gamma \vdash \mathtt{t} : \mathtt{T}$ ja Γ,$\mathtt{x:U}$ on hyvinmuodostettu, niin Γ,$\mathtt{x:U} \vdash \mathtt{t} : \mathtt{T}$.
2. Jos $\Gamma \vdash \mathtt{t} : \mathtt{T}$ ja Γ,$\mathtt{X<:U}$ on hyvinmuodostettu, niin Γ,$\mathtt{X<:U} \vdash \mathtt{t} : \mathtt{T}$.
3. Jos $\Gamma \vdash \mathtt{S} <: \mathtt{T}$ ja Γ,$\mathtt{x:U}$ on hyvinmuodostettu, niin Γ,$\mathtt{x:U} \vdash \mathtt{S} <: \mathtt{T}$.
4. Jos $\Gamma \vdash \mathtt{S} <: \mathtt{T}$ ja Γ,$\mathtt{X<:U}$ on hyvinmuodostettu, niin Γ,$\mathtt{X<:U} \vdash \mathtt{S} <: \mathtt{T}$. □

Lemma 3 (Vahvennus termimuuttujilla alityypitystodistuksissa)

Jos Γ,$\mathtt{x:T}$,$\Delta \vdash \mathtt{S} <: \mathtt{T}$, niin Γ,$\Delta \vdash \mathtt{S} <: \mathtt{T}$. □

Lemma 4 (Kavennus)

1. Jos Γ,$\mathtt{X<:Q}$,$\Delta \vdash \mathtt{S} <: \mathtt{T}$ ja $\Gamma \vdash \mathtt{P} <: \mathtt{Q}$, niin Γ,$\mathtt{X<:P}$,$\Delta \vdash \mathtt{S} <: \mathtt{T}$.
2. Jos Γ,$\mathtt{X<:Q}$,$\Delta \vdash \mathtt{t} : \mathtt{T}$ ja $\Gamma \vdash \mathtt{P} <: \mathtt{Q}$, niin Γ,$\mathtt{X<:P}$,$\Delta \vdash \mathtt{t} : \mathtt{T}$. □

Lemma 5 (Termin sijoittaminen säilyttää tyypityksen)

Jos Γ,$\mathtt{x:Q}$,$\Delta \vdash \mathtt{t} : \mathtt{T}$ ja $\Gamma \vdash \mathtt{q} : \mathtt{Q}$, niin Γ,$\Delta \vdash [\mathtt{x} \mapsto \mathtt{q}]\mathtt{t} : \mathtt{T}$. □

Koska päättelyissä voidaan sijoittaa tyyppejä tyyppimuuttujiin, tarvitsemme myös apulauseen koskien sijoituksen käyttäytymistä tyypityksessä.

Määritelmä 6 Merkitään $[\mathtt{X} \mapsto \mathtt{S}]\Gamma$ kontekstia, joka saadaan sijoittamalla $\mathtt{S}$ kaikkiin $\mathtt{X}$:n esiintymiin Γ:n sidontojen *sijoitettavissa* (ts. oikeissa puolissa).

Lemma 7 (Tyypin sijoittaminen säilyttää alityypityksen)

Jos Γ,$\mathtt{X<:Q}$,$\Delta \vdash \mathtt{S} <: \mathtt{T}$ ja $\Gamma \vdash \mathtt{P} <: \mathtt{Q}$, niin Γ,$[\mathtt{X} \mapsto \mathtt{P}]\Delta \vdash [\mathtt{X} \mapsto \mathtt{P}]\mathtt{S} <: [\mathtt{X} \mapsto \mathtt{P}]\mathtt{T}$. □

Huomioitavaa on, että näkyvyyssääntöjen nojalla riittää sijoittaa X:aan vain sen sidontaa seuraavassa osassa kontekstia.

Lemma 8 (Tyypin sijoittaminen säilyttää tyypityksen)

Jos Γ,$\mathtt{X<:Q}$,$\Delta \vdash \mathtt{t} : \mathtt{T}$ ja $\Gamma \vdash \mathtt{P} <: \mathtt{Q}$, niin Γ,$[\mathtt{X} \mapsto \mathtt{P}]\Delta \vdash [\mathtt{X} \mapsto \mathtt{P}]\mathtt{t} : [\mathtt{X} \mapsto \mathtt{P}]\mathtt{T}$. □

Seuraavaksi käsitellään eräs alityypityksen rakenteellinen ominaisuus.

Lemma 9 (Alityyppirelaation kääntäminen)

1. Jos $\Gamma \vdash \mathtt{S} <: \mathtt{X}$, niin $\mathtt{S}$ on tyyppimuuttuja.
2. Jos $\Gamma \vdash \mathtt{S} <: \mathtt{T}_1 \to \mathtt{T}_2$, niin $\mathtt{S}$ on joko tyyppimuuttuja, tai $\mathtt{S}$ on muotoa $\mathtt{S}_1 \to \mathtt{S}_2$ ja $\Gamma \vdash \mathtt{T}_1 <: \mathtt{S}_1$ sekä $\Gamma \vdash \mathtt{S}_2 <: \mathtt{T}_2$.
3. Jos $\Gamma \vdash \mathtt{S} <: \forall \mathtt{X}<:\mathtt{U}_1.\mathtt{T}_2$, niin $\mathtt{S}$ on joko tyyppimuuttuja, tai $\mathtt{S}$ on muotoa $\forall \mathtt{X}<:\mathtt{U}_1.\mathtt{S}_2$ ja $\Gamma,\mathtt{X}<:\mathtt{U}_1 \vdash \mathtt{S}_2 <: \mathtt{T}_2$. □

Sitten on vuorossa tyypityksen tärkeä rakenteellinen ominaisuus.

Lemma 10

1. Jos $\Gamma \vdash \lambda \mathtt{x}:\mathtt{S}_1.\mathtt{s}_2 : \mathtt{T}$ ja $\Gamma \vdash \mathtt{T} <: \mathtt{U}_1 \to \mathtt{U}_2$, niin $\Gamma \vdash \mathtt{U}_1 <: \mathtt{S}_1$ ja on olemassa sellainen $\mathtt{S}_2$ että $\Gamma,\mathtt{x}:\mathtt{S}_1 \vdash \mathtt{s}_2 : \mathtt{S}_2$ sekä $\Gamma \vdash \mathtt{S}_2 <: \mathtt{U}_2$.
2. Jos $\Gamma \vdash \lambda \mathtt{X}<:\mathtt{S}_1.\mathtt{s}_2 : \mathtt{T}$ ja $\Gamma \vdash \mathtt{T} <: \forall \mathtt{X}<:\mathtt{U}_1.\mathtt{U}_2$, niin $\mathtt{U}_1 = \mathtt{S}_1$ ja on olemassa sellainen $\mathtt{S}_2$ että $\Gamma,\mathtt{X}<:\mathtt{S}_1 \vdash \mathtt{s}_2 : \mathtt{S}_2$ sekä $\Gamma,\mathtt{X}<\mathtt{S}_1 \vdash \mathtt{S}_2 <: \mathtt{U}_2$. □

Nyt ollaan valmiita tyypinsäilymislauseen todist(uksen sivuutt)amiseen.

Lause 11 (Tyypinsäilyminen) Jos $\Gamma \vdash \mathtt{t} : \mathtt{T}$ ja $\mathtt{t} \to \mathtt{t'}$, niin $\Gamma \vdash \mathtt{t'} : \mathtt{T}$.

Todistus: Induktiolla tyypityksen $\Gamma \vdash \mathtt{t} : \mathtt{T}$ johtamisen suhteen. □

$F_{<:}$:n etenemislause saadaan lisäämällä pientä hienosäätöä $\lambda_{<:}$:n vastaavaan. Ensin kirjataan apulauseeksi suljettujen funktio- ja kvanttorityyppien mahdolliset muodot.

Lemma 12 (Kanoniset muodot)

1. Jos $\mathtt{v}$ on tyypin $\mathtt{T}_1 \to \mathtt{T}_2$ suljettu arvo, niin $\mathtt{v}$ on muotoa $\lambda \mathtt{x}:\mathtt{T}_1.\mathtt{t}_2$.
2. Jos $\mathtt{v}$ on tyypin $\forall \mathtt{X}<:\mathtt{T}_1.\mathtt{T}_2$ suljettu arvo, niin $\mathtt{v}$ on muotoa $\lambda \mathtt{X}<:\mathtt{T}_1.\mathtt{t}_2$. □

Lopuksi todist(uksetta vilaut)etaan etenemislause:

Lause 13 (Eteneminen) Jos $\mathtt{t}$ on suljettu ja hyvintyypitetty $F_{<:}$:n termi, niin joko $\mathtt{t}$ on arvo tai on olemassa sellainen $\mathtt{t'}$ että $\mathtt{t} \to \mathtt{t'}$.

Todistus: Induktiolla tyyppien johtamisten suhteen. □

6 Täysi $F_{<:}$

Tähän asti käytetty kvantifioitujen tyyppien alityypityssääntö (S-All) vaatii, että verrattavien tyyppien alityyppitysrajoite on sama. Tästä vaatimuksesta voidaan tinkiä: funktiotyyppien vertailua vastaavasti (universaalityypin operationaalinen tulkintahan oli myös funktionaalinen: $\forall \mathtt{X}.\mathtt{T}$ kuvaa tyypin $\mathtt{S}$ tyypiksi $[\mathtt{X} \mapsto \mathtt{S}]\mathtt{T}$) vaaditaan, että ylityypin rajoite on alityypin rajoitteen alityyppi, jolloin saadaan huomattavasti ilmaisuvoimaisempi versio (S-AllF)

$$\frac{\Gamma \vdash \mathtt{T}_1 <: \mathtt{S}_1 \qquad \Gamma, \mathtt{X} <: \mathtt{T}_1 \vdash \mathtt{S}_2 <: \mathtt{T}_2}{\Gamma \vdash \forall \mathtt{X} <: \mathtt{S}_1.\mathtt{S}_2 <: \forall \mathtt{X} <: \mathtt{T}_1.\mathtt{T}_2} \ (\mathrm{S-AllF})$$

Kun korvataan sääntö (S-All) säännöllä (S-AllF), saadaan vahvempi järjestelmä, jota kutsutaan nimellä *täysi* $F_{<:}$ — tähänastisesta versiosta voidaan käyttää nimitystä *ydin*-$F_{<:}$. Kaikki nähdyt esimerkit toimivat kummallakin versiolla.

Täysi $F_{<:}$ käyttäytyy teorian tasolla yhtä kiltisti kuin ydin-$F_{<:}$. Esimerkiksi oikeellisuusominaisuuksien todistukset ovat yhtä helppoja. Käytännön algoritmeja mietittäessä vahvempi ilmaisuvoima kuitenkin vaatii veronsa. Osoittautuu esimerkiksi, että täydellä versiolla tyypintarkastus on ratkeamaton tehtävä, kun ydinversiolla se vielä onnistuu. (Rajoitetun kvantifioinnin metateoriaan voi perehtyä tarkemmin kirjan [Pie02] luvussa 28.)

7 Rajoitettu eksistentiaalityyppi

Myös eksistentiaalityypeille voidaan asettaa rajoitteita.

7.1 Määrittelyt

Rajoitetun eksistentiaalityypin syntaktinen muoto on $\{\exists \mathtt{X} <: \mathtt{T}, \mathtt{T}\}$. Tyypityssäännöt muuttuvat hieman:

$$\frac{\Gamma \vdash \mathtt{t}_2 : [\mathtt{X} \mapsto \mathtt{U}]\mathtt{T}_2 \quad \Gamma \vdash \mathtt{U} <: \mathtt{T}_1}{\Gamma \vdash \{*\mathtt{U}, \mathtt{t}_2\} \text{ as } \{\exists \mathtt{X} <: \mathtt{T}_1, \mathtt{T}_2\} : \{\exists \mathtt{X} <: \mathtt{T}_1, \mathtt{T}_2\}} \ (\mathrm{T-Pack})$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \{\exists \mathtt{X} <: \mathtt{T}_{11}, \mathtt{T}_{12}\} \quad \Gamma, \mathtt{X} <: \mathtt{T}_{11}, \mathtt{x} : \mathtt{T}_{12} \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \vdash \mathtt{let}\ \{\mathtt{X}, \mathtt{x}\} = \mathtt{t}_1\ \mathtt{in}\ \mathtt{t}_2 : \mathtt{T}_2} \ (\mathrm{T-Unpack})$$

Järjestelmälle ydin-$F_{<:}$ saadaan alityypityssääntö

$$\frac{\Gamma, \mathtt{X} <: \mathtt{U} \vdash \mathtt{S}_2 <: \mathtt{T}_2}{\Gamma \vdash \{\exists \mathtt{X} <: \mathtt{U}, \mathtt{S}_2\} <: \{\exists \mathtt{X} <: \mathtt{U}, \mathtt{T}_2\}} \ (\mathrm{S-Some})$$

tai järjestelmälle täysi $F_{<:}$ vastaava sääntö

$$\frac{\Gamma \vdash T_1 <: S_1 \quad \Gamma, X <: T_1 \vdash S_2 <: T_2}{\Gamma \vdash \{\exists X <: S_1, S_2\} <: \{\exists X <: S_2, T_2\}} \text{ (S} - \text{SomeF)}$$

7.2 Osittain abstraktit tietotyypit

Aiemmin on nähty, kuinka eksistentiaalityypeillä voidaan ohjelmoida abstrakteja tietotyyppejä. Rajoitetuilla eksistentiaalityypeillä puolestaan voidaan ohjelmoida *osittain abstrakteja tietotyyppejä*. Tällöin piilotetaan vain valittuja osia varsinaisesti toteutuksesta.

Esimerkki: laskurin abstrakti tietotyyppi on ennestään tuttu, mutta nyt lisätään sen tyyppimäärittelyyn rajoite `Counter<:Nat`.

```
counterPkg = {*Nat,
   {new = 1, get = λi:Nat. i, inc = λi:Nat. succ(i)}}
   as {∃Counter<:Nat,
       {new: Counter, get: Counter→Nat, inc: Counter→Counter}};
▶ counterPkg : {∃Counter<:Nat,
   {new: Counter, get: Counter→Nat, inc: Counter→Counter}};
```

Tätä laskuria voidaan käyttää tutulla tavalla:

```
let {Counter,c} = counterPkg in c.get (c.inc (c.inc c.new));
▶ 3 : Nat
```

Lisäksi `Counter`-arvoja voidaan käyttää suoraan lukuina:

```
let {Counter,c} = counterPkg in succ (succ (c.inc c.new));
▶ 4 : Nat
```

Kuitenkaan ei ole mahdollista käyttää lukuja `Counter`-arvoina:

```
let {Counter,c} = counterPkg in c.inc 3;
▶ Error: parameter type mismatch
```

Näin abstrahoidaan laskuri, jota on helpompi käyttää, mutta jonka luominen on edelleen hallinnassa.

Aivan vastaavasti voidaan oliokoodauksessa mahdollistaa sisäisen rakenteen osittainen näyttäminen ulkomaailmalle.

A Liite: Järjestelmä $\mathbf{F}_{<:}$

(Kirjan [Pie02] sivu 392 — käytännöllisin ratkaisu lienee se, että tuon mukanani kopioita sivusta kirjattomille.)

B Liite: Lähteet

[Pie02] Pierce, B. C. *Types and Programming Languages.* MIT Press, 2002.