Tyyppitoimitukset (kinds)

Otso Kivekäs

Helsinki 6. toukokuuta 2003

Tyyppiteoria-seminaarin esitys

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Johdanto

Tyypintoimitukset (engl. kinds) ovat yksinkertaisesti sanottuna tapa tuoda abstraktio- ja sijoitusoperaatiot tyyppien tasolle, eli formalisoida *tyyppioperaatiot* jotka ottavat argumentikseen tyypin ja palauttavat tyypin.

Operaatioabstraktio $\lambda X.T$ eroaa systeemin F tyyppiabstraktiosta $\lambda X.t$ siinä, että sen sovellus palauttaa tyypin eikä termiä.

Kapale 2 esittelee tyypitetyn lambda-kalkyylin johon on lisätty tyyppioperaatiot. Kappale 3.1 lisää tähän universalityypit jolloin saadaan systeemi $F_\omega$ja kappale 3.2 eksistentiaalityypit. Lopuksi kappale 4 esittelee tapoja rajoittaa systeemiä F_ω.

Tämä esitys perustuu Benjamin C. Piercen kirjan **Types and Programming Languages** [PIE02] lukuihin 29 ja 30. Todistukset sivuutetaan, koska ne ovat kohtuullisen monimutkaisia, eikä niille yksinkertaisesti riitä tilaa tai aikaa.

1.1 Sananen notaatiosta

Tässä esityksessä pienet kirjaimet (s, t, u, ...) tarkoittavat aina termejä ja isot kirjaimet (A, B, T, X, Z, ...) tyyppejä, paitsi K joka on aina tyypintoimitin. Kirjoitusvirheet ovat kuitenkin mahdollisia, koska kaavoja on melko paljon.

$\lambda_\rightarrow$ on yksinkertainen tyypitetty lambdakalkyyli ilman perustyyppejä, F on aiemmin esitelty systeemi F, $\lambda_\omega$on yksinkertainen tyypitetty lambdakalkyyli johon on lisätty tyypintoimitukset ja $F_\omega$on systeemin F ja λ_ω-kalkyylin yhdistelmä, jossa on sekä universaalikvanttori että tyypintoimitukset.

Terminologia saattaa olla ajoittain hieman horjuvaa johtuen termien ad hoc käännöksestä.

2 Tyyppitoimitukset lambdakalkyylissä

Tyypintoimittimet ovat tapa tuoda termien tasolla käytetyt abstraktion ja soveltamisen mekanismit tyyppien tasolle.

Aitojen tyyppien (so. tyyppien jotka voivat olla minkään termin tyyppinä, ks kuva 1) tyypintoimitin on $*$. Tyyppioperaatioiden jotka ottavat parametrina aidon tyypin ja palauttavat aidon tyypin tyypintoimitin on $* \Rightarrow *$ ja niin edelleen. Tyyppioperaatioita, jotka ottavat parametrikseen muun kuin aidon tyypin, eli ovat esim. muotoa $(* \Rightarrow *) \Rightarrow *$, kutsutaan *korkeamman asteen tyyppioperaatioiksi*. Toisin kuin korkeamman asteen funktioille, niille ei ole paljoa mielenkiintoisia käyttökohteita.

Esimerkiksi tyyppioperaatiosta sopii $Pair$, joka esitellään Piercen kirjassa [PIE02] luvussa 11 yksinkertaisena laajennoksena tyypitettyyn lambda-kalkyyliin. Se on käytännössä tietue, jolla on vain kaksi kenttää, mikä tekee määrittelyistä hieman selkeämpiä. Systeemissä $\lambda_{\rightarrow}$ $Pair$ esiteltiin laajennoksena, mutta systeemissä λ_{ω}se voidaan esitellä tyyppioperaationa tyypintoimittimeltaan $* \Rightarrow * \Rightarrow *$.

$$Pair = \lambda Y :: *.\ \lambda Z :: *.\ \forall X.(Y \rightarrow Z \rightarrow X) \rightarrow X;$$

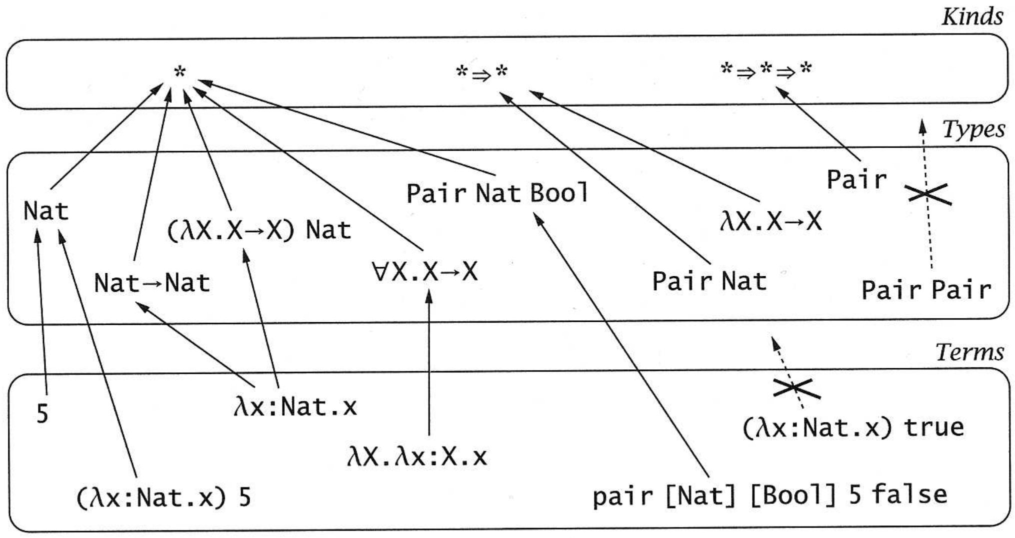
jolloin $Pair\ S\ T$ lavenee muotoon $(\lambda Y :: *.\ \lambda Z :: *.\ \forall X.(Y \rightarrow Z \rightarrow X) \rightarrow X)\ S\ T$. Tyypintoimittimen esittely $:: *$ on tapana jättää pois, mikäli se on $*$.

Itseasiassa tämä parin määrittely on oikeastaan systeemissä F_{ω}jota käsitellään myöhemmin, koska määrittelyssä on universaalikvanttori.

Kuva 1 esittelee termien, tyyppien ja tyypintoimittimien hierarkian ehkä havainnollisemmin kuin teksti.

Tyypintoimittimia onkirjallisuudessa kehitelty muunlaisiakin kuin tässä esitellyt abstraktio ja toteutus, mm. *record kind, power kind, singleton kind* (en edes yritä kääntää noita suomeksi). Niitä ei käsitellä tässä esityksessä enempää.

Seuraavat määrittelyt esittelevät yksinkertaisen tyypitetyn lambda-kalkyylin tyyppioperaatioilla täydennenttynä. Määrittely on annettu kokonaisuudessaan, jotta se olisi



Kuva 1: abstraktioiden suhteet toisiinsa

edes hieman luettava. pahoitteluni sotkuisesta esitysasusta; en löytänyt elegantimpaa tapaa latoa sitä.

Syntaksi

t ::=		termit
	x	muuttuja
	λ:T.t	abstraktio
	t t	sijoitus
v ::=		arvot:
	λ:T.t	abstraktio-arvo
T ::=		tyypit
	X	tyyppimuuttuja
	λ X::K.T	operaattoriabstraktio
	T T	operaattorisijoitus
	T $\rightarrow$ T	funktiotyyppi

Γ::=	kontekstit:
$\emptyset$	tyhjä konteksti
Γ, x:T	termimuuttujan sitominen
Γ, X::K	tyyppimuuttujan sitominen

K ::=	Tyypintoimittimet:
*	aitojen tyyppien tyyppitoimitin
K → K	operaattorien tyyppi

Evaluaatio t → t'

$$\frac{t_1 \to t_1'}{t_1 t_2 \to t_1' t_2} \quad \text{(E-App1)}$$

$$\frac{t_2 \to t_2'}{v_1 t_2 \to v_1 t_2'} \quad \text{(E-App2)}$$

$$(\lambda\ x : T_{11}.t_{12})v_2 \longrightarrow [x \to v_2]t_{12}) \quad \text{(E-AppAbs)}$$

Tyypintoimitus $\Gamma \vdash T :: K$

$$\frac{X :: K \in \Gamma}{\Gamma \vdash X :: K} \quad \text{(K-TVar)}$$

$$\frac{\Gamma, X :: K_1 \vdash T_2 :: K_2}{\Gamma \vdash \lambda X :: K_1.T_2 :: K_2} \quad \text{(K-Abs)}$$

$$\frac{\Gamma \vdash T_1 :: K_{11} \to K_{12} \quad \Gamma \vdash T_2 :: K_{11}}{\Gamma \vdash T_1 T_2 :: K_{12}} \quad \text{(K-App)}$$

$$\frac{\Gamma \vdash T_1 :: * \quad \Gamma \vdash T_2 :: *}{\Gamma \vdash T_1 \to T_2 :: *} \quad \text{(K-Arrow)}$$

Tyyppien ekvivalenssi $S \equiv T$

$$T \equiv T \quad \text{(Q-Refl)}$$

$$\frac{T \equiv S}{S \equiv T} \quad \text{(Q-Symm)}$$

$$\frac{S \equiv U \quad U \equiv T}{S \equiv T} \quad \text{(Q-Trans)}$$

$$\frac{S_1 \equiv T_1 \quad S_2 \equiv T_2}{S_1 \rightarrow S_2 \equiv T_1 \rightarrow T_2} \quad \text{(Q-Arrow)}$$

$$\frac{S_2 \equiv T_2}{\lambda X :: K_1.S_2 \equiv \lambda X :: K_1.T_2} \quad \text{(Q-Abs)}$$

$$\frac{S_1 \equiv T_1 \quad S_2 \equiv T_2}{S_1 S_2 \equiv T_1 T_2} \quad \text{(Q-App)}$$

$$(\lambda X :: K_{11}.T_{12})T2 \equiv [X \rightarrow T_2]T_{12} \quad \text{(Q-AppAbs)}$$

Tyypitys $\Gamma \vdash t : T$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \quad \text{(T-Var)}$$

$$\frac{\Gamma \vdash T_1 :: * \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1.t_2 : T_1 \rightarrow T_2} \quad \text{(T-Abs)}$$

$$\frac{\Gamma \vdash t_1 : T_1 1 \rightarrow t_1 2 \quad \Gamma \vdash t_2 : T_1 1}{\Gamma \vdash t_1 t_2 : t_1 2} \quad \text{(T-App)}$$

$$\frac{\Gamma \vdash t : S \quad S \equiv T \Gamma \vdash T :: *}{\Gamma \vdash t : T} \quad \text{(T-Eq)}$$

3 Kvanttorit ja λ_ω-kalkyyli

Seuraavaksi lisätään λ_ω-kalkyyliin universaalikvanttori, jolloin saadaan systeemi $F_\omega$ja eksistenssikvanttori. Esitelmä tyyppikvanttoreista on syytä olla käsillä tai hyvin muistissa, koska kvanttorien toimintaa, kaavoja tai notaatiota ei kerrata tässä.

3.1 Universaalikvanttori: systeemi F_ω

Lisäämällä kalkyyliin $\lambda_\omega$systeemistä F tuttu universaalikvanttori $\forall$ saadaan systeemi F_ω.

Verrattuna systeemiin F ja edellä esitettyyn λ_ω-kalkyyliin, systeemissä $F_\omega$on melko vähän uutta. Kaikille tyyppimuuttujille systeemin F säännöissä asetetaan tyypintoimitin ja lisäksi esitellään uudet tyypintoimitus ja tyyppiekvivalenssisäännöt K-All ja Q-All universaalikvanttorille.

$$\frac{\Gamma, X :: K_1 \vdash T_2 :: *}{\Gamma \vdash \forall X :: K_1.T_2 :: *} \quad \text{(K-All)}$$

$$\frac{S_2 \equiv T_2}{\forall X :: K_1.S_2 \equiv \forall X :: K_1.T_2} \quad \text{(Q-All)}$$

Systeemi $F_\omega$on esitetty Piercen kirjassa [PIE02] kokonaisena, mitä en tässä tee.

3.2 Eksistenssikvanttori

Eksistentiaalityyppien lisäämiseksi systeemiin F_ω, lisätään siihen eksistentiaalityyppien säännöt , Piercen kirjassa [PIE02] figure 24-1, siten, että operaatiot on asianmuakisesti tyypitetty ja lisäksi kaksi uutta sääntöä, K-Some eksistentiaalityypintoimittimien määrittelyyn ja Q-Some eksistentiaalityyppien ekvivalenssin määrittelyyn.

$$\frac{\Gamma, X :: K_1 \vdash T_2 :: *}{\Gamma \vdash \{\exists X :: K_1, T_2\} :: *} \quad \text{(K-Some)}$$

$$\frac{S_2 \equiv T_2}{\{\exists X :: K_1, S_2\} \equiv \{\exists X :: K_1, T_2\}} \quad \text{(Q-Some)}$$

3.3 Esimerkki

Kvantorien käyttöä tyyppioperaatioiden yhteydessä havainnollistanee seuraava esimerkki. Esimerkki on suoraan Piercen kirjasta, luvusta 30.2 ja siinä otetaan käyttöön myös perustyypit Nat ja $Bool$

Tarkoituksena on luoda polymorfinen abstrakti tietotyyppi (abstract data type, ADT) aiemmin mainituista pareista. Siis tyyppi, joka mahdollistaa parien rakentamisen ja purkamisen, mutta ei parin kenttien suoraa muokkaamista (tiedon piilotus) ja jota voidaan käyttää mitä tahansa aitoja tyyppejä olevien arvojen kanssa.

Ensin tietotyypin signature (tutummalta nimeltään rajapinta)

$$PairSig = \begin{array}{l} \{\exists Pair :: * \Rightarrow * \Rightarrow *, \\ \quad \{pair : \forall X.\ \forall Y.\ X \to Y \to (PairXY), \\ \quad\quad fst : \forall X.\ \forall Y.\ (PairXY) \to X \\ \quad\quad snd : \forall X.\ \forall Y.\ (PairXY) \to Y\}\}; \end{array}$$

Implememntaation pitää siis tarjota tyyppioperaatio $Pair$ ja polymorfiset funktiot $pair$, fst ja snd, jotka palauttavat parin, sen ensimmäisen jäsenen ja sen toisen jäsenen.

$$pairADT = \begin{array}{l} \{*\lambda X.\ \lambda Y.\ \forall R(X \to Y \to R) \to R, \\ \quad \{pair = \lambda X.\ \lambda Y.\ \lambda x : x.\ \lambda y : Y.\ \lambda \\ \quad\quad\quad \lambda R.\lambda p : X \to Y \to R.p\ x\ y, \\ \quad fst = \lambda X.\ \lambda Y.\ \lambda p : \forall R.\ (X \to Y \to R) \to R. \\ \quad\quad p[X](\lambda x : X.\ \lambda y : Y.\ x), \\ \quad snd = \lambda X.\ \lambda Y.\ \lambda p : \forall R.\ (X \to Y \to R) \to R. \\ \quad\quad p[Y](\lambda x : X.\ \lambda y : Y.\ y)\}\} as PairSig \end{array}$$

$pairADT$ voidaan nyt avata normaaliin tapaan

$$let\{Pair, pair\} = PairADT$$

ja esimerkiksi lause

$$inpair.fst[Nat][Bool](pair.pair[Nat][Bool]5true);$$

saa tällöin arvon 5.

KÄytänössä tyypintoimittimien ja kvantifioinnin avulla voidaan siis luoda sangen joustavia ja käyttökelpoisia abstrakteja tietotyyppejä - priytymistä lukuunottamatta toimiva oliojärjestelmä (periytymisestä lisää seuraavassa esityksessä).

4 Rajoitettu tyypintoimitus

Systeemin $F_\omega$osajoukoista voidaan muodostaa hierarkia $F_1, F_2, \ldots$ siten, että $\bigcup_{i\geq 0} F_i = F_\omega$.

Systeemissä F_1 ainoa tyypintoimitin on * eikä siinä ole tyyppikvanttoreita ($\forall$) tai tyyppiabstraktiota (λ). Se on siis oikeastaan yksinkertainen tyypitetty lambdakalkyyli $\lambda_\rightarrow$

Loput systeemit F_i voidaan määritellä tyypintoimittimien tasojen avulla.

$\mathcal{K}_1 = \emptyset$
$\mathcal{K}_{i+1} = \{*\} \cup \{J \Rightarrow K | J \in \mathcal{K}_i \text{ and } K \in \mathcal{K}_{i+1}\}$
$\mathcal{K}_\omega = \bigcup_{1\leq i} \mathcal{K}_i$

Systeemissä F_2 ainoa tyypintoimitin on * eikä tyyppiabstraktiota ole, mutta aitojen tyyppien kvantifiointi on sallittua. Se on sama kuin Systeemi F (jota kutsutaankin myös toisen asteen lambda-kalkyyliksi)

Systeemissä F_3 sallitaan tyyppioperaatioiden kvantifiointi, eli sillä voi kirjoittaa tyyppioperaatioita muotoa $\forall X :: K.T$, jossa $K \in \mathcal{K}_3$ ja operaatioabstraktioita $\lambda X :: *.T$, eli sillä voi käsitellä tyyppejä, joiden tyypintoimitin on esim $* \Rightarrow *$ tai $* \Rightarrow * \Rightarrow *$. Korkeammat tasot määräytyvät vastaavasti.

Piercen mukaan käytännössä ei ole juurikaan mielekkäitä sovelluksia millekään korkeampien tasojen kuin F_3 ohjelmille, mutta koko $F_\omega$sadaan tavallaan samaan hintaan: $F_3 : n$ käsittely tai toteutus ei olisi sen helpompaa.

Viitteet

PIE02 Pierce, B. C., *Types and Programming Languages*, MIT-Press, 2002