Yhtälö- ja toisinkirjoituslogiikkajärjestelmä Maude

Jyrki Niemi

Helsinki 19.4.2004

Ohjelmointikielten periaatteet -kurssin seminaariesitelmä

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Johdanto

Tässä esityksessä esittelen *yhtälö- ja toisinkirjoituslogiikkaa* (*equational and rewriting logic*) ja sillä ohjelmointia erityisesti Maude-kielen ja -järjestelmän näkökulmasta. Kuvaan ensin yhtälö- ja toisinkirjoituslogiikan yleisiä periaatteita keskittyen kuitenkin yksinkertaisempaan yhtälölogiikkaan. Sen jälkeen esittelen Maude-kielen ominaisuuksia osittain esimerkkien kautta.

Yhtälö- ja toisinkirjoituslogiikkaa voi ajatella toisaalta mallintamis- ja määrittelyvälineenä, toisaalta deklaratiivisena ohjelmointina. Toisinkirjoituslogiikka kuvaa muutosta, mutta sen sisältämä yhtälölogiikka on funktionaalista.

Yhtälö- ja toisinkirjoituslogiikoille on tehty useita toteutuksia, tässä käsittelemäni Mauden (Maude, Clavel et al. 2002) lisäksi mm. ELAN (Elan, Borovanský et al. 2002) ja CafeOBJ (CafeOBJ, Nagakawa et al. 1999).

2 Yhtälö- ja toisinkirjoituslogiikka

Toisinkirjoituslogiikan perusaksioomat ovat muotoa $t \rightarrow t'$ olevia *toisinkirjoitussääntöjä* (*rewrite rule*), missä t ja t' ovat *termejä* (*term*). Tällaisen toisinkirjoitussäännön voi tulkita laskennallisesti, siirtymänä tilasta t tilaan t', tai loogisesti, päättelysääntönä, jonka perusteella kaavasta t voi päätellä kaavan t'. Tietty aksioomajoukko muodostaa *toisinkirjoitusteorian* (*rewrite theory*).

Toisinkirjoituslogiikka sisältää yhtälölogiikan osajoukkonaan (osalogiikkanaan).

2.1 Termien toisinkirjoitus

Termien toisinkirjoituksessa käsiteltävät termit muodostuvat operaatiosymbolista (funktorista) ja sen argumenteista, jotka ovat termejä: esimerkiksi $f(a, b, c)$ tai sisämerkintäinen $a + b$. Tietyllä operaatiolla on aina kiinteä määrä argumentteja. Argumentiton termi on vakio.

Toisinkirjoituksessa termejä kirjoitetaan toisin sääntöjen perusteella: esimerkiksi toisinkirjoitussäännön $f(a, b, c) \rightarrow g(a, b)$ perusteella termi $f(a, b, c)$ kirjoitetaan termiksi $g(a, b)$. Toisinkirjoitusjärjestelmä koostuu yleensä useista toisinkirjoitussäännöistä.

Tyypillisesti toisinkirjoitussäännöt sisältävät muuttujia, jolloin ne soveltuvat moniin eri termeihin. Merkitsen seuraavassa muuttujia suuraakkosilla. Esimerkiksi toisinkirjoitussääntö $f(A, B, c) \rightarrow g(A, B)$ toimii termille $f(a, b, c)$ samoin kuin edellinen sääntö, mutta se soveltuu myös termiin $f(c, d, c)$, jonka se toisinkirjoittaa termiksi $g(c, d)$. Koska termin argumentit ovat termejä ja muuttujat voivat vastata rakenteisia termejä, sääntö soveltuu myös termiin $f(a, f(b, d, c), c)$, josta se tuottaa yhden väli-

vaiheen jälkeen termin $g(a, g(b, d))$. Säännön soveltamisjärjestys riippuu käytettävästä toisinkirjoitusstrategiasta.

Toisinkirjoitus voi olla *päättyvä* (*terminating*) tai päättymätön. Jos järjestelmä sisältää esimerkiksi toisinkirjoitussäännön $g(A, B) \rightarrow g(B, A)$, termin $g(c, d)$ toisinkirjoitus on selvästi päättymätön: siitä tulee ensin $g(d, c)$, josta jälleen $g(c, d)$, ja niin edelleen loputtomasti.

Jos toisinkirjoitussäännöt toisinkirjoittavat jokaisen termin aina yhdeksi tietyksi termiksi sääntöjen soveltamisjärjestyksestä riippumatta, toisinkirjoitusjärjestelmä on *konfluentti* (*confluent*). Jos toisinkirjoitus on päättyvä ja konfluentti, kullakin termillä on *normaalimuoto* (*normal form*), jonka toisinkirjoitussäännöt tuottavat. Toisinkirjoitus on tällöin determinististä.

Termien toisinkirjoitusta käsittelevät esimerkiksi Dershowitz and Jouannaud (1990) ja Klop (1992).

2.2 Yhtälölogiikka

Yhtälölogiikka esittää termien yhtälöitä, jotka voi kuvata toisinkirjoitussääntöinä. Voi ajatella, että toisinkirjoitussäännön kumpikin puoli tarkoittaa samaa asiaa.

Yhtälölogiikan yhtälöiden täytyy muodostaa päättyvä ja konfluentti toisinkirjoitusjärjestelmä. Tämän vuoksi yhtälölogiikka on funktionaalista: tietty termi sieventyy (toisinkirjoitetaan) aina tietyksi toiseksi termiksi.

2.3 Toisinkirjoituslogiikka

Toisinkirjoituslogiikka on ensisijaisesti muutoksen logiikkaa, jolla voi kuvata järjestelmän tilassa rinnakkaisesti tapahtuvia muutoksia. Toisinkirjoituslogiikassa termejä voi ajatella tiloina ja toisinkirjoitussääntöjä siirtymäsääntöinä tilasta toiseen.

Toisinkirjoituslogiikan toisinkirjoitussääntöjen ei tarvitse olla päättyviä eikä konfluentteja. Käytännön toteutuksissa voi tyypillisesti määritellä sovellettavan toisinkirjoitusstrategian eli miten ja missä järjestyksessä sääntöjä sovelletaan termeihin.

Toisinkirjoituslogiikka on reflektiivistä: sillä pystyy esittämään toisinkirjoituslogiikan teorian ja päättelysäännöt metatasona. Tämä mahdollistaa esimerkiksi toisinkirjoitusstrategioihin vaikuttamisen.

2.4 Toisinkirjoituslogiikan sovelluksia

Toisinkirjoituslogiikkaa on sovellettu moniin eri tehtäviin (Martí-Oliet and Meseguer 2002: 7–15). Toisinkirjoituslogiikkaa voi käyttää *semanttisena kehyksenä* (*semantic framework*) erilaisten laskennan mallien ja kielten esittämiseen tai *loogisena kehyksenä*

(*logical framework*) erilaisten logiikoiden ja päättelyjärjestelmien esittämiseen ja niiden ominaisuuksien päättelyyn.

Toisinkirjoituslogiikka on vahva laskennan malli: toisinkirjoitusteorioina voi kuvata monia erilaisia laskennan malleja neuroverkoista ja tosiaikajärjestelmistä lambda-kalkyyliin ja Turingin koneeseen asti. Ohjelmointikielten semantiikkaakin voi kuvata toisinkirjoitusteorioina. Toisinkirjoituslogiikkaa on käytetty myös ohjelmien ja tietoliikenneprotokollien formaaliin määrittelyyn ja analysointiin.

3 Maude

Tämä luku perustuu pääasiassa Maude-järjestelmän käyttöoppaaseen (Clavel et al. 2004) ja alkeisoppaaseen (McCombs 2003) sekä pieneltä osin Mauden vanhempaa versiota kuvaavaan oppaaseen (Clavel et al. 2000).

3.1 Maude-järjestelmä

Maude on yksi toisinkirjoituslogiikan toteutuksista: se on sekä kieli että kielen toteuttava järjestelmä. Mauden nykyinen toteutus perustuu välikielikääntäjään. Maude on GNU GPL-lisensoitu, ja se toimii useissa Unix-tyyppisissä järjestelmissä. Mauden tuorein versio on 2.1, joka on saatavilla Mauden kotisivulta (Maude).

Maude-järjestelmän toteutus koostuu kahdesta osasta: Core Maude sisältää C++-kielellä toteutetun ydinosan ja Full Maude Maudella itsellään toteutettuja laajennuksia. Tässä käsittelen lähinnä Core Maudea ja erityisesti sen yhtälölogiikkapuolta.

3.2 Mauden perusrakenteet

Maude-ohjelman perusyksikkö on moduuli, joka kuvaa yhtälö- tai toisinkirjoitusteorian eli tietyille operaattoreille (termeille) määriteltyjen yhtälöiden tai toisinkirjoitussääntöjen (tai molempien) joukon. Pelkkiä yhtälöitä sisältävä moduuli on *funktionaalinen moduuli*, joka määrittelee funktioita sievennyssääntöinä käytettyjen yhtälöiden avulla. Toisinkirjoitussääntöjä sisältävä moduuli on *järjestelmämoduuli* (*system module*).

Mauden funktionaalinen moduuli on muotoa `fmod` *moduuli* `is ... endfm`. Moduuli voi sisältää toisten moduulien käyttöönottoja sekä tyyppien, alityyppisuhteiden, operaattorien, muuttujien, tyyppien jäsenyyksien ja yhtälöiden määrittelyitä. Järjestelmämoduulin määrittelyn avainsanat ovat `mod` ja `endm`, ja se voi sisältää funktionaalisessa moduulissa sallittujen osien lisäksi varsinaisia toisinkirjoitussääntöjä.

3.3 Tyyppijärjestelmä

Maude on vahvasti tyypitetty järjestelmä: kullakin operaattorilla ja sen operandeilla on määrätty tyyppi. Tyypit muodostavat alityyppihierarkian, jossa alityypin alkiot kuuluvat ylityyppiin. Operaattori soveltuu määritellyn tyypin lisäksi myös sen alityypeille.

Tyyppi määritellään lauseella `sort` tai `sorts`, alityypit lauseella `subsort` tai `subsorts`:

```
sorts Elt List .
subsort Elt < List .
```

Tämä määrittelee tyypit `Elt` ja `List` sekä sen, että `Elt` on tyypin `List` alityyppi.

Tietyn tyypin sisältämät arvot (termit) voi määritellä joko *tyypinmuodostimina* (*constructor*) toimivien operaattorien avulla tai jäsenyyslauseella, jolla voi kuvata alityypin sisältävän määrätyt ylityypin arvot.

3.4 Operaattorit

Operaattorien (termien) muoto (*signature*) eli niiden nimi ja operandien ja tuloksen (eli koko termin) tyypit määritellään `op` tai `ops`-lauseilla. Lisäksi tässä yhteydessä voi määritellä operaattoreille tiettyjä algebrallisia ominaisuuksia, kuten liitännäisyyden tai vaihdannaisuuden. Esimerkiksi

```
op nil : -> List [ctor] .
op __ : List List -> List [ctor assoc id: nil] .
```

määrittelee nollapaikkaisen operaattorin `nil`, jonka tuloksena on tyyppiä `List` oleva termi, ja kaksipaikkaisen operaation, joka tuottaa kahdesta listasta yhden listan.

Maude sallii operaattorien nimille ja niiden operandien sijoittelulle varsin vapaan sisä- tai *sekamerkintäisen* (*mixfix*) syntaksin. Operaattorin nimessä alaviiva `_` kuvaa yhtä operandipaikkaa. Tässä kahden listan yhdistäminen tapahtuu peräkkäisyydellä, ilman erityistä operaattorisymbolia, mitä kuvaa operaattorin määrittelyssä `__`. Jos operandeja saavan operaattorin määritelmässä ei käytetä operandipaikkojen merkkejä, operaattori käyttää tavallista termien suljesyntaksia: esimerkiksi kun on määritelty

```
op 0 : -> Nat [ctor] .
op s : Nat -> Nat [ctor] .
```

on mahdollista muodostaa termi `s(s(s(0)))`.

Hakasulkeissa oleva lista kuvaa operaattorin attribuutteja tai ominaisuuksia. Tässä `ctor` tarkoittaa tyypinmuodostinta. Kaksioperandisen sisämerkintäisen operaattorin voi määritellä *liitännäiseksi* (*associative*) (`assoc`), *vaihdannaiseksi* (*commutative*) (`comm`) tai *idempotentiksi* (`idem`). Lisäksi operaattorille voi määritellä identiteettialkion (`id`), joka edellä olevassa esimerkissä on `nil`.

Nämä operaattorien ominaisuudet vastaavat periaatteessa vastaavien yhtälöiden määrittelyä. Niiden määrittely yhtälöinä tuottaisi kuitenkin helposti päättymättömiä sievennyksiä.

Attribuuttien avulla operaattoreille voi määritellä myös yksinkertaisen toisinkirjoitusstrategian: operaattorin operandeihin (termin alitermeihin) voi soveltaa yhtälöiden toisinkirjoitusta halutussa järjestyksessä tai jättää jotkin operandit kokonaan toisinkirjoittamatta. Oletuksena Maude käyttää soveltavaa laskentajärjestystä, eli se toisinkirjoittaa alitermit ennen termin itsensä toisinkirjoittamista.

Alityyppihierarkiaan perustuvan operaattorien nimien kuormittamisen lisäksi Maude sallii nimien vapaan kuormittamisen myös muille tyypeille.

3.5 Muuttujat

Maudessa muuttujat ovat aina yhtälö- tai sääntökohtaisia: muuttuja kiinnitetään yhtälön tai säännön tietyllä soveltamiskerralla tiettyyn termiin. Tämä muistuttaa jossain määrin Prologin tapaa käsitellä muuttujia, vaikka Maudessa ei olekaan unifikaatiota.

Tietynnimisen muuttujan tyypin voi määritellä moduulikohtaisesti `var`- tai `vars`-lauseella:

```
var L : List .
vars N M : Nat .
```

Tällöin kaikilla moduulissa olevilla tämännimisillä muuttujilla on tämä tyyppi. Muuttujan tyypin voi määritellä myös muuttujan käytön yhteydessä, mutta silloin tyyppi täytyy mainita erikseen muuttujan jokaisen käytön yhteydessä. Muuttujan tyyppiä ei voi määritellä suoraan yhdelle säännölle.

3.6 Yhtälöt

Yhtälöt kuvaavat Mauden funktionaalisten moduulien operaattoreista muodostettujen termien yhtäläisyyksiä eli miten niitä voi toisinkirjoittaa moduulin kuvaaman teorian sisällä. Yhtälö määritellään `eq`-lauseella. Esimerkiksi Peanon kokonaislukujen yhteenlasku voidaan määritellä seuraavasti:

```
op _+_ : Nat Nat -> Nat .
vars N M : Nat .
eq N + 0 = N .
eq N + s M = s (N + M) .
```

Yhdelle operaattorille on tyypillisesti useita yhtälöitä, joissa operanditermeillä on eri rakenne.

Yhtälö voi olla myös ehdollinen, jolloin käytetään avainsanaa `ceq`. Esimerkiksi luku voidaan jakaa toisella, jos jakaja ei ole nolla (toteutus jätetty pois):

```
op _/_ : Nat Nat -> Nat .
ceq N / M = ... if not (M == 0) .
```

Funktionaalisen moduulin yhtälöiden täytyy kuvata toisinkirjoituksen kannalta päättyvä ja konfluentti teoria. Maude ei kuitenkaan mitenkään tarkista näitä ominaisuuksia, joten laskenta voi jäädä päättymättömään silmukkaan.

3.7 Toisinkirjoitussäännöt

Varsinaisia toisinkirjoitussääntöjä voi käyttää vain Mauden järjestelmämoduuleissa. Toisinkirjoitussääntö kuvaa paikallisen rinnakkaisen siirtymän jossakin järjestelmässä:

```
sort weathercondition .
op sunnyday : -> weathercondition .
op rainyday : -> weathercondition .
rl [raincloud] : sunnyday => rainyday .
rl [sun] : rainyday => sunnyday .
```

3.8 Muita ominaisuuksia

Maude sisältää kirjastomoduuleina perustietotyyppien määrittelyt: totuusarvot, luonnolliset luvut, kokonais-, rationaali- ja liukuluvut sekä merkkijonot. Luonnolliset, kokonais- ja rationaaliluvut kuvaavat Peanon kokonaislukuihin pohjautuvan semantiikan, mutta niiden operaatiot ovat tehokkuussyistä sisäänrakennettuja.

Full Maude sisältää tuen olio-ohjelmoinnin käsitteille: luokille, olioille, viestille ja moniperinnälle. Ne kaikki on toteutettu toisinkirjoituslogiikan sisällä. Full Maude sisältää myös monipuolisen parametrisen polymorfismin. Moduulin voi parametreja voi rajoittaa vaatimalla niiltä tietyt operaatiot ja yhtälöt.

Maude on refleksiivinen kieli: Maude-ohjelmat voi esittää datana, jota ohjelma voi käsitellä. Full Maude käyttää tätä hyväksi erityisesti moduulien parametroinnin toteuttamisessa.

Mauden nykyinen versio ei mahdollista vuorovaikutusta ulkoisten olioiden kanssa: Maudella ei voi käsitellä esimerkiksi tiedostoja tai muuta syöte- tai tulostevirtaa, joskin tämä on ilmeisesti suunnitteilla (Clavel et al. 2004: 7). Maude-järjestelmää käytetään Maude-tulkin komentoriviltä, jossa voi antaa termin funktionaalisen moduulin yhtälöillä sievennettäväksi tai järjestelmämoduulin säännöillä toisinkirjoitettavaksi.

3.9 Esimerkki

Seuraava esimerkki esittää yksinkertaisen kokonaisen Mauden funktionaalisen moduulin, joka kuvaa Peanon kokonaisluvut (McCombs 2003: 15):

```
fmod PEANO-NAT is
  sort Nat .
  op 0 : -> Nat [ctor] .
  op s : Nat -> Nat [ctor iter] .
  op _+_ : Nat Nat -> Nat .
  vars M N : Nat .
  eq 0 + N = N .
  eq s(M) + N = s(M + N) .
endfm
```

Maude-järjestelmällä sieventäminen onnistuu seuraavasti:

```
Maude> reduce s(s(s(0))) + s(s(s(s(0)))) .
reduce in PEANO-NAT-EXTRA : s^3(0) + s^5(0) .
rewrites: 4 in 0ms cpu (0ms real) (~ rewrites/second)
result Nat: s^8(0)
```

Operaattorin `s` attribuutti `iter` sallii termien potenssiesitysmuodon.

4 Yhteenveto

Yhtälö- ja toisinkirjoituslogiikka ovat deklaratiivisia logiikoita, mutta ne voi toteuttaa myös laskennallisesti termien toisinkirjoittamisena. Toisinkirjoituslogiikka sisältää yhtälölogiikan, jossa kullakin termillä täytyy olla yksi normaalimuoto. Toisinkirjoituslogiikka on yhtäaikaisen muutoksen logiikkaa, jossa toisinkirjoitussäännöt ovat paikallisia tilasiirtymiä.

Toisinkirjoituslogiikalla voi kuvata monia erilaisia logiikoita ja laskennan malleja. Laskentamallina se itse on Turingin koneen vahvuinen.

Yksi yhtälö- ja toisinkirjoituslogiikan toteutus on Maude, joka on kieli ja sen toteutus.

Lähteet

Peter Borovanský, Horatiu Cirstea, Hubert Dubois, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, Quang-Huy Nguyen, Christophe Ringeissen, and Marian Vittek. *ELAN: User Manual*. LORIA, Villers-lès-Nancy, France, ELAN V3.6 edition, September 2002. URL `http://www.loria.fr/equipes/protheo/SOFTWARES/ELAN/manual/userm/manual.pdf`.

CafeOBJ. CafeOBJ. WWW-sivu, January 2001. URL `http://www.ldl.jaist.ac.jp/cafeobj/index.html`.

M[anuel] Clavel, F[rancisco] Durán, S[teven] Eker, P[atrick] Lincoln, N[arciso] Martí-Oliet, J[osé] Meseguer, and J[osé F.] Quesada. *A Maude Tutorial*. SRI International, 2000. URL `http://maude.cs.uiuc.edu/papers/postscript/CDELMMQtutorial_2000.ps.gz`.

Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243, August 2002. ISSN 0304-3975. Esiversio saatavilla osoitteesta `http://maude.cs.uiuc.edu/papers/postscript/CDELMMQspecprog_2001.ps.gz`.

Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *Maude Manual (Version 2.1)*. SRI International, Menlo Park, CA, March 2004. URL `http://maude.cs.uiuc.edu/maude2-manual/maude-manual.pdf`.

Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 243–320, Amsterdam, 1990. Elsevier – MIT Press.

Elan. Elan home page. WWW-sivu. URL `http://www.loria.fr/equipes/protheo/SOFTWARES/ELAN/`.

J[an] W[illem] Klop. Term rewriting systems. In S. Abramsky, Dov M. Gabbay, and T. S. E. Mailbaum, editors, *Handbook of Logic in Computer Science*, volume 2: Background: Computational Structures, pages 1–116, Oxford, 1992. Oxford University Press.

Narciso Martí-Oliet and José Meseguer. Rewriting logic: roadmap and bibliography. *Theoretical Computer Science*, 285(2):121–154, August 2002. ISSN 0304-3975. Esiversio saatavilla osoitteesta `http://maude.cs.uiuc.edu/papers/postscript/MMroadmap_2001.ps.gz`.

Maude. The Maude system. WWW-sivu, 2004. URL `http://maude.cs.uiuc.edu/`.

Theodore McCombs. Maude 2.0 primer, August 2003. URL `http://maude.cs.uiuc.edu/primer/maude-primer.pdf`. Version 1.0.

Ataru T. Nagakawa, Toshimi Sawada, and Kokichi Futatsugi. *CafeOBJ User's Manual*. Japan Advanced Institute of Science and Technology, ver. 1.4 edition, 1999. URL `http://www.ldl.jaist.ac.jp/cafeobj/manual/manual.ps.gz`.