Programare Logică

Ioana Leustean

http://moodle.fmi.unibuc.ro/course/view.php?id=186

Conținutul cursului

- Introducere in MAUDE
- Signaturi și algebre multisortate. Termeni
- Morfisme. Tipuri abstracte de date. Algebre inițialĕ
- Subalgebre. Algebre libere
- Semantica termenilor
- Ecuații. Relația de satisfacere
- Specificații algebrice
- Γ-algebra inițială

- Teorema constantelor. Demonstrații prin inducție
- Logica ecuațională: sintaxa
- Logica ecuațională: corectitudine și completitudine
- Regula "Subterm Replacement"
- Rescrierea termenilor
- Substituții. Algoritmul de unificare

- Sisteme de rescriere abstracte
- Rescrierea termenilor: terminare, confluență, completare
- Logica ecuațională și rescriere locală
- Deducție și rescriere modulo axiome
- Semantica algebrei inițiale
- Clauze Horn. Rezoluție

Programare Logica

Maude

http://maude.cs.uiuc.edu/

- Este dezvoltat la:
  - University of Illinois at Urbana-Champaign (UIUC)
  - Stanford Research Institute (SRI)
- Aparține familiei OBJ, al carei initiator a fost Joseph Goguen(1941 - 2006).

- Poate fi folosit ca demonstrator. Mecanismul de rescriere este un procedeu de demonstrare automata, dar sunt implementate si facilități speciale.
- Poate fi un instrument util in dezvoltarea de software, deoarece permite atât specificarea, cât și analiza unui limbaj de programare. Faptul ca este executabil oferă și avantajul unei implementări indirecte.

- Este un interpretor.
- Comenzile sunt introduse una câte una și sunt executate imediat.
- Un "program" este o mulțime de module.
- Modulele pot fi scrise în fișiere sau direct în linia de comandă.
- Modulele pot fi importate. Modulul predefinit BOOL este importat de orice modul.

Comenzi sistem

in ../*nume-fisier.mod*
show module *nume-modul* .
select *nume-modul* .
show *sorts* .
show *ops* .
reduce *termen* .
reduce *termen1* == *termen2* .
parse *term* .
set trace on .
set trace off .
quit

Manual de Maude (html)


```
Maude> in PL/Exemple-Curs1.maude
==========================================
fmod MYNAT
Maude> show module MYNAT .
fmod MYNAT is
  sorts Nat .
  op 0 : -> Nat .
  op s_ : Nat -> Nat .
endfm
Maude> select MYNAT .
```



```
Maude> reduce s s s s s 0 .
reduce in MYNAT : s s s s s 0 .
rewrites: 0 in 6641663802ms cpu (0ms real) (0 rew/sec)
result Nat: s s s s s 0
Maude> parse s s s s s 0 .
Nat: s s s s s 0
Maude> reduce  s s s 0 ==  s s 0 .
reduce in MYNAT : s s s 0 == s s 0 .
rewrites: 1 in 10534570934ms cpu (0ms real) (0 rew/sec)
result Bool: false
```


Exemplul 1

```
fmod MYNAT is
  sorts Nat .
  op 0 : -> Nat .
  op s_ : Nat -> Nat .
endfm
```

Acest modul introduce tipul de date `Nat`.
Datele de tip `Nat` sunt:

`0, s 0, s s 0, s s s 0, s s s s 0, s s s s s  0, ...`

Reprezentarea matematica:
$S = \{Nat\}$, $\Sigma = \{0 :\to Nat, s : Nat \to Nat\}$
(S, Σ) este o signatură și definește o clasă de algebre (structuri, structuri algebrice). Algebra $(\mathbb{N}, 0, succesor)$ este un obiect "privilegiat" al acestei clase.

fmod ... endfm

```
fmod MYNATLIST is
protecting MYNAT .
sort ListNat .
subsort Nat < ListNat .
op nil : -> ListNat .
op _;_ : ListNat ListNat -> ListNat [ assoc ] .
var L : ListNat .
eq nil ;  L = L .
eq L ; nil = L .
endfm
```

Structura generală a unui modul:

• importuri (protecting, extending, including),
• declararea sorturilor si a subsorturilor,
• declararea operațiilor,
• declararea variabilelor,
• ecuații.

Importuri

Exista trei modalitati de importare a modulelor

- **protecting**
  se foloseste atunci cand datele definite in modulul important nu sunt afectate de operatiile dau ecuatiile noului modul: nu se construiesc date noi pe sorturi vechi ("no junk") si nu sunt identificate date care in modulul initial erau diferite ("no confusion")
- **extending**
  permite aparitia unor date noi pe sorturile vechi ("junk") dar nu permite identificarea datelor care in modulul initial erau diferite ("no confusion")
- **including**
  nu are restrictii

[assoc]

```
op _;_ : ListNat ListNat -> ListNat [ assoc ] .
```

Atributul [assoc] inlocuieste ecuatiile:

```
eq (L;P);Q = L;(P;Q) .
eq L;(P;Q) = (L;P);Q .
```

Ecuatiile definite cu **eq** sunt transformate in reguli de rescriere. Cele doua ecuatii care definesc asociativitatea duc la neterminarea rescrierii, deci ele nu pot fi adaugate in sectiunea **eq**. Efectul atributului **[assoc]** este faptul ca rescrierea se face pe clase de termeni "modulo asociativitate". In mod asemanator, atributul **[comm]** se foloseste pentru a indica faptul ca o operatie este comutativa.

Exemplul 2

```
fmod MYNATLIST is
protecting MYNAT .
sort ListNat .
subsort Nat < ListNat .
op nil : -> ListNat .
op _;_ : ListNat ListNat -> ListNat [ assoc ] .
var L : ListNat .
eq nil ;  L = L .
eq L ; nil = L .
endfm
```

Exemplul 2

Reprezentarea matematica:

```
sort Nat ListNat .
subsort Nat < ListNat .
op 0 : -> Nat .
op s_ : Nat -> Nat .
op nil : -> ListNat .
op _;_ : ListNat ListNat -> ListNat .
```

Signatura (ordonat-sortată): $((S, \leq), \Sigma)$
$S = \{Nat, ListNat\}$, $\leq \subseteq S \times S$, $\leq = \{(Nat, ListNat)\}$
$\Sigma = \{0 :\to Nat, s : Nat \to Nat, nil :\to ListNat,$
$\quad ;: ListNat\ ListNat \to ListNat\}$

Exemplul 2

Reprezentarea matematica:

```
op _;_ : ListNat ListNat -> ListNat [ assoc ] .
var L : ListNat .
eq nil ;  L = L .
eq L ; nil = L .
```

Prezentare (multime de ecuatii):
$(\gamma_1)\forall L._{ListNat}(nil; L = L)$
$(\gamma_2)\forall L._{ListNat}(L; nil = L)$
$(\gamma_3)\forall L._{ListNat}\forall P._{ListNat}\forall Q._{ListNat}((L; P); Q = L; (P; Q))$
$\Gamma = \{\gamma_1, \gamma_2, \gamma_3\}$

Exemplul 2

```
fmod MYNATLIST is
protecting MYNAT .
sort ListNat .
subsort Nat < ListNat .
op nil : -> ListNat .
op _;_ : ListNat ListNat -> ListNat [ assoc ] .
var L : ListNat .
eq nil ;  L = L .
eq L ; nil = L .
endfm
```

Reprezentarea matematica a unui modul este o specificatie algebrica ordonat-sortată

$$((S, \leq), \Sigma, \Gamma)$$

Exemplul 2

Date de tipul ListNat sunt:

```
 nil ; s 0
 s s 0 ; s 0 ; 0
(nil ; s 0) ; s s 0 ; nil ; 0
```

Aceste date le numim **expresii** sau **termeni**.
Un program este un modul, adică o specificație; modelul matematic al unei specificații este o algebră de termeni; o execuție este o rescriere în algebra de termeni asociată.

```
Maude> select MYNATLIST .
Maude> reduce  (nil ; s 0) ; s s 0 ; nil ; 0 .
reduce in MYNATLIST : (nil ; s 0) ; s s 0 ; nil ; 0 .
rewrites: 2 in 2753904389ms cpu (0ms real)
result ListNat: s 0 ; s s 0 ; 0
```

Exemplul 2

```
Maude> set trace on .
Maude> reduce  (nil ; s 0) ; s s 0 ; nil ; 0 .
reduce in MYNATLIST : (nil ; s 0) ; s s 0 ; nil ; 0 .
*********** equation
eq nil ; L = L .
L --> s 0
nil ; s 0 ---> s 0
*********** equation
eq nil ; L = L .
L --> 0
nil ; 0 ---> 0
rewrites: 2 in 2753904388ms cpu (12ms real)
result ListNat: s 0 ; s s 0 ; 0
```


Exemplul 2

```
fmod MYNATLIST is
protecting MYNAT .
sort ListNat .
subsort Nat < ListNat .
op nil : -> ListNat .
op _;_ : ListNat ListNat -> ListNat [ assoc ] .
var L : ListNat .
eq nil ;  L = L .
eq L ; nil = L .
endfm
```

Ce date defineste MYNATLIST?

Exemplul 2

```
fmod MYNATLIST is
protecting MYNAT .
sort ListNat .
subsort Nat < ListNat .
op nil : -> ListNat .
op _;_ : ListNat ListNat -> ListNat [ assoc ] .
var L : ListNat .
eq nil ;  L = L .
eq L ; nil = L .
endfm
```

Modulul MYNATLIST definește $\bigcup_{k\geq 0} \mathbb{N}^k$
(secvențele ordonate finite de numere naturale).

```
Maude> reduce (0 ; s 0 ) == (0 ; s 0 ; 0) .
...
result Bool: false
```


Exemplul 3

```
fmod MYNATLIST1 is
protecting MYNAT .
sort ListNat .
subsort Nat < ListNat .
op nil : -> ListNat .
op _;_ : ListNat ListNat -> ListNat [ assoc comm ] .
var L : ListNat .
var I : Nat .
eq nil ;  L = L .
eq L ; nil = L .
eq I ; I = I .
endfm
```

```
Maude> reduce (0 ; s 0)  == (0 ; s 0 ; 0) .
...
result Bool: true
```

Ce defineste MYNATLIST1 ?

Exemplul 4

```
fmod COMPLEXINT is
protecting INT .
sort ComplexInt .
subsort Int < ComplexInt .
op _+i_ : Int Int -> ComplexInt .
op i_ : Int -> ComplexInt .
op _+_ : ComplexInt ComplexInt -> ComplexInt [ditto] .
var z : Int .
eq 0 +i z = i z .
eq z +i 0 = z .
...
endfm
```

Observați supraîncărcarea operației + si atributul [ditto]

$+ : Int\ Int -> Int$

$+ : ComplexInt\ ComplexInt -> ComplexInt$

Exemplul 5

```
fmod GRUP is
sort  Element .
op e : -> Element .
op  _+_  : Element Element -> Element [ assoc ] .
op -_  : Element -> Element .
vars x y : Element .
eq e + x = x .
eq x + e = x .
eq (- x) + x = e .
eq x + (- x)  = e .
eq - - x = x .
eq - e = e .
eq - (x + y) = (- y) + (- x) .
endfm
```


TRS canonic

O ecuație $l = r$ se transformă, prin orientare de la stânga la dreapta, într-o regulă de rescriere $l \to r$. Ecuațiile modulului GRUP determină astfel un sistem de rescriere **canonic**:

- orice șir de rescrieri se termină,
- ordinea de aplicare regulilor de rescriere nu schimbă rezultatul final.

O ecuație $t_1 = t_2$ din teoria de ordinul I a grupurilor este verificată în orice grup dacă și numai dacă există un termen t astfel încât $t_1 \stackrel{*}{\to} t$ și $t_2 \stackrel{*}{\to} t$.
Maude-ul poate fi utilizat ca demonstrator.

Exemplul 5

```
Maude> select GRUP .
Maude> reduce  x + (-( - y + x)) == y .
reduce in GRUP : x + - (- y + x) == y .
********** equation
eq - (x + y) = - y + - x .
x --> - y
y --> x
- (- y + x) ---> - x + - - y
********** equation
eq - - x = x .
x --> y
- - y ---> y
```

Exemplul 5

```
*********** equation
eq x + - x = e .
x --> x
x + - x + y ---> e + y
*********** equation
eq e + x = x .
x --> y
e + y ---> y
*********** equation
(built-in equation for symbol _==_)
y == y ---> true
rewrites: 5 in 179156223ms cpu (23ms real)
result Bool: true
```


Exemplul 6

```
fmod MYNATLIST is
protecting MYNAT .
sort ListNat .
subsort Nat < ListNat .
op nil : -> ListNat .
op _;_ : ListNat ListNat -> ListNat [ assoc ] .
var L : ListNat .
eq nil ;  L = L .
eq L ; nil = L .
endfm
```

```
fmod MYNATLIST2 is
protecting MYNATLIST .
op _<_ : Nat Nat -> Bool .
...
endfm
```


Exemplul 6

```
fmod MYNATLIST2 is
protecting MYNATLIST .
op _<_ : Nat Nat -> Bool .
vars I J : Nat .
eq s I < s J = I < J .
eq  0 < s I  = true .
eq  I < 0 = false .
ceq I ; J = J ; I if (J < I ) .
endfm
```

Observati **ecuatia conditionala**

```
ceq I ; J = J ; I if (J < I ) .
```

Ce tip de date defineste modulul MYNATLIST2 ?

Exemplul 6

```
Maude> select MYNATLIST2 .
Maude> reduce s s s 0 ; s s 0 ;  s s s 0 ; 0 .
...
result ListNat: 0 ; s s 0 ; s s s 0 ; s s s 0
```

Modulul MYNATLIST2 defineste liste de numere naturale ordonate crescator.

A se vedea si D. Dragulici, Revista de logica

Exemplul 7

```
fmod STACK{X :: TRIV} is
sorts EmptyStack{X} NonEmptyStack{X} Stack{X} .
subsorts EmptyStack{X} NonEmptyStack{X} < Stack{X} .
op empty : -> EmptyStack{X} .
op push : X$Elt Stack{X} -> NonEmptyStack{X} .
op pop_ : NonEmptyStack{X} -> Stack{X} .
op top_ : NonEmptyStack{X} -> X$Elt .
var E : X$Elt .
var S : Stack{X} .
eq top push (E , S) = E .
eq pop push (E , S) = S .
endfm
```

Modulul STACK$\{X\}$ crează o stivă generică.
Parametrul formal X este descris de teoria TRIV

```
fth TRIV is sort Elt. endfh
```


Exemplul 8

```
fth TRIV is
sort Elt.
endfh

fmod STACK{X :: TRIV} is ... endfm
```

Instanțiind modulul parametru X cu modulul predefinit NAT definim stivele de numere naturale.

```
view Inst from TRIV to NAT is
sort Elt to Nat .
endv

fmod STACKNAT is
protecting STACK{Inst} .
endfm
```


Exemplul 8

```
Maude> select STACKNAT .
Maude> show sorts .
sort Bool .
sort Zero .      subsort Zero < Nat .
sort NzNat .     subsort NzNat < Nat .
sort Nat .       subsorts NzNat Zero < Nat .
sort EmptyStack{Inst} .
subsort EmptyStack{Inst} < Stack{Inst} .
sort NonEmptyStack{Inst} .
subsort NonEmptyStack{Inst} < Stack{Inst} .
sort Stack{Inst} .
subsorts NonEmptyStack{Inst} EmptyStack{Inst} < Stack{Inst} .
```


Limbajul Maude este un limbaj de specificație bazat pe logica ecuațională. Un program este o colecție de module. Execuția este o rescriere. Câteva din caracteristicile acestui limbaj sunt:

- modularizare si parametrizare,
- definirea tipurilor de date este independentă de implementare,
- extensibilitate,
- permite tratarea erorilor și supraîncărcarea operațiilor,
- poate fi folosit ca demonstrator.

Signaturi. Algebre. Termeni

$S \neq \emptyset$

- O mulțime S- sortată A este o familie $A = \{A_s\}_{s \in S}$.
- Dacă $A = \{A_s\}_{s \in S}$ și $B = \{B_s\}_{s \in S}$ atunci
  - $\emptyset = \{\emptyset_s\}_{s \in S}$, $\emptyset_s = \emptyset$ or. $s \in S$,
  - $A \subseteq B \Leftrightarrow A_s \subseteq B_s$ or. $s \in S$,
  - $A \cup B = \{A_s \cup B_s\}_{s \in S}$, $A \cap B = \{A_s \cap B_s\}_{s \in S}$,
  - $A \times B = \{A_s \times B_s\}_{s \in S}$.

Exemplu: $S = \{nat, bool\}$, $A = \{A_{nat}, A_{bool}\}$,
$A_{nat} = \mathbb{N}$, $A_{bool} = \{T, F\}$

sorturi=tipuri, elemente de sort s= date de tip s

$A = \{A_s\}_{s \in S}$, $B = \{B_s\}_{s \in S}$, $C = \{C_s\}_{s \in S}$

- O funcție S- sortată $f : A \to B$ este o familie de funcții $f = \{f_s\}_{s \in S}$, unde $f_s : A_s \to B_s$ oricare $s \in S$.
- Dacă $f : A \to B$ și $g : B \to C$, definim $f; g : A \to C$, $f; g = \{f_s; g_s\}_{s \in S}$
- $f_s; g_s : A_s \to C_s$,
  $f_s; g_s(a) := g_s(f_s(a))$ or. $s \in S$, $a \in A_s$
- $1_A : A \to A$, $1_A = \{1_{A_s}\}_{s \in S}$
- $(f; g); h = f; (g; h)$ (compunerea este asociativă)
- $f; 1_B = f$, $1_A; f = f$ or. $f : A \to B$

$A = \{A_s\}_{s \in S}$, $B = \{B_s\}_{s \in S}$,

- O funcție S-sortată $f : A \to B$ se numește injectivă, (surjectivă, bijectivă) dacă f_s este injectivă, (surjectivă, bijectivă) oricare $s \in S$.
- O funcție S-sortată $f : A \to B$ se numește inversabilă dacă există $g : B \to A$ a.î. $f; g = 1_A$ și $g; f = 1_B$.

Propoziție. O funcție S-sortată $f : A \to B$ este inversabilă dacă și numai dacă este bijectivă (f_s este bijectivă oricare $s \in S$).

Signaturi multisortate

(S, Σ) signatură multisortată

- S mulțimea sorturilor
- Σ mulțimea simbolurilor de operații $\sigma : s_1 \cdots s_n \to s$
  - σ este simbolul (numele) operației
  - $s_1, \cdots, s_n, s \in S$
    $s_1, \cdots, s_n$ sorturile argumentelor
    s sortul rezultatului
    $< s_1 \cdots s_n, s >$ aritatea operației
  - dacă $n = 0$ atunci $\sigma :\to s$ este simbolul unei operații *constante*
- $\Sigma = (\Sigma_{w,s})_{w \in S^*, s \in S}$
  $\sigma \in \Sigma_{w,s} \Leftrightarrow \sigma : w \to S$
  $w = s_1 \cdots s_n \in S^*$
- este permisă supraîncărcarea operațiilor

Signaturi

(S, Σ) signatură multisortată

- $S^* := \bigcup_{n \geq 0} S^n$
  $S^0 := \{\lambda\}$, $S^n := \{s_1 \cdots s_n | s_i \in S \text{ or. } i\}$
- $\Sigma = (\Sigma_{w,s})_{w \in S^*, s \in S}$
  $\sigma \in \Sigma_{w,s} \Leftrightarrow \sigma : w \to s$
  $w = s_1 \cdots s_n \in S^*$
- σ este *supraîncărcat* (*overloaded*) dacă
  $\sigma \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$ și $< w_1, s_1 > \neq < w_2, s_2 >$
- este permisă supraîncărcarea operațiilor

Exemple

- $BOOL = (S, \Sigma)$
  - $S = \{bool\}$
  - $\Sigma = \{T :\to bool, F :\to bool,$
    $\neg : bool \to bool,$
    $\vee : bool\ bool \to bool,$
    $\wedge : bool\ bool \to bool\}$
- $NAT = (S, \Sigma)$
  - $S = \{nat\}$
  - $\Sigma = \{0 :\to nat, succ : nat \to nat\}$

Exemple

- $NATBOOL = (S, \Sigma)$
  - $S = \{bool, nat\}$
  - $\Sigma = \{T :\to bool, F :\to bool, 0 :\to nat,$
    $succ : nat \to nat,$
    $\leq : nat\ nat \to bool\}$
- $\Sigma = (\Sigma_{w,s})_{w \in S^*, s \in S}$
  $\Sigma_{\lambda,bool} = \{T, F\}$, $\Sigma_{\lambda,nat} = \{0\}$,
  $\Sigma_{nat,nat} = \{succ\}$, $\Sigma_{nat\ nat,bool} = \{\leq\}$,
  $\Sigma_{w,s} = \emptyset$ pentru celelalte $< w, s > \in S^* \times S$

Exemple

- $STIVA = (S, \Sigma)$
  - $S = \{elem, stiva\}$
  - $\Sigma = \{0 :\to elem, empty :\to stiva,$
    $push : elem\ stiva \to stiva,$
    $pop : stiva \to stiva,$
    $top : stiva \to elem\}$

Exemple

- $AUTOMAT = (S, \Sigma)$
  - $S = \{intrare, stare, iesire\}$
  - $\Sigma = \{s0 :\to stare,$
    $f : intrare\ stare \to stare,$
    $g : stare \to iesire\}$
- $GRAF = (S, \Sigma)$
  - $S = \{arc, nod\}$
  - $\Sigma = \{v0 : arc \to nod, \quad v1 : arc \to nod\}$

Signaturi ordonat-sortate

$(S, \leq, \Sigma)$ signatură ordonat-sortată

- (S, Σ) signatură multisortată
- $(S, \leq)$ mulțime parțial ordonată
- condiția de monotonie
  $\sigma \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$, $w_1 \leq w_2 \Rightarrow s_1 \leq s_2$

Exemplu:
$S = \{elem, stiva, nvstiva\}$, $elem \leq stiva$, $nvstiva \leq stiva$
$\Sigma = \{empty :\to stiva, \quad push : elem\ stiva \to nvstiva,$
$pop : nvstiva \to stiva, top : nvstiva \to elem\}.$
În practică se folosesc signaturi ordonat-sortate.

Algebre multisortate

(S, Σ) - signatură multisortată
O algebră multisortată de tip (S, Σ) este o pereche (A_S, A_Σ), unde

- $A_S = \{A_s\}_{s \in S}$ (mulțimea suport)
- $A_\Sigma = \{A_\sigma\}_{\sigma \in \Sigma}$ (familie de operații) a.î.
  - dacă $\sigma :\to s$ atunci $A_\sigma \in A_s$
  - dacă $\sigma : s_1 \cdots s_n \to s$ atunci $A_\sigma : A_{s_1} \times \cdots \times A_{s_n} \to A_s$

$A = (A_S, A_\Sigma)$ este o (S, Σ)-algebră

Exemple

- $BOOL = (S = \{bool\}, \Sigma)$
  $\Sigma = \{T :\to bool, F :\to bool, \neg : bool \to bool,$
  $\quad \vee : bool\ bool \to bool, \wedge : bool\ bool \to bool\}$
- $BOOL$-algebra A:
  $A_{bool} := \{0, 1\}$
  $A_T := 1,\ A_F := 0,\ A_\neg(x) := 1 - x,$
  $A_\vee(x, y) := max(x, y),\ A_\wedge(x, y) := min(x, y)$
- $BOOL$-algebra B:
  $B_{bool} := \mathcal{P}(\mathbb{N})$
  $B_T := \mathbb{N},\ B_F := \emptyset,\ B_\neg(X) := \mathbb{N} \setminus X,$
  $B_\vee(X, Y) := X \cup Y,\ B_\wedge(X, Y) := X \cap Y$

Exemple

- $NAT = (S = \{nat\}, \Sigma)$
  $\Sigma = \{0 :\to nat, succ : nat \to nat\}$
- NAT-algebra A:
  $A_{nat} := \mathbb{N}$
  $A_0 := 0,\ A_{succ}(x) := x + 1$
- NAT-algebra B:
  $B_{nat} := \{0, 1\}$
  $B_0 := 0,\ B_{succ}(x) := 1 - x$
- NAT-algebra C:
  $C_{nat} := \{2^n | n \in \mathbb{N}\}$
  $C_0 := 1,\ C_{succ}(2^n) := 2^{n+1}$

Exemple

- $STIVA = (S = \{elem, stiva\}, \Sigma)$
  $\Sigma = \{0 :\to elem, empty :\to stiva, pop : stiva \to stiva,$
  $\quad push : elem\ stiva \to stiva, top : stiva \to elem\}$
- $STIVA$-algebra A: $\quad A_{elem} := \mathbb{N},\ A_{stiva} := \mathbb{N}^*$
  $A_0 := 0,\ A_{empty} := \lambda,\ A_{push}(n, n_1 \cdots n_k) := n\, n_1 \cdots n_k,$
  $A_{pop}(\lambda) = A_{pop}(n) := \lambda,$
  $A_{pop}(n_1 n_2 \cdots n_k) := n_2 \cdots n_k$ pt. $k \geq 2,$
  $A_{top}(\lambda) := 0,\ A_{top}(n_1 \cdots n_k) := n_1$ pt. $k \geq 1.$
- $STIVA$-algebra B: $\quad B_{elem} := \{0\},\ B_{stiva} := \mathbb{N}$
  $B_0 := 0,\ B_{empty} := 0,\ B_{push}(0, n) := n + 1$ or. $n,$
  $B_{pop}(0) := 0,\ B_{pop}(n) := n - 1$ pt. $n \geq 1,$
  $B_{top}(n) := 0$ or. $n.$

Exemple

- $AUTOMAT = (S = \{intrare, stare, iesire\}, \Sigma)$
  $\Sigma = \{s0 :\to stare, f : intrare\ stare \to stare,$
  $\quad g : stare \to iesire\}$
- $AUTOMAT$-algebra A:
  $A_{intrare} = \{x, y\},\ A_{stare} = \{s0, s1\},\ A_{iesire} := \{T, F\}$
  $A_{s0} := s0,\ A_g(s0) := F,\ A_g(s1) := T,$
  $A_f(x, s0) := s0,\ A_f(y, s0) := s1,$
  $A_f(x, s1) := s0,\ A_f(y, s1) := s1$
- $AUTOMAT$-algebra B:
  $B_{intrare} = B_{stare} = B_{iesire} := \mathbb{N}$
  $B_{s0} := 0,\ B_f(m, n) := m + n,\ B_g(n) := n + 1$

Algebre ordonat-sortate

$(S, \leq, \Sigma)$ signatură ordonat-sortată
O algebră ordonat-sortată de tipul $(S, \leq, \Sigma)$ este o (S, Σ)-algebră (A_S, A_Σ) care satisface următoarele proprietăți:

- $s_1 \leq s_2 \Rightarrow A_{s_1} \subseteq A_{s_2}$
- $\sigma \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$, $w_1 \leq w_2 \Rightarrow$
  $A_\sigma^{w_2,s_2}(\mathbf{x}) = A_\sigma^{w_1,s_1}(\mathbf{x})$ oricare $\mathbf{x} \in A_{w_1}$.
- Semantica unui modul functional in **Maude** este o algebră ordonat-sortată.

Mulțime de variabile

(S, Σ) signatură multisortată

$|\Sigma| := \bigcup_{w,s} \Sigma_{w,s}$

$|X| := \bigcup_{s \in S} X_s$ dacă X mulțime S-sortată

O mulțime de variabile este o mulțime S-sortată X a.î.:

- $X_s \cap X_{s'} = \emptyset$ or. $s \neq s'$
- $|X| \cap |\Sigma| = \emptyset$

simbolurile de variabile sunt distincte între ele și sunt distincte de simbolurile de operații din Σ

Termeni (expresii, cuvinte)

(S, Σ) signatură, X mulțime de variabile

Mulțimea S-sortată termenilor cu variabile din X, $T_\Sigma(X)$, este *cea mai mică* mulțime de *șiruri finite* peste alfabetul
$$L = \bigcup_{s \in S} X_s \cup \bigcup_{w,s} \Sigma_{w,s} \cup \{\mathbf{(}, \mathbf{)}\} \cup \{\mathbf{,}\}$$
care verifică următoarele proprietăți:

- (T1) $X_s \subseteq T_\Sigma(X)_s$
- (T2) dc. $\sigma :\to s$, at. $\boldsymbol{\sigma} \in T_\Sigma(X)_s$,
- (T3) dc. $\sigma : s_1 \cdots s_n \to s$ și $t_i \in T_\Sigma(X)_{s_i}$ or. $i = 1, \ldots, n$
  at. $\boldsymbol{\sigma}(\mathbf{t_1}, \ldots, \mathbf{t_n}) \in T_\Sigma(X)_s$.

• $Var(t) :=$ mulțimea variabilelor care apar în $t \in T_\Sigma$

Termeni fără variabile (ground terms)

(S, Σ) signatură multisortată

Mulțimea S-sortată termenilor fără variabile, T_Σ, este *cea mai mică* mulțime de *șiruri finite* peste alfabetul
$$L = \bigcup_{w,s} \Sigma_{w,s} \cup \{\mathbf{(}, \mathbf{)}\} \cup \{\mathbf{,}\}$$
care verifică următoarele proprietăți:

- (T2) dc. $\sigma :\to s$, at. $\boldsymbol{\sigma} \in T_{\Sigma,s}$,
- (T3) dc. $\sigma : s_1 \cdots s_n \to s$ și $t_i \in T_{\Sigma,s_i}$ or. $i = 1, \ldots, n$
  at. $\boldsymbol{\sigma}(\mathbf{t_1}, \ldots, \mathbf{t_n}) \in T_{\Sigma,s}$.

• $T_\Sigma = T_\Sigma(\emptyset)$

Exemple

- $STIVA = (S, \Sigma)$
  $S = \{elem, stiva\}$, $X_{elem} = \{x, y\}$, $X_{stiva} = \emptyset$,
  $\Sigma = \{0 :\to elem, empty :\to stiva,$
  $push : elem\ stiva \to stiva,$
  $pop : stiva \to stiva,$
  $top : stiva \to elem\}$
- $T_\Sigma(X)_{elem} = \{\mathbf{0}, x, y, \mathbf{top(pop(empty))},$
  $\mathbf{top(push(}x\mathbf{, empty))}, \ldots\}$
  $T_\Sigma(X)_{stiva} = \{\mathbf{empty}, \mathbf{push(}y\mathbf{, empty)}, \mathbf{pop(empty)},$
  $\mathbf{push(top(empty), empty)}, \ldots\}$
- șiruri care nu sunt termeni
  $pop(0)$, $(pop)top(empty)$, $empty(y)$, ...

Exemple

- $NATBOOL = (S, \Sigma)$
  $S = \{bool, nat\}$
  $\Sigma = \{T :\to bool, F :\to bool, 0 :\to nat,$
  $s : nat \to nat,$
  $\leq : nat\ nat \to bool\}$
- $T_{NATBOOL,nat} = \{\mathbf{0}, \mathbf{s(0)}, \mathbf{s(s(0))}, \ldots\}$
  $T_{NATBOOL,bool} = \{\mathbf{T}, \mathbf{F}, \leq \mathbf{(0,0)}, \leq \mathbf{(0, s(0))}, \ldots\}$
- șiruri care nu sunt termeni
  $\leq (T, F)$, $s \leq (0)$, $Ts(0)$, ...

Exemple

- $AUTOMAT = (S, \Sigma)$, $S = \{intrare, stare, iesire\}$,
  $\Sigma = \{s0 :\to stare,$
  $f : intrare\ stare \to stare,$
  $g : stare \to iesire\}$
  $T_{AUTOMAT,stare} = \{\mathbf{s0}\}$, $T_{AUTOMAT,intrare} = \emptyset$,
  $T_{AUTOMAT,iesire} = \{\mathbf{g(s0)}\}$
- $GRAF = (S, \Sigma)$, $S = \{arc, nod\}$
  $\Sigma = \{v0 : arc \to nod, \quad v1 : arc \to nod\}$
  $T_{GRAF,arc} = T_{GRAF,nod} = \emptyset$

Expresii aritmetice

- $NATEXP = (S = \{nat\}, \Sigma)$
- $\Sigma = \{0 :\to nat, s : nat \to nat,$
  $+ : nat\ nat \to nat, * : nat\ nat \to nat\}$
- $T_{NATEXP} = \{\mathbf{0}, \mathbf{s(0)}, \mathbf{s(s(0))}, \ldots$
  $\mathbf{+(0,0)}, \mathbf{*(0, +(s(0), 0))}, \mathbf{*(s(0), s(s(0)))}, \ldots\}$
- șiruri care nu sunt termeni
  $+(0)$, $0(s)s(0)$, $*(0)$, ...
- T_{NATEXP} este mulțimea expresiilor aritmetice peste $\mathbb{N}$.

Inducția pe termeni

(S, Σ) signatură multisortată, X mulțime de variabile
Fie **P** o proprietate a.î. următoarele condiții sunt satisfăcute:

- pasul inițial:
  $\mathbf{P}(x) = true$ or. $x \in X$, $\mathbf{P}(\sigma) = true$ or. $\sigma :\to s$,
- pasul de inducție:
  dacă $t_1 \in T_\Sigma(X)_{s_1}, \ldots, t_n \in T_\Sigma(X)_{s_n}$ și
  $\mathbf{P}(t_1) = \cdots = \mathbf{P}(t_n) = true$ atunci
  $\mathbf{P}(\boldsymbol{\sigma}(\mathbf{t_1}, \ldots, \mathbf{t_n})) = true$ or. $\sigma : s_1 \ldots s_n \to s$.

Atunci $\mathbf{P}(t) = true$ oricare $t \in T_\Sigma(X)$.

Algebra termenilor

Algebra termenilor

(S, Σ) signatură, X mulțime de variabile
Mulțimea termenilor $T_\Sigma(X) = \{T_\Sigma(X)_s\}_{s \in S}$ este (S, Σ)-algebră cu operațiile definite astfel:

- pt. $\sigma :\to s$, operația corespunzătoare este $T_\sigma := \boldsymbol{\sigma}$
- pt. $\sigma : s_1 \cdots s_n \to s$, operația corespunzătoare este
  $T_\sigma : T_\Sigma(X)_{s_1} \times \cdots \times T_\Sigma(X)_{s_n} \to T_\Sigma(X)_s$
  $T_\sigma(t_1, \ldots, t_n) := \boldsymbol{\sigma}(\mathbf{t_1}, \ldots, \mathbf{t_n})$
  or. $t_1 \in T_\Sigma(X)_{s_1}, \ldots, t_n \in T_\Sigma(X)_{s_n}$

$T_\Sigma(X)$ algebra termenilor cu variabile din X
T_Σ algebra termenilor fără variabile ($X = \emptyset$)

Algebra expresiilor aritmetice

- $NATEXP = (S = \{nat\}, \Sigma)$
- $\Sigma = \{0 :\to nat, s : nat \to nat,$
  $+ : nat\ nat \to nat, * : nat\ nat \to nat\}$
- $T_0 := \mathbf{0}$,
- $T_s(t) := \mathbf{s(t)}$,
- $T_+(t_1, t_2) := \mathbf{+(t_1, t_2)}$
- $T_*(t_1, t_2) := \mathbf{*(t_1, t_2)}$

- Semantica unui modul în **Maude** (care conține numai declarații de sorturi, operații și variabile) este o algebră de termeni.

Morfisme.
Tipuri abstracte de date.
Algebre inițiale

Exemplu. Signatura
$MONOID = (S = \{s\}, \Sigma = \{e :\to s, * : ss \to s\})$
Fie $(A, 1, \star)$ și $(B, 0, +)$ monoizi
$(A_e := 1,\ A_* := \star,\ B_e := 0,\ B_* := +)$
O funcție $f : A \to B$ este morfism de monoizi dacă:

- $f(1) = 0 \Leftrightarrow f(A_e) = B_e$
- $f(x \star y) = f(x) + f(y) \Leftrightarrow f(A_*(x, y)) = B_*(f(x), f(y))$ or. x, $y \in A$.

$$\begin{array}{ccccc} A & \times & A & \stackrel{A_*}{\to} & A \\ f\downarrow & & f\downarrow & & f\downarrow \\ B & \times & B & \stackrel{B_*}{\to} & B \end{array}$$

(A_S, A_Σ), (B_S, B_Σ) (S, Σ)-algebre

Un morfism de (S, Σ)-algebre ((S, Σ)-*morfism*) este o funcție S-sortată $f : A \to B$ care verifică condițiile de compatibilitate:

- $f_s(A_\sigma) = (B_\sigma)$ oricare $\sigma :\to s$,
- $f_s(A_\sigma(a_1, \ldots, a_n)) = B_\sigma(f_{s_1}(a_1), \ldots, f_{s_n}(a_n))$ or. $\sigma : s_1 \cdots s_n \to s$, or. $(a_1, \ldots, a_n) \in A_{s_1} \times \cdots \times A_{s_n}$.

$$\begin{array}{ccccccc} A_{s_1} & \times & \cdots & \times & A_{s_n} & \stackrel{A_\sigma}{\to} & A_s \\ f_{s_1}\downarrow & & \cdots & & f_{s_n}\downarrow & & f_s\downarrow \\ B_{s_1} & \times & \cdots & \times & B_{s_n} & \stackrel{B_\sigma}{\to} & B_s \end{array}$$

Observație

$1_A : A \to A$ este morfism or. A (S, Σ)-algebră.

- $NAT = (S = \{nat\}, \Sigma = \{0 :\to nat, succ : nat \to nat\})$
- NAT-algebra A:
  $A_{nat} := \mathbb{N},\ A_0 := 0,\ A_{succ}(x) := x + 1$
- NAT-algebra B:
  $B_{nat} := \{0, 1\},\ B_0 := 0,\ B_{succ}(x) := 1 - x$
- $f : A \to B$, $f = \{f_{nat}\}$, $f_{nat}(n) = n(mod\ 2)$ este morfism de NAT-algebre
- Nu există morfism de NAT-algebre $g : B \to A$.

Exemple

- $STIVA = (S = \{elem, stiva\}, \Sigma)$
  $\Sigma = \{0 :\to elem, empty :\to stiva, pop : stiva \to stiva,$
  $push : elem\ stiva \to stiva, top : stiva \to elem\}$
- $STIVA$-algebra A: $A_{elem} := \mathbb{N}$, $A_{stiva} := \mathbb{N}^*$
  $A_0 := 0$, $A_{empty} := \lambda$, $A_{push}(n, n_1 \cdots n_k) := n\, n_1 \cdots n_k$,
  $A_{pop}(\lambda) = A_{pop}(n) := \lambda$,
  $A_{pop}(n_1 n_2 \cdots n_k) := n_2 \cdots n_k$ pt. $k \geq 2$,
  $A_{top}(\lambda) := 0$, $A_{top}(n_1 \cdots n_k) := n_1$ pt. $k \geq 1$.
- $STIVA$-algebra B: $B_{elem} := \{0\}$, $B_{stiva} := \mathbb{N}$
  $B_0 := 0$, $B_{empty} := 0$, $B_{push}(0, n) := n + 1$ or. n,
  $B_{pop}(0) := 0$, $A_{pop}(n) := n - 1$ pt. $n \geq 1$,
  $B_{top}(n) := 0$ or. n.

Exemple

- $STIVA$-algebra $A = (A_{elem} = \mathbb{N}, A_{stiva} = \mathbb{N}^*, \ldots)$
- $STIVA$-algebra $B = (B_{elem} = \{0\}, B_{stiva} := \mathbb{N}, \ldots)$
- $f : A \to B$, $f = (f_{elem} : \mathbb{N} \to \{0\}, f_{stiva} : \mathbb{N}^* \to \mathbb{N})$
  $f_{elem}(n) := 0$ or. n,
  $f_{stiva}(\lambda) := 0$, $f_{stiva}(n_1 \cdots n_k) := k$ pt. $k \geq 1$
- $g : B \to A$, $g = (g_{elem} : \{0\} \to \mathbb{N}, g_{stiva} : \mathbb{N} \to \mathbb{N}^*)$
  $g_{elem}(0) := 0$,
  $g_{stiva}(0) := \lambda$, $g_{stiva}(k) := \underbrace{0 \cdots 0}_{k}$ pt. $k \geq 1$

  f și g sunt morfisme de $STIVA$-algebre

Morfisme de mpo

Observație.

Mulțimile parțial ordonate se pot reprezenta ca (S, Σ)-algebre,
$S := \{elem, bool\}$,
$\Sigma := \{\leq : elem\ elem \to bool, T :\to bool, F :\to bool\}$.

- Mpo $(A, \leq)$ este reprezentată ca (S, Σ)-algebră prin $(A, \{0, 1\} \leq, T, F)$, unde
  $A_{elem} := A$, $A_{bool} := \{0, 1\}$, $A_T := 1$, $A_F := 0$,
  $A_{\leq}(x, y) = T \Leftrightarrow x \leq y$
- Ce relațiile este între funcții crescătoare și morfisme de (S, Σ)-algebre?
- Orice (S, Σ)-morfism este funcție crescătoare dar invers nu este adevărat!

Proprietăți

(S, Σ) signatură multisortată

Propoziția 1.

Fie A, B, C algebre și $f : A \to B$, $g : B \to C$ morfisme. Atunci $f; g : A \to C$ este morfism.

Dem.

- $\sigma :\to s \in \Sigma$
  $(f; g)_s(A_\sigma) = g_s(f_s(A_\sigma)) = g_s(B_\sigma) = C_\sigma$,
- $\sigma : s_1 \cdots s_n \to s \in \Sigma$, $(a_1, \ldots, a_n) \in A_{s_1} \times \cdots \times A_{s_n}$
  $(f; g)_s(A_\sigma(a_1, \ldots, a_n)) = g_s(f_s(A_\sigma(a_1, \ldots, a_n))) =$

  $g_s(B_\sigma(f_{s_1}(a_1), \ldots, f_{s_n}(a_n))) = C_\sigma(g_{s_1}(f_{s_1}(a_1)), \ldots, g_{s_n}(f_{s_n}(a_n)))$
  $= C_\sigma((g; f)_{s_1}(a_1), \ldots, (g; f)_{s_n}(a_n))$. *qed*

Proprietăți

Teorema 2.
Exista un unic morfism $f : T_\Sigma \to B$ or. B (S,Σ)-algebră.

Dem. Fie B (S,Σ)-algebră.
Existența. Definim $f : T_\Sigma \to B$ prin inducție pe termeni

$$(\mathbf{P}(t) = \text{"}\ f(t) \text{ este definită"}).$$

- pasul inițial: dc. $\sigma :\to s \in \Sigma$, at. $f_s(\boldsymbol{\sigma}) := B_\sigma$,
- pasul de inducție: dc. $\sigma : s_1 \dots s_n \to s \in \Sigma$ și $t_1 \in T_{\Sigma s_1}$, ..., $t_n \in T_{\Sigma s_n}$ a.î $f_{s_1}(t_1)$, $\cdots$, $f_{s_n}(t_n)$ definite at.
$$f_s(\boldsymbol{\sigma}(\mathbf{t_1,\dots,t_n})) := B_\sigma(f_{s_1}(t_1),\cdots,f_{s_n}(t_n))$$

Conform principiului inductiei pe termeni,
$f_s(t)$ este definită or. $t \in T_\Sigma$.

Continuare

Demonstrăm că f este morfism.

- dc. $\sigma :\to s \in \Sigma$, at. $f_s(T_{\Sigma\sigma}) = f_s(\boldsymbol{\sigma}) = B_\sigma$,
- dc. $\sigma : s_1 \dots s_n \to s \in \Sigma$ și $t_1 \in T_{\Sigma s_1}$, ..., $t_n \in T_{\Sigma s_n}$ at.
$$f_s(T_{\Sigma\sigma}(t_1,\dots,t_n)) = f_s(\boldsymbol{\sigma}(\mathbf{t_1,\dots,t_n})) = B_\sigma(f_{s_1}(t_1),\cdots,f_{s_n}(t_n))$$

Continuare

Unicitatea. Fie $g : T_\Sigma \to B$ un morfism.
Demonstram că $g = f$ prin inducție pe termeni

$$(\mathbf{P}(t) = \text{" dc. } t \in T_{\Sigma s} \text{ at. } g_s(t) \text{ și } f_s(t) \text{ sunt egale"}).$$

- pasul inițial: dc. $\sigma :\to s \in \Sigma$, at. $g_s(\boldsymbol{\sigma}) = B_\sigma = f_s(\boldsymbol{\sigma})$,
- pasul de inducție: dc. $\sigma : s_1 \dots s_n \to s \in \Sigma$ și $t_1 \in T_{\Sigma s_1}$, ..., $t_n \in T_{\Sigma s_n}$ a.î $g_{s_1}(t_1) = f_{s_1}(t_1)$, $\cdots$, $g_{s_n}(t_n) = f_{s_n}(t_n)$ at.
$$g_s(\boldsymbol{\sigma}(\mathbf{t_1,\dots,t_n})) = B_\sigma(g_{s_1}(t_1),\cdots,g_{s_n}(t_n)) = B_\sigma(f_{s_1}(t_1),\cdots,f_{s_n}(t_n)) = f_s(\boldsymbol{\sigma}(\mathbf{t_1,\dots,t_n}))$$

Conform principiului inductiei pe termeni,
$g_s(t) = f_s(t)$ or. $t \in T_{\Sigma s}$, adică $g = f$. *qed*

Exemplu

(S,Σ) signatură multisortată

- $D = (D_S, D_\Sigma)$, $D_s := \mathbb{N}$ or. $s \in S$
dacă $\sigma :\to s$ atunci $D_\sigma := 0$,
dacă $\sigma : s_1 \dots s_n \to s$, k_1, ..., $k_n \in \mathbb{N}$
$$D_\sigma(k_1,\dots,k_n) := 1 + max(k_1,\dots,k_n)$$
- $f_D : T_\Sigma \to D$ unicul morfism
Ce reprezintă valoarea $f_D(t)$ pentru un termen t?

Termeni ca arbori

Un termen t poat fi reprezentat ca un arbore $arb(t)$ astfel:

- $\boldsymbol{\sigma} \in T_{\Sigma_s} \Rightarrow \qquad arb(\boldsymbol{\sigma}) := \quad \boldsymbol{\sigma}$
- $t = \boldsymbol{\sigma}(\mathbf{t_1}, \ldots, \mathbf{t_n}) \Rightarrow$

$$arb(t) := \quad \begin{array}{ccc} & \boldsymbol{\sigma} & \\ \swarrow & & \searrow \\ arb(t_1) & \cdots & arb(t_n) \end{array}$$

Pentru orice t valoarea $f_D(t)$ este adâncimea arborelui $arb(t)$.

Izomorfisme

Un (S, Σ)-morfism $f : A \to B$ este izomorfism dacă există un morfism $g : B \to A$ a.î. $f; g = 1_A$ și $g; f = 1_B$.

Propoziția 3

Un morfism $f : A \to B$ este izomorfism dacă și numai dacă este funcție bijectivă, adică $f_s : A_s \to B_s$ este bijecție oricare $s \in S$.

Dem. Revine la a demonstra că implicația:

$$f \text{ morfism și bijecție} \Rightarrow f^{-1} \text{ morfism.}$$

Continuare

Demonstrăm că $f^{-1} : B \to A$ este morfism.

- dc. $\sigma :\to s \in \Sigma$, at. $f_s(A_\sigma) = B_\sigma$, deci $f_s^{-1}(B_\sigma) = A_\sigma$,
- dc. $\sigma : s_1 \ldots s_n \to s \in \Sigma$ și
  $b_1 \in B_{s_1}$, ..., $b_n \in B_{s_n}$ at.
  ex. $a_1 \in A_{s_1}$, ..., $a_n \in A_{s_n}$ a.î.
  $f_{s_1}(a_1) = b_1$, ..., $f_{s_n}(a_n) = b_n$ și
  $f_s(A_\sigma(a_1, \ldots, a_n)) = B_\sigma(f_{s_1}(a_1), \ldots, f_{s_n}(a_n)) = B_\sigma(b_1, \ldots, b_n)$

$$\text{Atunci } f_s{}^{-1}(B_\sigma(b_1, \ldots, b_n)) = A_\sigma(a_1, \ldots, a_n) = A_\sigma(f_{s_1}{}^{-1}(b_1), \ldots, f_{s_n}{}^{-1}(b_n)). \quad qed$$

Izomorfisme de mpo

- $(A = \{x_1, x_2, x_3, x_4\}, \leq)$, unde $x_1 \leq x_2 \leq x_3 \leq x_4$
- $(B = \{0, 1\}^2, \leq)$, unde $\leq$ ordinea pe componente
- A și B sunt (S, Σ)-algebre, unde
  $S := \{elem, bool\}$,
  $\Sigma := \{\leq: elem\ elem \to bool, T :\to bool, F :\to bool\}$
- $f : A \to B$
  $f(00) = x_1$, $f(01) = x_2$, $f(10) = x_3$, $f(11) = x_4$
  este funcție bijectivă și crescătoare
- $A \not\simeq B$

Izomorfisme

Spunem că algebrele $A = (A_S, A_\Sigma)$ și $B = (B_S, B_\Sigma)$ sunt izomorfe dacă există un izomorfism $f : A \to B$. În acest caz notăm $A \simeq B$.

- $A \simeq A$ (1_A este izomorfism)
- $A \simeq B \Rightarrow B \simeq A$
- $A \simeq B,\ B \simeq C \Rightarrow A \simeq C$

Relația de izomorfism este o relație de echivalență (reflexivă, simetrică și tranzitivă).

Observație.

$A \simeq B \Rightarrow A_s \simeq B_s$ oricare $s \in S$

Exemple

- $NAT = (S = \{nat\}, \Sigma = \{0 :\to nat, succ : nat \to nat\})$
- NAT-algebra A: $A_{nat} = \mathbb{N}$, $A_0 := 0$, $A_{succ}(x) := x + 1$
- NAT-algebra B: $B_{nat} := \{0, 1\}$, $B_0 := 0$, $B_{succ}(x) := 1 - x$
- NAT-algebra C: $C_{nat} := \{2^n | n \in \mathbb{N}\}$
  $C_0 := 1$, $C_{succ}(2^n) := 2^{n+1}$
- $A \not\simeq B$
- $A \simeq C$
  $f : A \to C$, $f(n) := 2^n$ este un NAT-izomorfism.

Exemple

- $BOOL = (S = \{bool\}, \Sigma)$
  $\Sigma = \{T :\to bool, F :\to bool, \neg : bool \to bool,$
  $\vee : bool\ bool \to bool, \wedge : bool\ bool \to bool\}$
- $A_{bool} := \{0, 1\}$
  $A_T := 1$, $A_F := 0$, $A_\neg(x) := 1 - x$,
  $A_\vee(x, y) := max(x, y)$, $A_\wedge(x, y) := min(x, y)$
- $B_{bool} := \mathcal{P}(\mathbb{N})$
  $B_T := \mathbb{N}$, $B_F := \emptyset$, $B_\neg(X) := \mathbb{N} \setminus X$,
  $B_\vee(X, Y) := X \cup Y$, $B_\wedge(X, Y) := X \cap Y$
- $A \not\simeq B$

Exemple

- $BOOL$-algebra C:
  $C_{bool} := \{t : \mathbb{N} \to \{0, 1\} |\ t$ funcție$\}$
  $C_T(n) := 1$, $C_F(n) := 0$ or. $n \in \mathbb{N}$
  $C_\neg(t)(n) := 1 - t(n)$ or. $t \in C_{bool}$, $n \in \mathbb{N}$
  $C_\vee(t_1, t_2)(n) := max(t_1(n), t_2(n))$ or. t_1, $t_2 \in C_{bool}$, $n \in \mathbb{N}$
  $C_\wedge(t_1, t_2)(n) := min(t_1(n), t_2(n))$ or. t_1, $t_2 \in C_{bool}$, $n \in \mathbb{N}$
- $B \simeq C$
  $f : B \to C$, $f(Y) := \chi_Y$ oricare $Y \in \mathcal{P}(\mathbb{N})$
  $f(Y)(n) = 1$ dacă $n \in Y$, $f(Y)(n) = 0$ dacă $n \notin Y$
  f este $BOOL$-izomorfism

Exemple

- $STIVA$-algebra $A = (A_{elem} = \mathbb{N}, A_{stiva} = \mathbb{N}^*, \ldots)$
- $STIVA$-algebra $B = (B_{elem} = \{0\}, B_{stiva} := \mathbb{N}, \ldots)$
- $A \not\simeq B$

Exemple

- $STIVA$-algebra A: $A_{elem} := \mathbb{N}$, $A_{stiva} := \mathbb{N}^*$ $A_0 := 0$,
  $A_{empty} := \lambda$, $A_{push}(n, n_1 \cdots n_k) := n\, n_1 \cdots n_k$,
  $A_{pop}(\lambda) = A_{pop}(n) := \lambda$,
  $A_{pop}(n_1 n_2 \cdots n_k) := n_2 \cdots n_k$ pt. $k \geq 2$,
  $A_{top}(\lambda) := 0$, $A_{top}(n_1 \cdots n_k) := n_1$ pt. $k \geq 1$.
- $STIVA$-algebra C: $C_{elem} := \mathbb{Z}$, $C_{stiva} := \mathbb{Z}^*$
  $C_0 := 0$, $C_{empty} := \lambda$, $C_{push}(x, x_1 \cdots x_k) := x_1 \cdots x_k x$,
  $C_{pop}(\lambda) = C_{pop}(x) := \lambda$,
  $C_{pop}(x_1 \cdots x_{k-1} x_k) := x_1 \cdots x_{k-1}$ pt. $k \geq 2$,
  $C_{top}(\lambda) := 0$, $C_{top}(x_1 \cdots x_k) := x_k$ pt. $k \geq 1$.

Exemple

- $STIVA$-algebra A: $A_{elem} := \mathbb{N}$, $A_{stiva} := \mathbb{N}^*$
- $STIVA$-algebra C: $C_{elem} := \mathbb{Z}$, $C_{stiva} := \mathbb{Z}^*$
- $f : A \to C$, $f = (f_{elem} : \mathbb{N} \to \mathbb{Z}, f_{stiva} : \mathbb{N}^* \to \mathbb{Z}^*)$

  $f_{elem}(2k) := k$, $f_{elem}(2k+1) := -k-1$ pt. $k \in \mathbb{N}$,

  $f_{stiva}(n_1 \cdots n_k) := f_{elem}(n_k) \cdots f_{elem}(n_1)$ pt. $n_1 \cdots n_k \in \mathbb{N}^*$.

  f este $STIVA$-izomorfism
- Algebrele izomorfe sunt "identice" (modulo redenumire).

ADT

- Un *tip abstract de date* este o mulțime de date (valori) și operații asociate lor, a căror descriere (specificare) este independentă de implementare.
  abstract=disassociated from any specific instance
- O algebră este formată dintr-o mulțime de elemente și o mulțime de operații. Algebrele pot modela tipuri de date.
- Două algebre izomorfe au același comportament, deci trebuie sa fie modele ale același tip de date. Aceasta asigură independența de implementare.

ADT

- O signatură (S, Σ) este interfața sintactică a unui tip abstract de date.
- O algebră $A = (A_S, A_\Sigma)$ este o posibilă implementare.
- Dacă $A \simeq B$, atunci A și B implementează același tip de date.
- Un tip abstract de date este o clasă $\mathcal{C}$ de (S, Σ)-algebre cu proprietatea că oricare două algebre sunt izomorfe:

$$A,\ B \in \mathcal{C} \Rightarrow A \simeq B.$$

- $[A] := \{B\ (S, \Sigma)$ algebră$|\ A \simeq B\}$ este tip abstract de date.

Algebra inițială

$\mathcal{K}$ clasă de (S, Σ)-algebre

- O (S, Σ)-algebră I este inițială în $\mathcal{K}$ dacă $I \in \mathcal{K}$ și pentru orice $B \in \mathcal{K}$ există un *unic morfism* $f : I \to B$.

Propoziția 4

- (a) I inițială în $\mathcal{K}$, $A \in \mathcal{K}$, $A \simeq I \Rightarrow A$ inițială în $\mathcal{K}$
- (b) A_1 și A_2 inițiale în $\mathcal{K} \Rightarrow A_1 \simeq A_2$.

Continuare

Dem. (a) I inițială în $\mathcal{K}$, $A \in \mathcal{K}$ și $\varphi_A : A \to I$ izomorfism

Fie $B \in \mathcal{K}$ și $f_B : I \to B$ unicul morfism. Demonstrăm că există un unic morfism $h : A \to B$.

- *Existența.* $h := \varphi_A; f_B : A \to B$ mofism
- *Unicitatea.* dacă $g : A \to B$ morfism, atunci ${\varphi_A}^{-1}; g : I \to B$ morfism,deci ${\varphi_A}^{-1}; g = f_B$; rezultă $g = \varphi_A; f_B = h$.

(b) A_1 și A_2 inițiale în $\mathcal{K}$

Există un unic morfism $f : A_1 \to A_2$ și un unic morfism $g : A_2 \to A_1$. Atunci $f; g : A_1 \to A_1$ și $1_{A_1} : A_1 \to A_1$, deci $f; g = 1_{A_1}$. Analog $g; f = 1_{A_2}$, deci $A_1 \simeq A_2$. *qed*

(S, Σ) -algebra inițială

(S, Σ) signatură multisortată

- $\mathcal{K}$ clasa tuturor (S, Σ)-algebrelor.
- I este (S, Σ)-algebră inițială dacă oricare B (S, Σ)-algebră există un unic morfism $f : I \to B$.

  Teorema 2. T_Σ este (S, Σ)-algebră inițială.

- $\mathcal{I}_{(S,\Sigma)} = \{I\ |\ I\ (S, \Sigma)$ -algebră inițială$\}$ este un tip abstract de date și $T_\Sigma \in \mathcal{I}_{(S,\Sigma)}$.

  Un modul funcțional în **Maude** (care conține numai declarații de sorturi și operații) definește un astfel de tip abstract de date și construiește efectiv algebra T_Σ.

Subalgebre. Algebre libere

Subalgebre

(S, Σ) - signatură multisortată, A (S, Σ)-algebră

$B \subseteq A$ este parte stabilă a lui A dacă

- $A_\sigma \in B_s$ or. $\sigma : \to s$,
- $A_\sigma(b_1, \ldots, b_n) \in B_s$ or. $\sigma : s_1 \cdots s_n \to s$, or. $(b_1, \ldots, b_n) \in B_{s_1} \times \cdots \times B_{s_n}$.

Daca B este parte stabila a lui A atunci $(B_S = B, B_\Sigma)$ este (S, Σ)-subalgebră a lui (A_S, A_Σ), unde

- $B_\sigma := A_\sigma$ or. $\sigma : \to s$,
- $B_\sigma(b_1, \ldots, b_n) := A_\sigma(b_1, \ldots, b_n)$ or. $\sigma : s_1 \cdots s_n \to s$, or. $(b_1, \ldots, b_n) \in B_{s_1} \times \cdots \times B_{s_n}$.

Exemple

- *BOOL*-algebra B:
  $B_{bool} := \mathcal{P}(\mathbb{N})$
  $B_T := \mathbb{N}$, $B_F := \emptyset$, $B_\neg(X) := \mathbb{N} \setminus X$,
  $B_\vee(X, Y) := X \cup Y$, $B_\wedge(X, Y) := X \cap Y$
- $B_1 = \{\emptyset, \mathbb{N}\} \cup \{\{n\} | n \in \mathbb{N}\}$
  nu este subalgebră.
- $B_2 = \{\emptyset, \mathbb{N}\} \cup \{\{n\}, \mathbb{N} \setminus \{n\}\}$
  este subalgebră ($n \in \mathbb{N}$ fixat).

Exemple

- *AUTOMAT*-algebra A
  $A_{intrare} = \{x, y\}$, $A_{stare} = \{s0, s1\}$, $A_{iesire} := \{T, F\}$
  $A_{s0} := s0$, $A_g(s0) := F$, $A_g(s1) := T$,
  $A_f(x, s0) := s0$, $A_f(y, s0) := s1$,
  $A_f(x, s1) := s0$, $A_f(y, s1) := s1$
- $P = \{P_{intrare} := \{x\}, P_{stare} := \{s0\}, P_{iesire} := \{F\}\}$
  este subalgebră A

Mulțimi de generatori

(S, Σ) signatură multisortată,
A-algebră, X mulțime, $X \subseteq A$

- Algebra generată de X în A este *cea mai mică* ($\subseteq$) subalgebră B a lui A cu $X \subseteq B$. Vom nota $B = \overline{X}$
- $\mathcal{F} := \{B \subseteq A \mid B \text{ subalgebră}, \ X \subseteq B\}$
  - $A \in \mathcal{F}$, deci $\mathcal{F} \neq \emptyset$
  - $\{B_i\}_{i \in I} \subseteq \mathcal{F}$ implică $\bigcap_{i \in I} B_i \in \mathcal{F}$
  - $\overline{X} = \bigcap\{B \mid B \in \mathcal{F}\}$
- Spunem că A este generată de X dacă $X \subseteq A$ și $\overline{X} = A$. În acest caz X este mulțime de generatori pentru A.

Construcția algebrei generate

(S, Σ) signatură multisortată,
A-algebră, X mulțime, $X \subseteq A$

- Construim un șir de mulțimi S-sortate $(X_n)_n$ astfel:
  - $X_0 := X$,
  - $X_{n+1,s} := X_{n,s} \cup \{A_\sigma \mid \sigma :\to s\} \cup \{A_\sigma(a_1, \ldots, a_n) \mid \sigma : s_1 \ldots s_n \to s, (a_1, \ldots, a_n) \in X_{n,s_1} \times \cdots \times X_{n,s_n}\}$.
- Propoziția 5. $\overline{X} = \bigcup_n X_n$.
  Dem: va fi adăugată!

Proprietăți

$h : A \to B$ funcție
$h^{-1} : \mathcal{P}(B) \to \mathcal{P}(A)$, $h^{-1}(Y) := \{a \in A | h(a) \in Y\}$
imaginea inversă

Propoziția 6.
Fie $h : A \to B$ morfism.

- (a) dc. $C \subseteq A$ subalgebră at. $h(C) \subseteq B$ subalgebră,
- (b) dc. $D \subseteq B$ subalgebră at. $h^{-1}(D) \subseteq B$ subalgebră,
- (c) $h(\overline{X}) = \overline{h(X)}$ pentru $X \subseteq A$,
- (d) $\overline{h^{-1}(Y)} \subseteq h^{-1}(\overline{Y})$ pentru $Y \subseteq B$.

Dem. (a), (b), (d) exercitii.

Continuare

(c) $h(\overline{X}) = \overline{h(X)}$ pentru $X \subseteq A$,.

$\overline{X} \subseteq A$ subalgebră $\Rightarrow h(\overline{X}) \subseteq B$ subalgebră

$h(\overline{X}) \subseteq B$ subalgebră, $h(X) \subseteq h(\overline{X}) \Rightarrow \overline{h(X)} \subseteq h(\overline{X})$

$h^{-1}(\overline{h(X)}) \subseteq A$ subalgebră, $X \subseteq h^{-1}(\overline{h(X)}) \Rightarrow \overline{X} \subseteq h^{-1}(\overline{h(X)}) \Rightarrow$
$h(\overline{X}) \subseteq h(h^{-1}(\overline{h(X)})) \subseteq \overline{h(X)}$

$\overline{h(X)} \subseteq h(\overline{X})$ csi $h(\overline{X}) \subseteq \overline{h(X)} \Rightarrow h(\overline{X}) = \overline{h(X)}$ $\square$

Algebre libere

(S, Σ) signatură multisortată, X mulțime de variabile

O algebră A este liber generată de X dacă

- $X \subseteq A$,
- oricare ar fi B o algebră și $f : X \to B$ o funcție există un unic morfism $\tilde{f} : A \to B$ cu $\tilde{f}_s(x) = f_s(x)$ oricare $x \in X_s$, $s \in S$.

Observații

* Dacă A_1 și A_2 sunt liber generate de X, atunci $A_1 \simeq A_2$.
* Dacă $X \subseteq A$ și A este liber generată de X, atunci $\overline{X} = A$. Invers nu este întotdeauna a adevă rat.

- T_Σ este liber generată de mulțimea $\emptyset$.

Mulțimi de generatori liberi

(S, Σ) signatură multisortată,
A-algebră, X mulțime de variabile, $X \subseteq A$

Dacă A este liber generată de X, atunci $\overline{X} = A$.
Spunem că X este mulțime de generatori liberi pentru A.

- $NAT = (S = \{s\}, \Sigma)$, $\Sigma = \{0 :\to nat, succ : nat \to nat\}$
  $A_{nat} := \mathbb{N}$, $A_0 := 0$, $A_{succ}(x) := x + 1$
  - A este liber generată de $\emptyset$
  - $\{1\}$ este mulțime de generatori pentru A
  - $\{1\}$ nu este mulțime de generatori liberi pentru A

Exemple

- $STIVA = (S = \{elem, stiva\}, \Sigma)$
  $\Sigma = \{0 :\to elem, empty :\to stiva, pop : stiva \to stiva,$
  $push : elem\ stiva \to stiva, top : stiva \to elem\}$
- $STIVA$-algebra A: $A_{elem} := \mathbb{N}$, $A_{stiva} := \mathbb{N}^*$
  $A_0 := 0$, $A_{empty} := \lambda$, $A_{push}(n, n_1 \cdots n_k) := n\, n_1 \cdots n_k$,
  $A_{pop}(\lambda) = A_{pop}(n) := \lambda$,
  $A_{pop}(n_1 n_2 \cdots n_k) := n_2 \cdots n_k$ pt. $k \geq 2$,
  $A_{top}(\lambda) := 0$, $A_{top}(n_1 \cdots n_k) := n_1$ pt. $k \geq 1$.
- Dacă $P := \overline{\emptyset}$ atunci $P_{elem} = \{0\}$ și $P_{stiva} = \{0\}^*$.
- A este liber generată de X, unde $X_{elem} := \mathbb{N}$ și $X_{stiva} := \emptyset$.

Semantica termenilor

Mulțime de variabile

(S, Σ) signatură multisortată

$|\Sigma| := \bigcup_{w,s} \Sigma_{w,s}$

$|X| := \bigcup_{s \in S} X_s$ dacă X mulțime S-sortată

O mulțime de variabile este o mulțime S-sortată X a.î.:

- $X_s \cap X_{s'} = \emptyset$ or. $s \neq s'$
- $|X| \cap |\Sigma| = \emptyset$

simbolurile de variabile sunt distincte între ele și sunt distincte de simbolurile de operații din Σ

Termeni cu variabile din X

(S, Σ) signatură, X mulțime de variabile

Mulțimea S-sortată termenilor cu variabile din X, $T_\Sigma(X)$, este *cea mai mică* mulțime de *șiruri finite* peste alfabetul

$$L = \bigcup_{s \in S} X_s \cup \bigcup_{w,s} \Sigma_{w,s} \cup \{\mathbf{(},\mathbf{)}\} \cup \{\mathbf{,}\}$$

care verifică următoarele proprietăți:

- (T1) $X_s \subseteq T_\Sigma(X)_s$
- (T2) dc. $\sigma :\to s$, at. $\boldsymbol{\sigma} \in T_\Sigma(X)_s$,
- (T3) dc. $\sigma : s_1 \cdots s_n \to s$ și $t_i \in T_\Sigma(X)_{s_i}$ or. $i = 1, \ldots, n$ at. $\boldsymbol{\sigma}\mathbf{(t_1, \ldots, t_n)} \in T_\Sigma(X)_s$.

Algebra de termeni $T_\Sigma(X)$

(S, Σ) signatură, X mulțime de variabile
Mulțimea termenilor $T_\Sigma(X) = \{T_\Sigma(X)_s\}_{s \in S}$ este (S, Σ)-algebră astfel:

- pt. $\sigma :\to s$, operația corespunzătoare este $T_\sigma := \boldsymbol{\sigma}$
- pt. $\sigma : s_1 \cdots s_n \to s$, operația corespunzătoare este
  $T_\sigma : T_\Sigma(X)_{s_1} \times \cdots \times T_\Sigma(X)_{s_n} \to T_\Sigma(X)_s$
  $T_\sigma(t_1, \ldots, t_n) := \boldsymbol{\sigma}\mathbf{(t_1, \ldots, t_n)}$
  or. $t_1 \in T_\Sigma(X)_{s_1}, \ldots, t_n \in T_\Sigma(X)_{s_n}$

- $T_\Sigma(X)$ algebra termenilor cu variabile din X
- vom nota $\boldsymbol{\sigma}\mathbf{(t_1, \ldots, t_n)} := \sigma(t_1, \ldots, t_n)$

Semantica termenilor

(S, Σ) signatură, X mulțime de variabile

Teorema 7 (Evaluarea termenilor în algebre).

Fie A o (S, Σ)-algebră. Orice funcție $\mathbf{e} : X \to A$ (evaluare, atribuire, interpretare)se extinde la un unic (S, Σ)-morfism $\tilde{\mathbf{e}} : T_\Sigma(X) \to A$.

Dem.

- Definim $\tilde{\mathbf{e}}(t)$ prin inducție pe termeni:
  - or. $x \in X_s$, $\tilde{\mathbf{e}}_s(x) := \mathbf{e}_s(x)$,
  - or. $\sigma :\to s$, $\tilde{\mathbf{e}}_s(\sigma) := A_\sigma$
  - or. $\sigma : s_1 \cdots s_n \to s$, or. $t_1 \in T_\Sigma(X)_{s_1}$, ..., $t_n \in T_\Sigma(X)_{s_n}$ $\tilde{\mathbf{e}}_s(\sigma(t_1, \ldots, t_n)) := A_\sigma(\tilde{\mathbf{e}}_{s_1}(t_1), \ldots, \tilde{\mathbf{e}}_{s_n}(t_n))$
- Demonstrăm că $\tilde{\mathbf{e}} : T_\Sigma(X) \to A$ este morfism.
- Dacă $f : T_\Sigma(X) \to A$ morfism și $f|_X = \mathbf{e}$ atunci se demonstrează prin inducție pe termeni că $f = \tilde{\mathbf{e}}$. $\square$

Semantica expresiilor aritmetice

- $NATEXP = (S = \{nat\}, \Sigma)$, $X = \{x, y\}$
- $\Sigma = \{0 :\to nat, s : nat \to nat,$
  $+ : nat\ nat \to nat, * : nat\ nat \to nat\}$
- $T_{NATEXP}(X) = \{0, x, y, s(0), s(x), s(y), s(s(0)), \ldots$
  $+(0,0), +(0,x), +(x,y), *(0, +(s(0), x)), *(s(y), s(s(x))), \ldots\}$
- $A = (\mathbb{Z}_4, 0, s, +, *)$ cu operațiile obișnuite
- $\mathbf{e} : \{x, y\} \to \mathbb{Z}_4$, $\mathbf{e}(x) := 1$, $\mathbf{e}(y) := 3$
- $\tilde{\mathbf{e}}(\mathbf{+(x,y)}) = A_+(\mathbf{e}(x), \mathbf{e}(y)) = 1 + 3 = 0\ (mod\ 4)$
  $\tilde{\mathbf{e}}(\mathbf{*(s(x), s(s(0)))}) = A_*(A_s(\mathbf{e}(x)), A_s(A_s(A_0))) =$
  $(1+1) * (0+1+1) = 2 * 2 = 0\ (mod\ 4)$

Exemple

- $STIVA = (S = \{elem, stiva\}, \Sigma)$
  $\Sigma = \{0 :\to elem, empty :\to stiva, pop : stiva \to stiva,$
  $push : elem\ stiva \to stiva, top : stiva \to elem\}$
- $STIVA$-algebra A: $A_{elem} := \mathbb{N}$, $A_{stiva} := \mathbb{N}^*$
  $A_0 := 0$, $A_{empty} := \lambda$, $A_{push}(n, n_1 \cdots n_k) := n\, n_1 \cdots n_k$,
  $A_{pop}(\lambda) = A_{pop}(n) := \lambda$,
  $A_{pop}(n_1 n_2 \cdots n_k) := n_2 \cdots n_k$ pt. $k \geq 2$,
  $A_{top}(\lambda) := 0$, $A_{top}(n_1 \cdots n_k) := n_1$ pt. $k \geq 1$.
- $STIVA$-algebra B: $B_{elem} := \{0\}$, $B_{stiva} := \mathbb{N}$
  $B_0 := 0$, $B_{empty} := 0$, $B_{push}(0, n) := n + 1$ or. n,
  $B_{pop}(0) := 0$, $B_{pop}(n) := n - 1$ pt. $n \geq 1$,
  $B_{top}(n) := 0$ or. n.

Exemplu

$X_{elem} := \{x, y\}$, $X_{stiva} := \{s\}$,
$t := \mathbf{push(x, push(y, s))} \in T_\Sigma(X)_{stiva}$

- $\mathbf{e} : X \to A$
  $\mathbf{e}(x) := 5$, $\mathbf{e}(y) := 3$, $\mathbf{e}(s) := 6\ 7$

  $\tilde{\mathbf{e}}(t) = A_{push}(\mathbf{e}(x), A_{push}(\mathbf{e}(y), \mathbf{e}(s))) = 5\ 3\ 6\ 7$
- $\mathbf{e} : X \to B$
  $\mathbf{e}(x) := 0$, $\mathbf{e}(y) := 0$, $\mathbf{e}(s) := 10$

  $\tilde{\mathbf{e}}(t) = B_{push}(\mathbf{e}(x), B_{push}(\mathbf{e}(y), \mathbf{e}(s))) =(10 + 1) + 1 = 12$

Consecințe

- $T_\Sigma(X)$ este (S,Σ)-algebră liber generată de X în clasa tuturor (S,Σ)-algebrelor.

  o expresie este un element al unei algebre libere

- $X = \emptyset$
  T_Σ este (S,Σ)-algebră inițială.
- $A = T_\Sigma(Y)$
  O substituție este o atribuire $\boldsymbol{\nu} : X \to T_\Sigma(Y)$.

  Orice substituție $\nu : X \to T_\Sigma(Y)$ se extinde la un unic morfism de (S,Σ)-algebre $\tilde{\nu} : T_\Sigma(X) \to T_\Sigma(Y)$.

Proprietăți

Propoziția 8.

Fie A o (S,Σ)-algebră. Dacă $f : T_\Sigma(X) \to A$ și $g : T_\Sigma(X) \to A$ sunt morfisme, atunci $g = f \Leftrightarrow g|_X = f|_X$

Dem. Presupunem că $g_s(x) = f_s(x)$ or. $s \in S$, $x \in X_s$.
Demonstram că $g = f$ prin inducție pe termeni

- pasul inițial: dc. $s \in S$ și $x \in X_s$ at. $g_s(x) = f_s(x)$ (ipoteză),
  dc. $\sigma :\to s \in \Sigma$, at. $g_s(\boldsymbol{\sigma}) = A_\sigma = f_s(\boldsymbol{\sigma})$,
- pasul de inducție: dc. $\sigma : s_1 \ldots s_n \to s \in \Sigma$ și
  $t_1 \in T_\Sigma(X)_{s_1}, \ldots, t_n \in T_\Sigma(X)_{s_n}$ a.î
  $g_{s_1}(t_1) = f_{s_1}(t_1), \cdots, g_{s_n}(t_n) = f_{s_n}(t_n)$ at.
  $$g_s(\boldsymbol{\sigma}(\mathbf{t_1,\ldots,t_n})) = A_\sigma(g_{s_1}(t_1),\cdots,g_{s_n}(t_n)) = A_\sigma(f_{s_1}(t_1),\cdots,f_{s_n}(t_n)) = f_s(\boldsymbol{\sigma}(\mathbf{t_1,\ldots,t_n}))$$

Conform principiului inductiei pe termeni,
$g_s(t) = f_s(t)$ or. $t \in T_\Sigma(X)_s$, adică $g = f$. $\square$

Proprietăți

Propoziția 9.

$X \simeq Y \Leftrightarrow T_\Sigma(X) \simeq T_\Sigma(Y)$

Dem. Fie $\mathbf{u} : X \to Y$ o functie bijectiva. Din Teorema 7 rezultă că

- există un unic morfism $f : T_\Sigma(X) \to T_\Sigma(Y)$ aî.
  $f_s(x) = \mathbf{u}_s(x)$ or. $s \in S$, $x \in X_s$,
- există un unic morfism $g : T_\Sigma(Y) \to T_\Sigma(X)$ aî.
  $g_s(y) = \mathbf{u}_s^{-1}(y)$ or. $s \in S$, $y \in X_s$.

Observăm că $(f;g)_s(x) = g_s(f_s(x)) = x$ or. $s \in S$, $x \in X$.
Aplicand Propozitia 6, obținem că $f;g = 1_{T_\Sigma(X)}$.

Similar, $g;f = 1_{T_\Sigma(Y)}$, deci $T_\Sigma(X) \simeq T_\Sigma(Y)$. $\square$

Proprietăți

Propoziția 10.

Fie $h : A \to B$ morfism *surjectiv*.
Or. $f : T_\Sigma(X) \to B$ morfism,
ex. $g : T_\Sigma(X) \to A$ a. î. $g;h = f$

$$\begin{tikzcd} A \arrow[r, "h"] & B \\ & T_\Sigma(X) \arrow[u, "f"'] \arrow[ul, dotted, "g"] \end{tikzcd}$$

Dem. Dacă $f : T_\Sigma(X) \to B$ morfism, atunci
or. $s \in S$, $x \in X_s$ ex. $a \in A_s$ a.î. $h_s(a) = f_s(x)$.
Oricare $s \in S$, $x \in X_s$, alegem $a \in A_s$ a.î. $h_s(a) = f_s(x)$ și definim $\mathbf{e}_s(x) := a$. Atunci $e : X \to A$ este o evaluare și .
$(\tilde{\mathbf{e}}; h)_s(x) = f_s(x)$ r. $s \in S$, $x \in X_s$.
Din Teorema 7, $\tilde{\mathbf{e}}; h = f$, deci $g := \tilde{\mathbf{e}}$. $\square$

Definirea operațiilor derivate

A o Σ-algebră, $t \in T_\Sigma(X)$

Definim funcția termen $A_t : A^n \to A$ prin

$A_t(a_1, \ldots, a_n) := \tilde{\mathbf{e}}(t)$, unde $\mathbf{e}(x_i) := a_i$ or. $i = 1, \ldots, n$

A_t este operație derivată pe A

- $\Sigma = \{0, 1, \vee, \wedge, ^-\}$ signatura algebrelor Boole, $X = \{x, y\}$, $t = \overline{x} \vee y \in T_\Sigma(X)$
  Dacă B este o algebră Boole, atunci $B_t(b_1, b_2) = b_1 \to b_2$ oricare b_1, $b_2 \in B$.

Semantica instrucțiunii de atribuire x:=e

- X este mulțimea variabilelor, $x \in X$, $t \in T_\Sigma(X)$
- D este Σ-algebra datelor
- o stare a memoriei este o funcție $\mathbf{e} : X \to D$
- semantica unei instrucțiuni descrie modul în care instrucțiunea modifică starile memoriei
- $Mem := \{\mathbf{e} : X \to D \mid \mathbf{e}$ funcție$\}$
  $Sem(x := t) : Mem \to Mem$
  $Sem(x := t)(\mathbf{e})(y) := \begin{cases} \tilde{\mathbf{e}}(t) & \text{dacă } y = x, \\ \mathbf{e}(y) & \text{dacă } y \neq x. \end{cases}$

Ecuații. Relația de satisfacere.

(S, Σ) signatură multisortată

- O (S, Σ)-ecuație este formată dintr-o mulțime de variabile X și din doi termeni t și t' de același sort din $T_\Sigma(X)$. Vom nota o ecuație prin
$$(\forall X)t \doteq_s t'.$$
- Spunem că o (S, Σ)-algebră A satisface o ecuație $(\forall X)t \doteq_s t'$ (A este model al ecuației) dacă $\tilde{\mathbf{e}}_s(t) = \tilde{\mathbf{e}}_s(t')$ pentru orice $\mathbf{e} : X \to A$. În acest caz, vom nota
$$A \models (\forall X)t \doteq_s t'.$$

$\doteq$ egalitate formală, $=$ egalitate efectivă

- $S = \{s, bool\}$, $\Sigma := \{T :\to bool, F :\to bool, g : s \to bool\}$
- $T_{\Sigma,s} = \emptyset$, $T_{\Sigma,bool} = \{T, F\}$
- $T_\Sigma \not\models (\forall\emptyset)T \doteq_{bool} F$
- $T_\Sigma \models (\forall X)T \doteq_{bool} F$, unde $X_s := \{x\}$, $X_{bool} := \emptyset$

O (S, Σ)-ecuație condiționată este notată prin
$$(\forall X)t \doteq_s t' \ if \ H.$$
și este formată din:

- o mulțime de variabile X,
- doi termeni de același sort t, $t' \in T_\Sigma(X)_s$.
- o mulțime H de ecuații $u \doteq_{s'} v$, cu $u,v \in T_\Sigma(X)_{s'}$.

În practică H este finită $H = \{u_1 \doteq_{s_1} v_1, \ldots, u_n \doteq_{s_n} v_n\}$. Ecuațiile din H sunt cuantificate cu X.

O ecuație $(\forall X)t \doteq_s t'$ este ecuație condiționată în care mulțimea condițiilor H este vidă $(\forall X)t \doteq_s t' \ if \ \emptyset$.

Ecuațiile condiționate

$\mathcal{A}$ (S,Σ)-algebră $\mathcal{A}$, $\gamma := (\forall X)t \doteq_s t'$ *if* H

$\mathcal{A}$ satisface γ ($\mathcal{A}$ este model al lui γ) dacă,

$$\tilde{\mathbf{e}}_{s'}(u) = \tilde{\mathbf{e}}_{s'}(v) \text{ or. } u \doteq_{s'} v \in H \Rightarrow \tilde{\mathbf{e}}_s(t) = \tilde{\mathbf{e}}_s(t').$$

pentru orice $\mathbf{e} : X \to A$.

Vom nota $\mathcal{A} \models (\forall X)t \doteq_s t'$ *if* H.

- $\mathcal{A} \models (\forall X)t \doteq_s t' \Leftrightarrow \mathcal{A} \models (\forall X)t \doteq_s t'$ *if* $\emptyset$

Exemplu

- $STIVA = (S,\Sigma)$, $X_{elem} := \{E\}$, $X_{stiva} := \{S, Q\}$
  $\gamma := (\forall X)top(S) \doteq_{elem} E$ *if* $\{S \doteq_{stiva} push(E,Q)\}$
- $STIVA$-algebra $\mathcal{A}$: $A_{elem} := \mathbb{N}$, $A_{stiva} := \mathbb{N}^*$
  $A_0 := 0$, $A_{empty} := \lambda$, $A_{push}(n, n_1\cdots n_k) := n\, n_1\cdots n_k$,
  $A_{pop}(\lambda) = A_{pop}(n) := \lambda$,
  $A_{pop}(n_1 n_2\cdots n_k) := n_2\cdots n_k$ pt. $k \geq 2$,
  $A_{top}(\lambda) := 0$, $A_{top}(n_1\cdots n_k) := n_1$ pt. $k \geq 1$.
- $STIVA$-algebra $\mathcal{B}$: $B_{elem} := \{0\}$, $B_{stiva} := \mathbb{N}$
  $B_0 := 0$, $B_{empty} := 0$, $B_{push}(0,n) := n+1$ or. n,
  $B_{pop}(0) := 0$, $B_{pop}(n) := n-1$ pt. $n \geq 1$, $B_{top}(n) := 0$ or. n.
- $\mathcal{A} \models \gamma$ și $\mathcal{B} \models \gamma$

Exemplu

$X_{elem} := \{E\}$, $X_{stiva} := \{S, Q\}$

$\mathcal{A} \models (\forall X)top(S) \doteq_{elem} E$ *if* $\{S \doteq_{stiva} push(E,Q)\}$

Fie $\mathbf{e} : X \to A$ evaluare a. î. $\tilde{\mathbf{e}}_{stiva}(S) = \tilde{\mathbf{e}}_{stiva}(push(E,Q))$.

Rezultă $\tilde{\mathbf{e}}_{stiva}(S) = A_{push}(\tilde{\mathbf{e}}_{elem}(E), \tilde{\mathbf{e}}_{stiva}(Q))$

Notam $n = \tilde{\mathbf{e}}_{elem}(E)$, $w := \tilde{\mathbf{e}}_{stiva}(S)$, $w' := \tilde{\mathbf{e}}_{stiva}(Q)$.

Rezulta $w = nw'$ și

$$\tilde{\mathbf{e}}_{elem}(top(S)) = A_{top}(\tilde{\mathbf{e}}_{stiva}(S)) = A_{top}(w) = A_{top}(nw') = n = \tilde{\mathbf{e}}_{elem}(E)$$

□

Exemplu

- $STIVA = (S,\Sigma)$, $X_{elem} := \{E\}$, $X_{stiva} := \{S, Q\}$
  $\gamma := (\forall X)top(S) \doteq_{elem} E$ *if* $\{S \doteq_{stiva} push(E,Q)\}$
- $STIVA$-algebra $\mathcal{C}$: $C_{elem} := \mathbb{N}$, $C_{stiva} := \mathbb{N}^*$
  $C_0 := 0$, $C_{empty} := \lambda$, $C_{push}(n, n_1\cdots n_k) := n_1\cdots n_k\, n$,
  $C_{pop}(\lambda) = C_{pop}(n) := \lambda$,
  $C_{pop}(n_1 n_2\cdots n_k) := n_2\cdots n_k$ pt. $k \geq 2$,
  $C_{top}(\lambda) := 0$, $C_{top}(n_1\cdots n_k) := n_1$ pt. $k \geq 1$.
- $\mathcal{C} \not\models \gamma$
  $\mathbf{e} : X \to C$ evaluarea definită prin
  $\mathbf{e}_{elem}(E) = 2$, $\mathbf{e}_{stiva}(Q) = 3\ 4$, $\mathbf{e}_{stiva}(S) = 3\ 4\ 2$
  Atunci $\tilde{\mathbf{e}}_{stiva}(S) = \tilde{\mathbf{e}}_{stiva}(push(E,Q))$,
  dar $\tilde{\mathbf{e}}_{elem}(E) = 2 \neq 3 = \tilde{\mathbf{e}}_{elem}(top(S))$.

Specificații algebrice

Γ mulțime de ecuații condiționate

O algebră A este Γ-algebră (A este model pentru Γ) dacă $A \models \gamma$ oricare $\gamma \in \Gamma$.

În acest caz, vom nota $A \models \Gamma$.

Vom nota cu $Alg(S, \Sigma, \Gamma)$ clasa Γ-algebrelor

$$Alg(S, \Sigma, \Gamma) := \{A \in Alg(S, \Sigma) \mid A \models \Gamma\}$$

Teorema 11.

Fie $A \simeq B$ (S, Σ)-algebre, $\gamma := (\forall X) t \doteq_s t'$ *if* H.

$$A \models \gamma \Leftrightarrow B \models \gamma$$

Dem. Notăm $\varphi : B \to A$ un izomorfism.
"$\Rightarrow$"Fie $\mathbf{e} : X \to B$ a.î. $\tilde{\mathbf{e}}_{s'}(u) = \tilde{\mathbf{e}}_{s'}(v)$ or. $u \doteq_{s'} v \in H$.
Definim $\mathbf{f} : X \to A$ prin $\mathbf{f} := \mathbf{e}; \varphi$
Atunci $\tilde{\mathbf{f}} = \tilde{\mathbf{e}}; \varphi$, deoarece $\tilde{\mathbf{f}}|_X = (\tilde{\mathbf{e}}; \varphi)|_X$.
$\tilde{\mathbf{f}}_{s'}(u) = \varphi_{s'}(\tilde{\mathbf{e}}_{s'}(u)) = \varphi_{s'}(\tilde{\mathbf{e}}_{s'}(v)) = \tilde{\mathbf{f}}_{s'}(v)$ or. $u \doteq_{s'} v \in H$
Din ipoteză rezultă $\tilde{\mathbf{f}}_s(t) = \tilde{\mathbf{f}}_s(t')$, i.e.
$\varphi_s(\tilde{\mathbf{e}}_s(t)) = \varphi_s(\tilde{\mathbf{e}}_s(t'))$.
φ este injectiv, deci $\tilde{\mathbf{e}}_s(t) = \tilde{\mathbf{e}}_s(t')$. $\square$

- O specificație este un triplet (S, Σ, Γ), unde (S, Σ) este o signatură multisortată și Γ este o mulțime de ecuații condiționate. Specificația (S, Σ, Γ) definește clasa modelelor $Alg(S, \Sigma, \Gamma)$, care reprezintă semantica ei.
- În **Maude** teoriile

  ```
  fth ...  endfth
  ```

  au ca semantică $Alg(S, \Sigma, \Gamma)$, unde S este mulțimea sorturilor, Σ este mulțimea simbolurilor de operații, Γ este mulțimea ecuațiilor definite în modul, iar fiecare ecuație
  `eq t = t'` și `ceq t = t' if H`
  este cuantificată de variabilele care apar în `t` și `t'`.

Specificații echivalente

```
fth GROUP is
sort Element .
op e : -> Element.
op  _+_  : Element Element -> Element [assoc] .
op -_  : Element -> Element  .
vars x y : Element .
eq e + x = x .
eq x + e = x .
eq (- x) + x = e .
eq x + (- x)  = e .
endfth
```


Consecințe semantice

(S, Σ, Γ) specificație, θ ecuație, Θ mulțime de ecuații

Ecuația θ este o consecință semantică a lui Γ dacă

$$A \models \Gamma \text{ implică } A \models \theta$$

pentru orice algebră A. În acest caz, vom nota $\Gamma \models \theta$.

$\Gamma \models \Theta \Leftrightarrow \Gamma \models \theta$ or. $\theta \in \Theta$

Teoria grupurilor

$(S = \{Element\}, \Sigma := \{e, -, +\}, \Gamma)$

$$\Gamma := \Big\{(\forall\{x,y,z\})(x+y)+z \doteq x+(y+z),\\ (\forall\{x\})e+x \doteq x,\\ (\forall\{x\})x+e \doteq x,\\ (\forall\{x\})(-x)+x \doteq e,\\ (\forall\{x\})x+(-x) \doteq e\Big\}$$

- $\theta_1 := (\forall\{x,y,z\})x \doteq y \text{ if } \{x+z \doteq y+z\}$,
- $\theta_2 := (\forall\{x,y\})x+y \doteq y+x$

$\Gamma \models \theta_1$, $\Gamma \not\models \theta_2$

Specificații echivalente

Două specificații (S, Σ, Γ_1) și (S, Σ, Γ_2) sunt echivalente dacă definesc aceeași clasă de modele:

$$A \models \Gamma_1 \Leftrightarrow A \models \Gamma_2$$

Observație

Fie Γ și Θ mulțimi de ecuații. Dacă $\Gamma \models \Theta$ atunci (S, Σ, Γ) și $(S, \Sigma, \Gamma \cup \Theta)$ sunt specificații echivalente.

Γ-algebra inițială

(S, Σ, Γ) specificație

$$\mathcal{I}_{\Sigma,\Gamma} := \{A \mid A \text{ inițială în } Alg(S, \Sigma, \Gamma)\}$$

este un tip abstract de date.

$\mathcal{I}_{\Sigma,\Gamma}$ reprezintă semantica unui modul funcțional în **Maude**

```
fmod ... endfm
```

unde S este mulțimea sorturilor, Σ este mulțimea simbolurilor de operații, Γ este mulțimea ecuațiilor definite în modul.
Fiecare ecuație
`eq t = t'` și `ceq t = t' if H`
este cuantificată de variabilele care apar în `t` și `t'`.

Specificație corecta

(S, Σ) o signatură multisortată, A (S, Σ)-algebră

O specificație (S, Σ, Γ) este adecvată pentru A dacă A este Γ-algebră inițială, i.e. $A \in \mathcal{I}_{\Sigma,\Gamma}$.

Exemplu

Determinați o specificație adecvată pentru $A = (\mathbb{Z}_4, 0, succ)$, unde $succ(x) := x + 1 \pmod 4$.

Continuare

```
fmod Z4 is
sort s .
op 0 : -> s .
op  succ  : s -> s  .
vars x : s .
eq  succ(succ(succ(succ(x)))) = x .
endfm
```

$S := \{s\}$, $\Sigma := \{0 :\to s, succ : s \to s\}$,
$\Gamma := \{(\forall x) succ(succ(succ(succ(x)))) \doteq x\}$

A este Γ-algebră inițială dacă:

- (a) $A \models \gamma$ or. $\gamma \in \Gamma$,
- (b) or. $B \models \Gamma$ ex. unic $f : A \to B$ morfism.

Continuare

$A = (\mathbb{Z}_4, 0, succ)$, unde $succ(x) := x + 1$

(a) $A \models (\forall x) succ(succ(succ(succ(x)))) \doteq x$

Dacă $X := \{x\}$ și $\mathbf{e} : X \to A$, atunci

$$\begin{aligned}\tilde{\mathbf{e}}(succ(succ(succ(succ(x))))) &= \\ A_{succ}(A_{succ}(A_{succ}(A_{succ}(\mathbf{e}(x))))) &= \mathbf{e}(x) + 4 \pmod 4 \\ &= \mathbf{e}(x) = \tilde{\mathbf{e}}(x)\end{aligned}$$

(b) Fie B o Γ-algebră.
Existența: definim $f : A \to B$ prin
$f(0) := B_0$, $f(x + 1) := B_{succ}(f(x))$ pt. $0 \leq x \leq 2$.

f morfism: $f(A_{succ}(x)) = f(x + 1) = B_{succ}(f(x))$ pt. $0 \leq x \leq 2$.
Ramâne de demontrat $f(A_{succ}(3)) = B_{succ}(f(3))$.

Continuare

(1) $f(A_{succ}(3)) = f(0) = B_0$

(2) $B_{succ}(f(3)) = B_{succ}(B_{succ}(f(2))) =$
$B_{succ}(B_{succ}(B_{succ}(B_{succ}(B_0)))$

Deoarece $B \models (\forall x) succ(succ(succ(succ(x)))) \doteq x$,
pentru $\mathbf{e}' : X \to B$, $\mathbf{e}'(x) := B_0$ obținem

$$B_{succ}(B_{succ}(B_{succ}(B_{succ}(B_0)))) = \tilde{\mathbf{e}}'(succ(succ(succ(succ(x))))) = \tilde{\mathbf{e}}'(x) = B_0$$

Din (1) și (2) rezultă $f(A_{succ}(3)) = B_{succ}(f(3))$.

Unicitatea: fie $g : A \to B$ un morfism;
dem. că $g(x) = f(x)$ prin inductie pt. $i \in \{0, 1, 2, 3\}$

$g(0) = g(A_0) = B_0 = f(0)$,
$g(x+1) = g(A_{succ}(x)) = B_{succ}(g(x)) = B_{succ}(f(x)) = f(A_{succ}(x)) = f(x+1)$. $\square$

Exercițiu

Determinați un model ințial pentru specificația:

```
fmod NSET is
sorts nat nset .
op 0 : -> nat .
op s : nat  -> nat .
op vid  : -> nset .
op ins : nat nset -> nset .
vars x y : nat .
var  A : nset .
eq  ins(x,ins(x,A)) = ins(x,A) .
eq  ins(x,ins(y,A)) = ins(y,ins(x,A)) .
endfm
```

Congruențe. Γ-algebra inițială

(S, Σ) signatură, A (S, Σ) algebră

- O relație S-sortată $\equiv = \{\equiv_s\}_{s \in S} \subseteq A \times A$ este congruența dacă:
  - $\equiv_s \subseteq A_s \times A_s$ echivalență or. $s \in S$,
  - $\equiv$ este compatibilă cu operațiile
    $a_i \equiv_{s_i} b_i$ or. $i = 1, \ldots, n \Rightarrow A_\sigma(a_1, \ldots, a_n) \equiv_s A_\sigma(b_1, \ldots, b_n)$
    or. $\sigma : s_1 \ldots s_n \to s$

(S, Σ) signatură, A (S, Σ) algebră

- NAT-algebra A:
  $A_{nat} := \mathbb{N}$, $A_0 := 0$, $A_{succ}(x) := x + 1$
  $n_1 \equiv_k n_2 \Leftrightarrow k | (n_1 - n_2)$ pentru $k \in \mathbb{N}$ fixat
  • $n_1 \equiv_k n_2 \Rightarrow A_{succ}(n_1) \equiv_k A_{succ}(n_2)$
- $AUTOMAT$-algebra B:
  $B_{intrare} = B_{stare} = B_{iesire} := \mathbb{N}$
  $B_{s0} := 0$, $B_f(m, n) := m + n$, $B_g(n) := n + 1$
  $\equiv$ este congruență pe B, unde $\equiv_{intrare} = \equiv_{stare} = \equiv_{iesire} := \equiv_k$

(S, Σ) signatură, A algebră, $\equiv$ congruență pe A

- $[a]_{\equiv_s} := \{a' \in A_s \mid a \equiv_s a'\}$ (clasa lui a)
- $A_s/_{\equiv_s} := \{[a]_{\equiv_s} \mid a \in A_s\}$ or. $s \in S$
- $A/_{\equiv} := \{A_s/_{\equiv_s}\}_{s \in S}$ este (S, Σ)-algebră
  - $(A/_{\equiv})_\sigma := [A_\sigma]$ or. $\sigma : \to s$,
  - $(A/_{\equiv})_\sigma([a_1]_{\equiv_{s_1}}, \ldots, [a_n]_{\equiv_{s_n}}) := [A_\sigma(a_1, \ldots, a_n)]_{\equiv_s}$
    or. $\sigma : s_1 \ldots s_n \to s$ și $(a_1, \ldots, a_n) \in A_{s_1} \times \cdots \times A_{s_n}$
- $[\cdot]_{\equiv} : A \to A/_{\equiv}$, $a \mapsto [a]_{\equiv_s}$ or. $a \in A_s$ este morfism.

$$[a]_{\equiv_s} = [b]_{\equiv_s} \Leftrightarrow a \equiv_s b$$

Exemple

- *STIVA*-algebra A: $A_{elem} := \mathbb{N}$, $A_{stiva} := \mathbb{N}^*$
  $A_0 := 0$, $A_{empty} := \lambda$, $A_{push}(n, n_1 \cdots n_k) := n\, n_1 \cdots n_k$,
  $A_{pop}(\lambda) = A_{pop}(n) := \lambda$,
  $A_{pop}(n_1 n_2 \cdots n_k) := n_2 \cdots n_k$ pt. $k \geq 2$,
  $A_{top}(\lambda) := 0$, $A_{top}(n_1 \cdots n_k) := n_1$ pt. $k \geq 1$.
- $\equiv_{elem} := \mathbb{N} \times \mathbb{N}$,
  $\equiv_{stiva} := \{(w, w') \mid (w, w') \in \mathbb{N}^* \times \mathbb{N}^*, |w| = |w'|\}$
  $\equiv = \{\equiv_{elem}, \equiv_{stiva}\}$ congruență
- $A/_{\equiv} \simeq B$, unde
  STIVA-algebra B: $B_{elem} := \{0\}$, $B_{stiva} := \mathbb{N}$
  $B_0 := 0$, $B_{empty} := 0$, $B_{push}(0, n) := n + 1$ or. n,
  $B_{pop}(0) := 0$, $B_{pop}(n) := n - 1$ pt. $n \geq 1$, $B_{top}(n) := 0$ or. n.

$Ker(f)$

(S, Σ) signatură, A și B algebre, $f : A \to B$ morfism

$Ker(f) = \{Ker(f_s)\}_{s \in S}$, unde
$Ker(f_s) := \{(a, a') \in A_s \times A_s \mid f_s(a) = f_s(a')\}$ or. $s \in S$

Propoziția 12

- (a) $Ker(f)$ este congruență pe A.
- (b) Dacă $\equiv$ congruență pe A, atunci $Ker([\cdot]_{\equiv}) = \equiv$.

Dem. exercițiu

Proprietatea de universalitate

(S, Σ) signatură, A algebră, $\equiv$ congruență pe A

Teorema 13 (Proprietatea de universalitate a algebrei cât)

Oricare ar fi B o algebră și $h : A \to B$ un morfism a.î. $\equiv \subseteq Ker(h)$ există un unic morfism $\overline{h} : A/_{\equiv} \to B$ a.î. $[\cdot]_{\equiv}; \overline{h} = h$.

Dem. $h : A \to B$, $\equiv \subseteq Ker(h)$

$$\begin{tikzcd} A \arrow[r, "{[\cdot]_{\equiv}}"] \arrow[d, "h"'] & A/_{\equiv} \arrow[dl, dotted, "\overline{h}"] \\ B & \end{tikzcd}$$

Existența: definim $\overline{h}_s([a]_{\equiv_s}) := h_s(a)$
or. $a \in A_s$

Continuare

$\overline{h}$ este bine definit: $[a_1]_{\equiv_s} = [a_2]_{\equiv_s} \Rightarrow h_s(a_1) = h_s(a_2)$

$[a_1]_{\equiv_s} = [a_2]_{\equiv_s} \Rightarrow (a_1, a_2) \in \equiv_s \subseteq Ker(h_s) \Rightarrow h_s(a_1) = h_s(a_2)$

$\overline{h}$ morfism:

- dc. $\sigma : \to s \in \Sigma$, at.
  $\overline{h}_s(A/_{e}quiv_{\sigma}) = \overline{h}_s([A_\sigma]_{\equiv}) = h_s(A_\sigma) = B_\sigma$,
- dc. $\sigma : s_1 \ldots s_n \to s \in \Sigma$ și
  $a_1 \in A_{s_1}, \ldots, a_n \in A_{s_n}$ at.
  $$\overline{h}_s(A_{\equiv\sigma}([a_1]_{\equiv}, \ldots, [a_n]_{\equiv})) = \overline{h}_s([A_\sigma(a_1, \ldots, a_n)]_{\equiv}) = h_s(A_\sigma(a_1, \ldots, a_n)) == B_\sigma(h_{s_1}(a_1), \cdots, h_{s_n}(a_n)) = B_\sigma(\overline{h}_{s_1}([a_1]_{\equiv}), \cdots, \overline{h}_{s_n}([a_n]_{\equiv}))$$

Unicitatea: fie $g : A/_{\equiv} \to B$ a.î. $[\cdot]_{\equiv}; g = h$

$g_s([a]_s) = h_s(a) = \overline{h}_s([a]_{\equiv_s})$ or. $a \in A_s$. $\square$

Consecințe

Teorema 14 (Teoremă I de izomorfism)

Oricare $f : A \to B$ morfism, $A/_{Ker(f)} \simeq f(A)$.

Dem. exercițiu

Propoziția 15

Fie $\mathcal{K}$ o clasă de (S, Σ)-algebre. Dacă

$$\equiv_{\mathcal{K}} := \bigcap \{Ker(h) \mid h : T_\Sigma \to B \in \mathcal{K},\ h \text{ morfism}\},$$

atunci următoarele proprietăți sunt adevărate:

- $\equiv_{\mathcal{K}}$ este congruență pe T_Σ,
- or. $B \in \mathcal{K}$ ex. unic morfism $\overline{h} : T_\Sigma/_{\equiv_{\mathcal{K}}} \to B$.

Dem. Intersecția unei familii arbitrare de congruențe este congruență (exercițiu).

Continuare

Fie $B \in \mathcal{K}$ si $h : T_\Sigma \to B$ unicul morfism.

Existența: deoarece $\equiv_{\mathcal{K}} \subseteq Ker(h)$, din Proprietatea de universalitate a algebrei cât ex. unic morfism $\overline{h} : T_\Sigma/_{\equiv_{\mathcal{K}}} \to B$ a.î. $[\cdot]_{\equiv_{\mathcal{K}}} ; \overline{h} = h$.

$$\begin{tikzcd} T_\Sigma \arrow[r, "{[\cdot]_{\equiv_{\mathcal{K}}}}"] \arrow[d, "h"'] & T_\Sigma/_{\equiv_{\mathcal{K}}} \arrow[dl, dotted, "\overline{h}"] \\ B & \end{tikzcd}$$

Unicitatea: fie $g : T_\Sigma/_{\equiv_{\mathcal{K}}} \to B$ un morfism; atunci $[\cdot]_{\equiv_{\mathcal{K}}} ; g : T_\Sigma \to B$, deci $[\cdot]_{\equiv_{\mathcal{K}}} ; g = h$, deci g verifică proprietatea care-l definește în mod unic pe $\overline{h}$; rezultă $g = \overline{h}$. □

Γ-algebre

(S, Σ, Γ) specificație,
A algebră și $\equiv$ o congruență pe A

Spunem că $\equiv$ este închisă la substituție dacă:
CS(Γ,A)

$$\boxed{\begin{array}{l} \text{or. } (\forall X) t \doteq_s t' \textit{ if } H \in \Gamma, \text{ or. } \mathbf{e} : X \to A \\ \tilde{\mathbf{e}}_{s'}(u) \equiv_{s'} \tilde{\mathbf{e}}_{s'}(v) \text{ or. } u \doteq_{s'} v \in H \Rightarrow \tilde{\mathbf{e}}_s(t) \equiv_s \tilde{\mathbf{e}}_s(t'). \end{array}}$$

Propoziția 16

Dacă $\equiv$ este o congruență pe A închisă la substituție atunci $A/_{\equiv} \models \Gamma$.

Continuare

Dem. Fie $(\forall X) t \doteq_s t'$ *if* $H \in \Gamma$ și $f : T_\Sigma(X) \to A/_{\equiv}$ a.î. $f_{s'}(u) = f_{s'}(v)$ or. $u \doteq_{s'} v \in H$.

$[\cdot]_{\equiv} : A \to A/_{\equiv}$ morfism *surjectiv*,
$f : T_\Sigma(X) \to A/_{\equiv}$ morfism

Din Propoziția 10, ex. $\tilde{\mathbf{e}} : T_\Sigma(X) \to A$ a. î. $\tilde{\mathbf{e}} ; [\cdot]_{\equiv} = f$

$$\begin{tikzcd} A \arrow[r, "{[\cdot]_{\equiv}}"] & A/_{\equiv} \\ & T_\Sigma(X) \arrow[ul, dotted, "\tilde{\mathbf{e}}"] \arrow[u, "f"'] \end{tikzcd}$$

Atunci $\tilde{\mathbf{e}}_{s'}(u) \equiv_{s'} \tilde{\mathbf{e}}_{s'}(v)$ or. $u \doteq_{s'} v \in H$ și, deoarece $\equiv$ este închisă la substituție, obținem $\tilde{\mathbf{e}}_s(t) \equiv_s \tilde{\mathbf{e}}_s(t')$, adică $f_s(t) = f_s(t')$.
□

Echivalența semantică

(S, Σ, Γ) specificație, A o (S, Σ)-algebră

$\equiv_{\Gamma,A} := \bigcap\{Ker(h) \mid h : A \to B \text{ morfism } \; B \models \Gamma\}$
echivalenă semantică pe A

- $A = T_\Sigma$ notăm $\equiv_\Gamma := \equiv_{\Gamma, T_\Sigma}$.

Propoziția 17

$\equiv_{\Gamma,A}$ este o congruență pe A închisă la substituție.

Continuare

Dem. Notăm $\equiv := \equiv_{\Gamma,A}$

Fie $(\forall X)t \doteq_s t'$ *if* $H \in \Gamma$ și $\mathbf{e} : X \to A$ a.î.
$\tilde{\mathbf{e}}_{s'}(u) \equiv_{s'} \tilde{\mathbf{e}}_{s'}(v)$ or. $u \doteq_{s'} v \in H$.
Rezultă $(\tilde{\mathbf{e}}_{s'}(u), \tilde{\mathbf{e}}_{s'}(v)) \in \equiv \subseteq Ker(h)$ or. $h : A \to B \models \Gamma$, deci
$h(\tilde{\mathbf{e}}_{s'}(u)) = h(\tilde{\mathbf{e}}_{s'}(v))$ or. $u \doteq_{s'} v \in H$.
Fie $B \models \Gamma$; $h : A \to B$. Atunci $\tilde{\mathbf{e}}; h : T_\Sigma(X) \to B$ și
$h(\tilde{\mathbf{e}}_{s'}(u)) = h(\tilde{\mathbf{e}}_{s'}(v))$ or. $u \doteq_{s'} v \in H$, deci $h(\tilde{\mathbf{e}}_s(t)) = h(\tilde{\mathbf{e}}_s(t'))$.
Rezultă $(\tilde{\mathbf{e}}_s(t), \tilde{\mathbf{e}}_s(t')) \in Ker(h)$ or. $h : A \to B \models \Gamma$, deci
$(\tilde{\mathbf{e}}_s(t), \tilde{\mathbf{e}}_s(t')) \in \equiv$, i.e. $\tilde{\mathbf{e}}_s(t) \equiv_s \tilde{\mathbf{e}}_s(t')$. $\square$

Γ-algebra inițială

(S, Σ, Γ) specificație

- Definim pe T_Σ congruența semantică
  $\equiv_\Gamma := \bigcap\{Ker(f) \mid f : T_\Sigma \to A, \; A \models \Gamma\}$,
  care este închisă la substituție cf. Propoziției 17.
- $T_\Sigma/_{\equiv_\Gamma} \models \Gamma$ cf. Propoziției 16.
- $\equiv_\Gamma = \equiv_\mathcal{K}$, unde $\mathcal{K} = Alg(S, \Sigma, \Gamma)$
  or. $B \models \Gamma$ ex. unic morfism $f : T_\Sigma/_{\equiv_\Gamma} \to B$ cf. Propoziției 15.

Teoremă 18.

$T_\Sigma/_{\equiv_\Gamma}$ este Γ-algebră inițială.

ADT

(S, Σ, Γ) specificație

$$\mathcal{I}_{\Sigma,\Gamma} := \{A \mid A \text{ inițială în } Alg(S, \Sigma, \Gamma)\}$$

este un tip abstract de date.

Un modul funcțional `fmod ... endfm` în **Maude** definește tipul abstract de date $\mathcal{I}_{\Sigma,\Gamma}$ și construiește efectiv algebra $T_\Sigma/_{\equiv_\Gamma}$, unde S este mulțimea sorturilor, Σ este mulțimea simbolurilor de operații, Γ este mulțimea ecuațiilor definite în modul. Fiecare ecuație
`eq t = t'` și `ceq t = t' if H`
este cuantificată de variabilele care apar în `t` și `t'`.

Consecințe

(S, Σ, Γ) specificație, A algebră, $h_A : T_\Sigma \to A$ unicul morfism

Teoremă

Sunt echivalente:

- A este Γ-algebră inițială
- A verifică următoarele proprietăți:
  - No Junk: h_A este surjectiv
  - No Confusion: $h_{A_s}(t_1) = h_{A_s}(t_2) \Leftrightarrow \Gamma \models (\forall\emptyset) t_1 \doteq_s t_2$
    or. t_1, $t_2 \in T_{\Sigma,s}$

Teorema constantelor
Demonstrații prin inducție

(S, Σ) signatura, X mulțime de variabile
Γ mulțime de ecuații condiționate

$$\Gamma \overset{?}{\models} (\forall X)t \doteq_s t' \text{ if } H$$

Propoziția 19

- (a) Dacă $\Gamma \models (\forall X)t \doteq_s t' \text{ if } H$ atunci
$$\Gamma \bigcup \{(\forall X)u \doteq_{s'} v \mid u \doteq_{s'} v \in H\} \models (\forall X)t \doteq_s t'.$$
- (b) Implicația inversă nu este adevărtă în general .
- (c) Dacă $\Gamma \bigcup \{(\forall \emptyset)u \doteq_{s'} v \mid u \doteq_{s'} v \in H\} \models (\forall \emptyset)t \doteq_s t'$ atunci $\Gamma \models (\forall \emptyset)t \doteq_s t' \text{ if } H$.

Dem. (a) Fie $A \models \Gamma \bigcup \{(\forall X)u \doteq_{s'} v \mid u \doteq_{s'} v \in H\}$ și $\mathbf{e} : X \to A$ o evaluare. Dar $A \models (\forall X)u \doteq_{s'} v$ or. $u \doteq_{s'} v \in H$, deci $\tilde{\mathbf{e}}_{s'}(u) = \tilde{\mathbf{e}}_{s'}(v)$ or. $u \doteq_{s'} v \in H$. Deoarece $A \models \Gamma$, din ipoteză, $A \models (\forall X)t \doteq_s t' \text{ if } H$. Deoarec $\mathbf{e} : X \to A$ este o evaluare a.î. $\tilde{\mathbf{e}}_{s'}(u) = \tilde{\mathbf{e}}_{s'}(v)$ or. $u \doteq_{s'} v \in H$, rezultă $\tilde{\mathbf{e}}_s(t) = \tilde{\mathbf{e}}_s(t')$.

(b) Implicația inversă nu este în general adevărată.
$S = \{s\}$, $\Sigma = \{0 :\to s\}$, $X = \{x, y\}$
$\{(\forall X)x \doteq 0\} \models (\forall X)y \doteq 0$ este adevărată, dar
$\not\models (\forall X)y \doteq 0 \text{ if } \{x \doteq 0\}$.

(b) Fie $A \models \Gamma$, $\tilde{\mathbf{e}} : T_\Sigma \to A$ unicul morfism. Dacă $\tilde{\mathbf{e}}_{s'}(u) = \tilde{\mathbf{e}}_{s'}(v)$ or. $u \doteq_{s'} v \in H$, atunci $A \models \Gamma \bigcup \{(\forall \emptyset)u \doteq_{s'} v \mid u \doteq_{s'} v \in H\}$. Din ipoteză, obținem $A \models (\forall \emptyset)t \doteq_s t'$, deci $\tilde{\mathbf{e}}_s(t) = \tilde{\mathbf{e}}_s(t')$. Am demonstrat ca dacă unicul morfism $\tilde{\mathbf{e}} : T_\Sigma \to A$ verifică $\tilde{\mathbf{e}}_{s'}(u) = \tilde{\mathbf{e}}_{s'}(v)$ or. $u \doteq_{s'} v \in H$, atunci $\tilde{\mathbf{e}}_s(t) = \tilde{\mathbf{e}}_s(t')$. $\square$

(S, Σ) signatură, X mulțime de variabile

- mulțimea X poate fi privită ca o signatură care are numai operații constante:
variabila $x \in X_s$ devine operația constantă $c_x :\to s$
- $\Sigma(X) := (\Sigma(X)_{w,s})_{w \in S^*, s \in S}$
  - $\Sigma(X)_{w,s} = \Sigma_{w,s}$ pentru $w \neq \lambda$
  - $\Sigma(X)_{\lambda,s} = \Sigma_{\lambda,s} \cup \{c_x | x \in X_s\}$

$\Sigma(X)$-algebre

A (S,Σ)-algebră, $\mathbf{a} : X \to A$ atribuire

- $(A, \mathbf{a})$ este $\Sigma(X)$-algebră
- $A_{c_x} := \mathbf{a}_s(x)$ oricare $x \in X_s$, $s \in S$
- orice $\Sigma(X)$-algebră poate fi construită astfel

$T_{\Sigma(X)} = \{T_{\Sigma(X),s}\}_{s \in S}$

- $T_{\Sigma(X),s} = T_{\Sigma}(X)_s$
- $T_{c_x} := x$ oricare $x \in X_s$, $s \in S$

Observație

$T_{\Sigma(X)}$ este $\Sigma(X)$-algebră inițială
$(A, \mathbf{a})$ $\Sigma(X)$-algebră
$\tilde{\mathbf{a}} : T_{\Sigma(X)} \to A$ unicul morfism de $\Sigma(X)$-algebre

Teorema constantelor I

(S,Σ) signatura, X mulțime de variabile

Teorema 20(Teorema constantelor I)

Sunt echivalente:

- (a) $A \models_{\Sigma} (\forall X)t \doteq_s t'$,
- (b) $(A, \mathbf{a}) \models_{\Sigma(X)} (\forall \emptyset)t \doteq_s t'$ or. $\mathbf{a} : X \to A$.

Dem. Ambele enunțuri sunt echivalente cu:
$\tilde{\mathbf{a}}_s(t) = \tilde{\mathbf{a}}_s(t')$ or. $\mathbf{a} : X \to A$ $\square$

Teorema constantelor II

(S,Σ) signatura, X mulțime de variabile
Γ o mulțime de Σ-ecuații necondiționate a.î.
nici o variabilă din X nu apare în Γ.

Teorema 21 (Teorema constantelor II).

Sunt echivalente:

- (a) $\Gamma \models_{\Sigma} (\forall X)t \doteq_s t'$ *if* H
- (b) $\Gamma \bigcup \{(\forall \emptyset)u \doteq_{s'} v \mid u \doteq_{s'} v \in H\} \models_{\Sigma(X)} (\forall \emptyset)t \doteq_s t'$

Dem. Deoarece nici o variabila din X nu apare în Γ, pentru orice (S,Σ)-algebră A sunt echivalnte:

- $A \models_{\Sigma} \Gamma$,
- $(A, \mathbf{a}) \models_{\Sigma(X)} \Gamma$ or. $\mathbf{a} : X \to A$.

Continuare

(a) $\Rightarrow$ (b) Fie $(A, \mathbf{a})$ o $\Sigma(X)$-algebră a.î.
$(A, \mathbf{a}) \models_{\Sigma(X)} \Gamma \bigcup \{(\forall \emptyset)u \doteq_{s'} v \mid u \doteq_{s'} v \in H\}$. Avem
$A \models \Gamma$, deoarece $(A, \mathbf{a}) \models_{\Sigma(X)} \Gamma$ și
$\tilde{\mathbf{a}}_{s'}(u) = \tilde{\mathbf{a}}_{s'}(v)$ or. $u \doteq_{s'} v \in H$, deoarece
$(A, \mathbf{a}) \models_{\Sigma(X)} \{(\forall \emptyset)u \doteq_{s'} v \mid u \doteq_{s'} v \in H\}$.
Din (a) rezultă că $\tilde{\mathbf{a}}_s(t) = \tilde{\mathbf{a}}_s(t')$, adică
$(A, \mathbf{a}) \models_{\Sigma(X)} (\forall \emptyset)t \doteq_s t'$.

(b) $\Rightarrow$ (a) Fie A o (S,Σ)-algebră a.î. $A \models_{\Sigma} \Gamma$ și
$\mathbf{a} : X \to A$ a.î. $\tilde{\mathbf{a}}_{s'}(u) = \tilde{\mathbf{a}}_{s'}(v)$ or. $u \doteq_{s'} v \in H$.Atunci
$(A, \mathbf{a}) \models_{\Sigma(X)} \Gamma$ și
$(A, \mathbf{a}) \models_{\Sigma(X)} \{(\forall \emptyset)u \doteq_{s'} v \mid u \doteq_{s'} v \in H\}$.
Din (b) rezultă că $(A, \mathbf{a}) \models_{\Sigma(X)} (\forall \emptyset)t \doteq_s t'$, adică
$\tilde{\mathbf{a}}_s(t) = \tilde{\mathbf{a}}_s(t')$. $\square$

Demonstrații prin inducție

```
fmod MYNAT is
  sort    MyNat .
  op 0 : -> MyNat .
  op s_ : MyNat -> MyNat .
  op _+_ : MyNat MyNat -> MyNat [comm assoc  prec 50 ] .
  op _*_ : MyNat MyNat -> MyNat [ comm assoc  prec 49 ] .
  vars  x y   : MyNat .
  eq x + 0 = x .
  eq x + s y = s(x + y) .
  eq x * 0 = 0 .
  eq x * s y = (x * y) + x .
  endfm
```


$cx * (cy + cz) = (cx * cy) + (cx * cz)$

```
fmod IND0 is
including  MYNAT .
ops  cy cz  : -> MyNat .
endfm
 *** pasul initial
 red 0 * (cy + cz) == 0 * cy + 0 * cz .
fmod IND is
including  IND0 .
op cx  : -> MyNat .
*** ipoteza inductie
eq cx * (cy + cz) = cx * cy + cx * cz .
endfm
*** pasul de inductie
red s cx * ( cy + cz) == s cx * cy + s cx * cz .
```


Demonstrații prin inducție

```
==========================================
fmod MYNAT
==========================================
fmod IND0
==========================================
reduce in IND0 : 0 * (cy + cz) == 0 * cy + 0 * cz .
rewrites: 5 in 10534559934ms cpu (0ms real)
result Bool: true
==========================================
fmod IND
==========================================
reduce in IND : s cx * (cy + cz) == cy * s cx + cz * s cx .
rewrites: 5 in 10534543934ms cpu (0ms real)
result Bool: true
```


Constructori $S = \{s\}$

. În exemplul anterior,
$\Sigma = \{0 :\to MyNat, s : MyNat \to MyNat,$
$+ : MyNat\ MyNat \to MyNat, * : MyNat\ MyNat \to MyNat\}$
dar în inductıa sructurală am folosit numai
$\Sigma_c = \{0 :\to MyNat, s : MyNat \to MyNat\}$.

Fie Σ o signatură (monosortată) și A o Σ-algebră . O subsignatură $Cons\Sigma \subseteq \Sigma$ se numește signatură de constructori pentru A dacă unicul morfism $h : T_{Cons\Sigma} \to A$ este surjectiv.

Dacă $P = (\Sigma, \Gamma)$ este o prezentare, notăm $T_P = T_\Sigma/_{\equiv_\Gamma}$. Atunci $Cons\Sigma$ se numește signatură de constructori pentru P dacă este signatură de constructori pentru T_P.

Constructori $S = \{s\}$

.

$P = (\Sigma, \Gamma)$ o prezentare, $Cons\Sigma$ signatură de constructori
$T_P := T_\Sigma/_{\equiv_\Gamma}$
$t \in T_\Sigma$, $[t] := [t]_{\equiv_\Gamma}$

*Teorema (Demonstrații prin inducție)**

Fie $\mathbf{Q}$ o proprietate a elementelor lui T_P a.î. următoarele condiții sunt satisfăcute:

- pasul inițial:
  $\mathbf{Q}([\boldsymbol{\sigma}]) = true$ or. $\sigma \in Cons\Sigma_{\lambda,s}$
- pasul de inducție:
  dacă $t_1, \ldots, t_n \in T_{\Sigma_s}$ și $\mathbf{Q}([t_1]) = \cdots = \mathbf{Q}([t_n]) = true$ atunci $\mathbf{Q}([\boldsymbol{\sigma}(\mathbf{t_1}, \ldots, \mathbf{t_n})]) = true$ or. $\sigma \in Cons\Sigma_{s^n,s}$.

Atunci $\mathbf{Q}([t]) = true$ oricare $[t] \in T_P$.

Sintaxa logicii ecuaționale

Regulile deducției ecuaționale

(S,Σ) signatură, X și Y mulțimi de variabile
Γ mulțime de ecuații (necondiționate)

R $\dfrac{}{(\forall X)t \doteq_s t}$

S $\dfrac{(\forall X)t_1 \doteq_s t_2}{(\forall X)t_2 \doteq_s t_1}$

T $\dfrac{(\forall X)t_1 \doteq_s t_2,\ (\forall X)t_2 \doteq_s t_3}{(\forall X)t_1 \doteq_s t_3}$

C_Σ $\dfrac{(\forall X)t_1 \doteq_{s_1} t_1', \ldots, (\forall X)t_n \doteq_{s_n} t_n'}{(\forall X)\sigma(t_1,\ldots,t_n) \doteq_s \sigma(t_1',\ldots,t_n')}$ $\quad \sigma : s_1 \cdots s_n \to s$

Sub_Γ $\dfrac{}{(\forall X)\tilde{\theta}(t) \doteq_s \tilde{\theta}(t')}$ unde
$(\forall Y)t \doteq_s t' \in \Gamma$, $\theta : Y \to T_\Sigma(X)$

Deducția sintactică

Fie Γ o mulțime de ecuații, numite axiome sau ipoteze. Spunem că ecuația $\epsilon := (\forall X)t \doteq_s t'$ se deduce sintactic din Γ dacă există o secvență de ecuații $\epsilon_1, \ldots, \epsilon_n$ a. î.

- $\epsilon_n = \epsilon$ și
- pentru orice $i \in \{1,\ldots,n\}$ $\epsilon_i \in \Gamma$ sau ϵ_i se obține din ecuațiile $\epsilon_1, \ldots, \epsilon_{i-1}$ aplicând una din regulile R, S, T, C_Σ, Sub_Γ.

În acest caz scriem $\Gamma \vdash \epsilon$ și spunem că ϵ este *deductibilă (sintactic), demonstrabilă, derivabilă* din Γ. Secvența $\epsilon_1, \ldots, \epsilon_n = \epsilon$ este o demonstrație pentru ϵ din ipotezele Γ.

Exemplu

$\Gamma = \{x + 0 \doteq x, x + succ(y) \doteq succ(x+y)\}$
$E \vdash 0 + succ(0) \doteq succ(0)$
(1) $x + succ(y) \doteq succ(x+y) \in E$
(2) $0 + succ(0) \doteq succ(0+0)$ $(1, Sub\{x, y \leftarrow 0\})$
(3) $x + 0 = x \in E$
(4) $0 + 0 \doteq 0$ $(3, Sub\{x \leftarrow 0\})$
(5) $succ(0+0) \doteq succ(0)$ $(4, C_\Sigma)$
(6) $0 + succ(0) \doteq succ(0)$ $(2, 5, T)$

$$\dfrac{\dfrac{x + succ(y) \doteq succ(x+y)}{0 + succ(0) \doteq succ(0+0)}(Sub\{x,y \leftarrow 0\}) \quad \dfrac{\dfrac{x + 0 \doteq x}{0 + 0 \doteq 0} Sub\{x \leftarrow 0\}}{succ(0+0) \doteq succ(0)}(C_\Sigma)}{0 + succ(0) \doteq succ(0)}(T)$$

Fie Γ o mulțime de ecuații condiționate.

$$\boxed{\textsf{Sub}_\Gamma \quad \frac{(\forall X)\theta(u_1) \doteq_{s_1} \theta(v_1), \ldots, (\forall X)\theta(u_n) \doteq_{s_n} \theta(v_n)}{(\forall X)\theta(t) \doteq_s \theta(t')}}$$

unde $(\forall Y)t \doteq_s t'$ *if* $H \in \Gamma$, $H = \{u_1 \doteq_{s_1} v_1, \ldots, u_n \doteq_{s_n} v_n\}$ și
$\theta : Y \to T_\Sigma(X)$ substituție

- Identificăm substituția $\theta : Y \to T_\Sigma(X)$ cu $\tilde{\theta} : T_\Sigma(Y) \to T_\Sigma(X)$.
- Dacă $H = \emptyset$ atunci $\boxed{\textsf{Sub}_\Gamma \quad \frac{}{(\forall X)\theta(t) \doteq_s \theta(t')}}$

$NATBOOL = (S, \Sigma)$, $S = \{n, b\}$, $\Sigma = \{T, F, 0, s, *, >\}$
$\Gamma = \{\gamma, \epsilon_1, \epsilon_2\}$
$\gamma := \forall\{x, y, z\} x \doteq_n y \text{ if } \{z * x \doteq_n z * y\, ,\ z > 0 \doteq_b T\}$
$\epsilon_1 := \forall\{a, c\} s(s(s(0))) * a \doteq_n s(s(s(0))) * c$,
$\epsilon_2 := \forall\{a, c\} s(s(s(0))) > 0 \doteq_b T$

$$\Gamma \vdash \forall\{a, c\} a \doteq_n c$$

$$\boxed{\textsf{Sub}_\Gamma \quad \frac{\epsilon_1, \epsilon_2}{a \doteq_s c}}$$

$$\gamma \in \Gamma,\ \theta_n(x) := a,\ \theta_n(y) := c,\ \theta_n(z) := s(s(s(0)))$$

Logică ecuațională
Corectitudine. Completitudine

(S, Σ) signatură, Xși Y mulțimi de variabile
Γ mulțime de ecuații (necondiționate)

R $\quad \dfrac{}{(\forall X)t \doteq_s t}$

S $\quad \dfrac{(\forall X)t_1 \doteq_s t_2}{(\forall X)t_2 \doteq_s t_1}$

T $\quad \dfrac{(\forall X)t_1 \doteq_s t_2,\ (\forall X)t_2 \doteq_s t_3}{(\forall X)t_1 \doteq_s t_3}$

$C_\Sigma \quad \dfrac{(\forall X)t_1 \doteq_{s_1} t'_1, \ldots, (\forall X)t_n \doteq_{s_n} t'_n}{(\forall X)\sigma(t_1, \ldots, t_n) \doteq_s \sigma(t'_1, \ldots, t'_n)} \quad \sigma : s_1 \cdots s_n \to s$

$Sub_\Gamma \quad \dfrac{(\forall X)\theta(u_1) \doteq_{s_1} \theta(v_1), \ldots, (\forall X)\theta(u_n) \doteq_{s_n} \theta(v_n)}{(\forall X)\tilde{\theta}(t) \doteq_s \tilde{\theta}(t')}$ unde

$(\forall Y)t \doteq_s t'$ *if* $\{u_1 \doteq_{s_1} v_1, \ldots, u_n \doteq_{s_n} v_n\} \in \Gamma$, $\theta : Y \to T_\Sigma(X)$

Γ mulțime de ecuații condiționate

O regula de deductie $\boxed{\dfrac{\epsilon_1, \ldots, \epsilon_n}{\epsilon}}$ este corectă dacă

$$\Gamma \models \epsilon_1, \ldots, \Gamma \models \epsilon_n \Rightarrow \Gamma \models \epsilon$$

Propoziția 22

Regulile deducției ecuaționale R, S, T, C_Σ, Sub_Γ sunt corecte.

- Demonstrăm că C_Σ este corectă.

Fie $\sigma : s_1 \cdots s_n \to s$ și presupunem că
$\Gamma \models (\forall X)t_1 \doteq_{s_1} t'_1$, ..., $\Gamma \models (\forall X)t_n \doteq_{s_n} t'_n$ (*)
Trebuie să arătăm că $\Gamma \models (\forall X)\sigma(t_1, \ldots, t_n) \doteq_s \sigma(t'_1, \ldots, t'_n)$.
Fie $A \models \Gamma$ și $f : T_\Sigma(X) \to A$ un morfism. Din (*),
$f_{s_1}(t_1) = f_{s_1}(t'_1)$, ..., $f_{s_n}(t_n) = f_{s_n}(t'_n)$, deci

$$f_s(\sigma(t_1, \ldots, t_n)) = A_\sigma(f_{s_1}(t_1), \ldots, f_{s_n}(t_n)) =$$
$$A_\sigma(f_{s_1}(t'_1), \ldots, f_{s_n}(t'_n)) = f_s(\sigma(t_1, \ldots, t_n)).$$

În consecință, $A \models (\forall X)\sigma(t_1, \ldots, t_n) \doteq_s \sigma(t'_1, \ldots, t'_n)$.
Deoarece A este o Γ-algebră arbitrară,
$\Gamma \models (\forall X)\sigma(t_1, \ldots, t_n) \doteq_s \sigma(t'_1, \ldots, t'_n)$. $\square$

Continuare

- Demonstrăm că Sub_Γ este corectă.

Fie $(\forall Y)t \doteq_s t'$ *if* $H \in \Gamma$, $H = \{u_1 \doteq_{s_1} v_1, \ldots, u_n \doteq_{s_n} v_n\}$ și $\theta : Y \to T_\Sigma(X)$ substituție a.î.
$\Gamma \models (\forall X)\theta(u_1) \doteq_{s_1} \theta(v_1)$, ..., $\Gamma \models (\forall X)\theta(u_n) \doteq_{s_n} \theta(v_n)$ (*)
Trebuie să arătăm că $\Gamma \models (\forall X)\theta(t) \doteq_s \theta(t')$.
Fie $A \models \Gamma$ și $f : T_\Sigma(X) \to A$ un morfism. Atunci $\tilde{\theta}; f : T_\Sigma(Y) \to A$ și, din (*),
$(\tilde{\theta}; f)_{s_1}(u_1) = (\tilde{\theta}; f)_{s_1}(v_1)$, ..., $(\tilde{\theta}; f)_{s_n}(u_n) = (\tilde{\theta}; f)_{s_n}(v_n)$.
Deoarece $A \models (\forall Y)t \doteq_s t'$ *if* $H \in \Gamma$, obținem

$$(\tilde{\theta}; f)_s(t) = (\tilde{\theta}; f)_s(t') \text{ i.e. } f_s(\tilde{\theta}(t)) = f_s(\tilde{\theta}(t')).$$

În consecință, $A \models (\forall X)\tilde{\theta}(t) \doteq_s \tilde{\theta}(t')$.
Deoarece A este o Γ-algebră arbitrară,
$\Gamma \models (\forall X)\tilde{\theta}(t) \doteq_s \tilde{\theta}(t')$. $\square$

Corectitudinea deducției ecuaționale

Γ mulțime de ecuații condiționate

Teorema 23 (Corectitudine)

$\Gamma \vdash (\forall X)t \doteq_s t' \Rightarrow \Gamma \models (\forall X)t \doteq_s t'$

Dem.: Fie $\boldsymbol{\epsilon_1}$, ..., $\boldsymbol{\epsilon_n} = (\forall X)t \doteq_s t'$ o demonstrație din ipotezele Γ. Demonstrăm $\Gamma \models \epsilon_i$ prin inducție după $i = 1, \ldots, n$.
Observăm că $\epsilon_1 \in \Gamma$ sau $\epsilon_1 = (\forall X)t \doteq_s t$, deci $\Gamma \models \epsilon_1$.
Presupunem că $\Gamma \models \boldsymbol{\epsilon_1}$, ..., $\Gamma \models \boldsymbol{\epsilon_{i-1}}$. Știm că $\boldsymbol{\epsilon_i}$ se obține din ecuațiile $\boldsymbol{\epsilon_1}$, ..., $\boldsymbol{\epsilon_{i-1}}$ aplicând una din regulile de deducție.Deoarece R, S, T, C_Σ, Sub_Γ sunt corecte, rezultă $\Gamma \models \epsilon_i$.
$\square$

Închiderea la reguli de deducție

(S, Σ) o signatură multisortată, X o mulțime de variabile

Reg $\boxed{\dfrac{(\forall X)t_1 \doteq_{s_1} t_1', \ldots, (\forall X)t_n \doteq_{s_n} t_n'}{(\forall X)t \doteq_s t'}}$

Relație binară $\sim \subseteq T_\Sigma(X) \times T_\Sigma(X)$ este închisă la **Reg** dacă

$$t_1 \sim_{s_1} t_1', \ldots, t_n \sim_{s_n} t_n' \Rightarrow t \sim_s t'$$

Propoziția 24

Sunt echivalente:

- (a) $\sim$ este congruență pe $T_\Sigma(X)$,
- (b) $\sim$ este închisă la R, S, T, C_Σ .

Continuare

Dem. (a)$\Rightarrow$(b) Presupunem că $\sim$ este congruență și demonstrăm închiderea la C_Σ.Fie $\sigma : s_1 \cdots s_n \to s$ și $t_1 \sim_{s_1} t_1'$, ..., $t_n \sim_{s_n} t_n'$.
Deoarece $\sim$ este congruență, $\sigma(t_1, \ldots, t_n) \sim_s \sigma(t_1', \ldots, t_n')$,
deci $\sim$ este închisă la C_Σ.

(b)$\Rightarrow$(a) Presupunem $\sim$ este echivalență închisă la C_Σ și demonstrăm că $\sim$ este compatibilă cu operațiile. Fie $\sigma : s_1 \cdots s_n \to s$ și $t_1 \sim_{s_1} t_1'$, ..., $t_n \sim_{s_n} t_n'$. Deoarece $\sim$ este închisă la C_Σ, $\sigma(t_1, \ldots, t_n) \sim_s \sigma(t_1', \ldots, t_n')$, deci $\sim$ este compatibilă cu operațiile. $\square$

Γ-algebre

(S, Σ, Γ) specificație, X o mulțime de variabile,
$\sim$ o congruență pe $T_\Sigma(X)$

$\mathrm{CS}(\Gamma, T_\Sigma(X))$

or. $(\forall Y)t \doteq_s t'$ *if* $H \in \Gamma$, or. $h : T_\Sigma(Y) \to T_\Sigma(X)$ morfism $h_{s'}(u) \sim_{s'} h_{s'}(v)$ or. $u \doteq_{s'} v \in H \Rightarrow h_s(t) \sim_s h_s(t')$.

Propoziția 25

Sunt echivalente:

- (a) $\sim$ verifică $\mathrm{CS}(\Gamma, T_\Sigma(X))$,
- (b) $\sim$ este închisă la Sub_Γ,
- (c) $T_\Sigma(X)/_\sim \models \Gamma$.

Echivalența sintactică

(S, Σ, Γ) specificație, X o mulțime de variabile

Pe $T_\Sigma(X)$ definim următoarea relație S-sortată

$$t {\sim_\Gamma}_s t' \Leftrightarrow \Gamma \vdash (\forall X)t \doteq_s t' \text{ oricare } s \in S.$$

este echivalența sintactică determinată de Γ.

Propoziția 26

$\sim_\Gamma$ este o congruență închisă la substituție.

Dem. Din definiția deducției sintactice $\vdash$, rezultă că $\sim_\Gamma$ este închisă la R, S, T, C_Σ, Sub_Γ. $\square$

Echivalența semantică

(S, Σ, Γ) specificație, A o (S, Σ)-algebră

$$\equiv_{\Gamma,A} := \bigcap\{Ker(h) \mid h : A \to B \text{ morfism } \ B \models \Gamma\}$$

este echivalența semantică pe A determinată de Γ.

Propoziția 27

$\equiv_{\Gamma,A}$ este cea mai mică congruență pe A închisă la substituție.

Dem. Din Propoziția 17, $\equiv_{\Gamma,A}$ este congruentă A închisă la substituție. Fie $\sim$ o altă congruență pe A închisă la substituție și $p : A \to A/_\sim$ surjecția canonică ($p(a) = [a]_\sim$ or. $a \in A$). Din Propoziția 16, $A/_\sim \models \Gamma$, deci $\sim = Ker(p) \subseteq \equiv_{\Gamma,A}$. $\square$

Completitudinea deducției ecuaționale

Γ mulțime de ecuații condiționate, $\epsilon = (\forall X)t \doteq_s t'$ ecuație

Teorema 28 (Completitudine)

$\Gamma \models (\forall X)t \doteq_s t' \Rightarrow \Gamma \vdash (\forall X)t \doteq_s t'$

Dem. Din Propoziția 26 și Propoziția 16, $T_\Sigma(X)/_{\sim_\Gamma}$ este o Γ-algebră, deci $T_\Sigma(X)/_{\sim_\Gamma} \models \epsilon$.Fie $p : T_\Sigma(X) \to T_\Sigma(X)/_{\sim_\Gamma}$ surjecția canonică, $p(t) := [t]$ oricare $t \in T_\Sigma(X)$, unde $[t]$ este clasa de echivalență a lui t determinată de $\sim_\Gamma$. Rezultă $p_s(t) = p_s(t')$, i.e. $[t] = [t']$.În consecință $t {\sim_\Gamma}_s t'$, adică $\Gamma \vdash (\forall X)t \doteq_s t'$. $\square$

Demonstrație scurtă: $\equiv_\Gamma \subseteq \sim_\Gamma$

Se aplică Propozițiile 26 și 27.

Teorema de completitudine

(S, Σ) signatura, X mulțime de variabile, $t, t' \in T_\Sigma(X)_s$

- $t \sim_\Gamma t' \Leftrightarrow \Gamma \vdash (\forall X) t \doteq_s t'$ (echiv. sintactică)
- $t \equiv_\Gamma t' \Leftrightarrow \Gamma \models (\forall X) t \doteq_s t'$ (echiv. semantică)
- Corectitudinea regulilor de deducție: $\sim_\Gamma \subseteq \equiv_\Gamma$
- Completitudinea regulilor de deducție: $\equiv_\Gamma \subseteq \sim_\Gamma$

Teorema 29 (Teorema de completitudine) $\equiv_\Gamma = \sim_\Gamma$

$$\Gamma \models (\forall X) t \doteq_s t' \Leftrightarrow \Gamma \vdash (\forall X) t \doteq_s t'$$

Reguli de deducție

(S, Σ, Γ) specificație, X, Y mulțimi disjuncte de variabile

Abstractizarea $\dfrac{(\forall X) t_1 \doteq_s t_2}{(\forall X \cup Y) t_1 \doteq_s t_2}$

Concretizarea $\dfrac{(\forall X \cup Y) t_1 \doteq_s t_2}{(\forall X) t_1 \doteq_s t_2}$ $\quad t_1, t_2 \in T_\Sigma(X)_s$, $\quad Y_s \neq \emptyset \Rightarrow T_{\Sigma,s} \neq \emptyset$

Propoziția 30

Abstractizarea și Concretizarea sunt reguli de deducție corecte.

Dem. exercițiu

Exemplu

Orice funcție inversabilă la dreapta este injectivă.

$S := \{s\}$, $\Sigma := \{f : s \to s, g : s \to s\}$
$\Gamma := \{(\forall \{z\})\, g(f(z)) \doteq z\}$ (g inversa la dreapta a lui f)

$\Gamma \vdash (\forall \{x, y\})\, x \doteq y \text{ if } \{f(x) \doteq f(y)\}$

$\Gamma \cup \{(\forall \emptyset)\, f(c_x) \doteq f(c_y)\} \vdash (\forall \emptyset)\, c_x \doteq c_y$ (teorema constantelor)
(1) $(\forall \emptyset)\, f(c_x) \doteq f(c_y)$ (ipoteză)
(2) $(\forall \emptyset)\, g(f(c_x)) \doteq g(f(c_y))$ (C_Σ)
(3) $(\forall \emptyset)\, g(f(c_y)) \doteq c_y$ (Sub$\{z \leftarrow c_y\}$)
(4) $(\forall \emptyset)\, g(f(c_x)) \doteq c_y$ (2,3,T)
(5) $(\forall \{z\})\, z \doteq g(f(z))$ (S)
(6) $(\forall \emptyset)\, c_x \doteq g(f(c_x))$ (5, Sub$\{z \leftarrow c_x\}$)
(7) $(\forall \emptyset)\, c_x \doteq c_y$ (4,6, T)

Programare logică

Logică ecuațională

Regula SR (Subterm Replacement)

Contexte

(S, Σ) signatură, X mulțime de variabile

- Pentru $t \in T_\Sigma(X)$ și $y \in X$ notăm
  $nr_y(t) =$ numărul de apariții ale lui y în t
- Dacă $z \notin X$ atunci un termen $c \in T_\Sigma(X \cup \{z\})$ se numește context dacă $nr_z(c) = 1$.
- Dacă $t_0 \in T_\Sigma(X)$ și t_0 are același sort cu z, atunci notăm
  $$c[z \leftarrow t_0] := \{z \leftarrow t_0\}(c)$$
  pentru un context $c \in T_\Sigma(X \cup \{z\})$, unde
  $\{z \leftarrow t_0\} : X \cup \{z\} \to T_\Sigma(X)$ este substituția
  $\{z \leftarrow t_0\}(z) = t_0$ și $\{z \leftarrow t_0\}(x) = x$ or. $x \in X$

Regula de deducție SR_Γ

(S, Σ, Γ) specificație

$$\boxed{\mathrm{SR}_\Gamma \quad \frac{(\forall X)\theta(u) \doteq_{s'} \theta(v) \text{ or. } u \doteq_s v \in H}{(\forall X)c[z \leftarrow \theta(l)] \doteq_{s''} c[z \leftarrow \theta(r)]}}$$

unde $(\forall Y)l \doteq_s r \text{ if } H \in \Gamma$, $\theta : Y \to T_\Sigma(X)$ substituție
$c \in T_\Sigma(X \cup \{z\})_{s''}$, $z \notin X$, $nr_z(c) = 1$

- $c = z \Rightarrow \mathrm{SR}_\Gamma = \mathrm{Sub}_\Gamma$
- E mulțime de ecuații necondiționate

$$\boxed{\mathrm{SR}_E \quad \frac{}{(\forall X)c[z \leftarrow \theta(l)] \doteq_{s''} c[z \leftarrow \theta(r)]}}$$

unde $(\forall Y)l \doteq_s r \in E$, $\theta : Y \to T_\Sigma(X)$ substituție
$c \in T_\Sigma(X \cup \{z\})_{s''}$, $z \notin X$, $nr_z(c) = 1$

Exemplu

$E = \{x + 0 \doteq x, x + succ(y) \doteq succ(x + y)\}$

$E \vdash_{SR_E} 0 + succ(0) \doteq succ(0)$

(1') $x + succ(y) \doteq succ(x + y) \in E$

(2') $0 + succ(0) \doteq succ(0 + 0)$ $(1', SR_E, c := z, \theta := \{x, y \leftarrow 0\})$

(3') $x + 0 = x \in E$

(4') $succ(0 + 0) \doteq succ(0)$ $(3', SR_E, c := succ(z), \theta := \{x \leftarrow 0\})$

(5') $0 + succ(0) \doteq succ(0)$ $(2', 4', T)$

---

(1) $x + succ(y) \doteq succ(x + y) \in E$

(2) $0 + succ(0) \doteq succ(0 + 0)$ $(1, Sub\{x, y \leftarrow 0\})$

(3) $x + 0 = x \in E$

(4) $0 + 0 \doteq 0$ $(3, Sub\{x \leftarrow 0\})$

(5) $succ(0 + 0) \doteq succ(0)$ $(4, C)$

(6) $0 + succ(0) \doteq succ(0)$ $(2, 5, T)$

Regula de deducție SR_Γ

Propoziția 31

SR_Γ este regulă de deducție corectă.

Dem. Fie $(\forall Y)l \doteq_s r$ *if* $H \in \Gamma$, $\theta : Y \to T_\Sigma(X)$ substituție $c \in T_\Sigma(X \cup \{z\})_{s''}$, $z \notin X$, $nr_z(c) = 1$ a.î. $\Gamma \models (\forall X)\theta(u) \doteq_{s'} \theta(v)$ or. $u \doteq_s v \in H$.
Demonstrăm că $\Gamma \models (\forall X)c[z \leftarrow \theta(l)] \doteq_{s''} c[z \leftarrow \theta(r)]$ prin inducție după $|c|$ (lungimea lui c).

- Dacă $|c| = 1$, atunci $c = z$ și $\Gamma \models (\forall X)\theta(l) \doteq_s \theta(r)$ deoarece Sub_Γ este corectă.

Continuare

- Presupunem că $\Gamma \models (\forall X)c'[z \leftarrow \theta(l)] \doteq_{s''} c'[z \leftarrow \theta(r)]$ dacă $|c'| < |c|$. Atunci ex. $\sigma : s_1 \cdots s_n \to s'' \in \Sigma$, $t_1 \in T_\Sigma(X)_{s_1}, \ldots, t_n \in T_\Sigma(X)_{s_n}$ și k a.î. $c = \sigma(t_1, \ldots, t_k, \ldots, t_n)$ și $nr_z(t_k) = 1$.
  Pentru t_k aplicăm ipoteza de inducție:
  $\Gamma \models (\forall X)t_k[z \leftarrow \theta(l)] \doteq_{s_i} t_k[z \leftarrow \theta(r)]$.
  Deoarece $\Gamma \models (\forall X)t_i \doteq_{s_i} t_i$ or. $i \neq k$, aplicând corectitudinea regulii C_Σ, obținem
  $$\Gamma \models (\forall X)\sigma(t_1, \ldots, t_k[z \leftarrow \theta(l)], \ldots, t_n) \doteq_{s''} \sigma(t_1, \ldots, t_k[z \leftarrow \theta(r)], \ldots, t_n).$$
  Observăm că $c[z \leftarrow t] = \sigma(t_1, \ldots, t_k[z \leftarrow t], \ldots, t_n)$ or. $t \in T_\Sigma(X)_s$, deci $\Gamma \models (\forall X)c[z \leftarrow \theta(l)] \doteq_{s''} c[z \leftarrow \theta(r)]$.
  $\square$

Regula de deducție SR_Γ

Definim $\sim_{SR} \subseteq T_\Sigma(X) \times T_\Sigma(X)$ prin
$t \sim_{SR} t' \Leftrightarrow \Gamma \vdash_{R,S,T,SR_\Gamma} (\forall X)t \doteq_s t'$

Propoziția 32

$\sim_{SR}$ este congruență închisă la substituție.

Dem. Din definiție, $\sim_{SR}$ este închisă la R, S, T, deci este o relație de echivalență.

- $\sim_{SR}$ este închisă la Sub_Γ
  Sub_Γ se obține aplicând SR_Γ cu $c = z$.

Continuare

- $\sim_{SR}$ este închisă la C_Σ (schiță)
  Fie $\sigma : s_1 \cdots s_n \to s'' \in \Sigma$ și $t_1 \sim_{SR} t'_1, \ldots, t_n \sim_{SR} t'_n$.

(1) Dacă $k \in \{1, \ldots, n\}$ și $(\forall X)t_k \doteq_{s_k} t'_k$ se deduce prin aplicarea regulii SR_Γ, atunci există $(\forall Y)l \doteq_s r$ *if* $H \in \Gamma$, $\theta : Y \to T_\Sigma(X)$ substituție $c \in T_\Sigma(X \cup \{z\})_{s_k}$, $z \notin X$, $nr_z(c) = 1$ a.î. $t_k = c[z \leftarrow \theta(l)]$ și $t'_k = c[z \leftarrow \theta(r)]$.
Definim $c' = \sigma(t_1, \ldots, t_{k-1}, c, t_{k+1}, \ldots, t_n)$, atunci
$\sigma(t_1, \ldots, t_k, \ldots, t_n) = c'[z \leftarrow \theta(l)]$ și
$\sigma(t_1, \ldots, t'_k, \ldots, t_n) = c'[z \leftarrow \theta(r)]$,
deci $(\forall X)\sigma(t_1, \ldots, t_k, \ldots, t_n) \doteq_s \sigma(t_1, \ldots, t'_k, \ldots, t_n)$ se deduce prin aplicarea regulii SR_Γ.

(2) Dacă $k \in \{1, \ldots, n\}$ și $t_k \sim_{SR} t'_k$ atunci
$\sigma(t_1, \ldots, t_{k-1}, t_k, t_{k+1}, \ldots, t_n) \sim_{SR} \sigma(t_1, \ldots, t_{k-1}, t'_k, t_{k+1}, \ldots, t_n)$.

(3) $\sigma(t_1, t_2, \ldots, t_n) \sim_{SR} \sigma(t'_1, t_2, \ldots, t_n) \sim_{SR} \sigma(t'_1, t'_2, t_3 \ldots, t_n) \sim_{SR} \sigma(t'_1, t'_2, t'_3, \ldots, t_n) \sim_{SR} \cdots \sim_{SR} \sigma(t'_1, t'_2, \ldots, t'_n)$ $\square$

Completitudinea SR_Γ

(S, Σ, Γ) specificație, X mulțime de variabile, t, $t' \in T_\Sigma(X)_s$

Teorema 33

Sunt echivalente:

- (a) $\Gamma \vdash_{R,S,T,C_\Sigma,Sub_\Gamma} (\forall X)t \doteq_s t'$
- (b) $\Gamma \vdash_{R,S,T,SR_\Gamma} (\forall X)t \doteq_s t'$

Dem. (a) $\Rightarrow$ (b) ($\sim_\Gamma \subseteq \sim_{SR}$) Rezultă din Propoziția 32 și din faptul că $\sim_\Gamma = \equiv_\Gamma$ este cea mi mică congruență închisă la substituție, definită pe $T_\Sigma(X)$.

(b) $\Rightarrow$ (a) ($\sim_{SR} \subseteq \sim_\Gamma$) Din corectitudinea regulii SR$_\Gamma$ (Propoziția 31) și Teorema de completitudine (Teorema 29), rezultă $\sim_{SR} \subseteq \equiv_\Gamma = \sim_\Gamma$. □

Rescrierea termenilor

Rescrierea

(S, Σ) signatură

- O regulă de rescriere este formată dintr-o mulțime de variabile Y și doi termeni l, r de același sort din $T_\Sigma(Y)$ a.î.
  - l nu este variabilă
  - $Var(r) \subseteq Var(l) = Y$.

  Vom nota $l \to r$ ($l \to_s r$).
- Un sistem de rescriere (TRS) este o mulțime finită de reguli de rescriere.
- $R = \{x + 0 \to x, x + succ(y) \to succ(x + y)\}$

Relația $\to_R$

- Fie R un sistem de rescriere. Dacă t, $t' \in T_\Sigma(X)_s$ definim relația $t \to_R t'$ astfel:

$$\boxed{\begin{array}{lll} t \to_R t' & \Leftrightarrow & t \text{ este } c[z \leftarrow \theta(l)] \text{ și} \\ & & t' \text{ este } c[z \leftarrow \theta(r)], \text{ unde} \\ & & c \in T_\Sigma(X \cup \{z\}),\ z \notin X,\ nr_z(c) = 1 \\ & & l \to r \in R \text{ cu } Var(l) = Y, \\ & & \theta : Y \to T_\Sigma(X) \text{ este o substituție.} \end{array}}$$

- $t \to_R t'$ dacă și numai dacă t' se obține din t înlocuind o instanță a lui l cu o instanță a lui r.

Sistemul de rescriere R_E

(S, Σ) specificație, E mulțime de ecuații necondiționate a.î. $Var(r) \subseteq Var(l) = Y$ or. $(\forall Y) l \doteq_s r \in E$.

$R_E := \{l \to r \mid (\forall Y) l \doteq_s r \in E\}$ este sistemul de rescriere determinat de E.

În acest caz vom nota $\to_E := \to_{R_E}$

ecuațiile devin reguli de rescriere prin orientare

În **Maude** ecuațiile `eq t = t'` trebuie sa verifice condiția $Var($`t'`$) \subseteq Var($`t`$)$ deoarece în reduceri ecuațiile sunt folosite ca reguli de rescriere.

Exemplu

$E = \{x + 0 \doteq x, x + succ(y) \doteq succ(x + y)\}$

$R_E = \{x + 0 \to x, x + succ(y) \to succ(x + y)\}$

$0 + succ(0) \to_E succ(0 + 0)$
$(l := x + succ(y), r := succ(x + y), c := z, \theta := \{x, y \leftarrow 0\})$

$succ(0 + 0) \to_E succ(0)$
$(l := x + 0, r := x, c := succ(z), \theta := \{x \leftarrow 0\})$

$0 + succ(0) \to_E succ(0 + 0) \to_E succ(0)$

Maude

```
fmod INTLIST is
protecting INT .
sort IntList .
subsort Int < IntList .
op nil : -> IntList .
op _,_ : IntList IntList -> IntList [ assoc id: nil] .
vars L1 L2 L3 : IntList . var x : Int .
eq L1 , x , L2 , - x , L3 = L1 , L2 , L3 .
endfm

set trace on .
reduce  4 , 3 , 5 , 6 , - 3 , 8 , -5 .
```


Maude

```
reduce in INTLIST : 4,3,5,6,-3,8,-5 .
*********** equation
eq L1,x,L2,- x,L3 = L1,L2,L3 .
L1 --> nil
x --> 5
L2 --> 6,-3,8
L3 --> nil
5,6,-3,8,-5
--->
nil,(6,-3,8),nil
```


Maude

```
*********** equation
eq L1,x,L2,- x,L3 = L1,L2,L3 .
L1 --> nil
x --> 3
L2 --> 6
L3 --> nil
3,6,-3,8
--->
(nil,6,nil),8
rewrites: 2 in 6264381186ms cpu (27ms real)
result IntList: 4,6,8
```

Sisteme de rescriere abstracte

Un sistem de rescriere abstract este o pereche $(T, \to)$, unde T este o mulțime și $\to \subseteq T \times T$ ($\to$ este o relație binară pe T).

Definiții

$\leftarrow := \to^{-1}$ (relația inversă)

$\leftrightarrow := \to \cup \leftarrow$ (închiderea simetrică)

$\stackrel{*}{\to} := (\to)^*$ (închiderea reflexivă și tranzitivă)

$\stackrel{*}{\leftrightarrow} := (\leftrightarrow)^*$ (echivalența generată)

$(T_\Sigma(X)_s, \to_s)$ sistem de rescriere abstract

Exemplu

$T := \mathbb{N} \setminus \{0, 1\}$, $\to := \{(m, k) \mid k < m, k|m\}$

- $\leftarrow = \{(k, m) \mid k < m, k|m\}$
- $\leftrightarrow = \{(k_1, k_2) \mid k_1 \neq k_2, k_1|k_2$ sau $k_2|k_1\}$
- $\stackrel{+}{\to} = \{(m, k) \mid$ ex.$n \geq 0$, ex. $k_1, \cdots, k_n \in T$
  $m \to k_1 \to \cdots \to k_n \to k\}$
- $\stackrel{*}{\to} = \stackrel{+}{\to} \cup \{(k, k) | k \in T\}$
- Cine este $\stackrel{*}{\leftrightarrow}$?

Rescriere și deducție

(S, Σ) signatură, X mulțime de variabile,
E mulțime de ecuații necondiționate,
R_E sistemul de rescriere determinat de E,
$\to_E \subseteq T_\Sigma(X) \times T_\Sigma(X)$ relația de rescriere

Teorema 34.

Fie t, $t' \in T_\Sigma(X)_s$. Sunt echivalente:

(1) $E \models (\forall X) t \doteq_s t'$

(2) $E \vdash_{R,S,T,C_\Sigma,Sub_E} (\forall X) t \doteq_s t'$

(3) $E \vdash_{R,S,T,SR_E} (\forall X) t \doteq_s t'$

(4) $t \stackrel{*}{\leftrightarrow}_E t'$,

unde $\stackrel{*}{\leftrightarrow}_E$ este echivalența generată de $\to_E$.

Continuare

Dem. (1) $\Leftrightarrow$ (2) $\Leftrightarrow$ (3) rezultă din $\equiv_E = \sim_E = \sim_{SR}$

(3) $\Leftrightarrow$ (4) revine la $\sim_{SR} = \stackrel{*}{\leftrightarrow}_E$

Din $\to_E \subseteq \sim_{SR}$ rezultă $\stackrel{*}{\leftrightarrow}_E \subseteq \sim_{SR}$.

Pentru a demonstra implicația inversă vom arăta că $\stackrel{*}{\leftrightarrow}_E$ este închisă la C_Σ și Sub_E.

- $\stackrel{*}{\leftrightarrow}_E$ este închisă la Sub_E
  $(\forall Y) t \doteq_s t' \in E$, $\theta : Y \to T_\Sigma(X)$ substituție implică $t \to_E t'$ sau $t' \to_E t$.
  Dacă $t \to_E t'$ atunci $\tilde{\theta}(t) \to_E \tilde{\theta}(t')$ pentru $c = z$.
  Dacă $t' \to_E t$ atunci $\tilde{\theta}(t') \to_E \tilde{\theta}(t)$ pentru $c = z$.
  Rezultă $\tilde{\theta}(t) \leftrightarrow_E \tilde{\theta}(t')$.
- $\stackrel{*}{\leftrightarrow}_E$ este închisă la C_Σ
  Fie $\sigma : s_1 \cdots s_n \to s'' \in \Sigma$ și $k \in \{1, \ldots, n\}$.

Continuare

(1) $t_k \to_E t'_k$ implică $\sigma(t_1, \ldots, t_k, \ldots, t_n) \to_E \sigma(t_1, \ldots, t'_k, \ldots, t_n)$

Din ipoteză, t_k este $c[z \leftarrow \theta(l)]$, t'_k este $c[z \leftarrow \theta(r)]$, unde $c \in T_\Sigma(X \cup \{z\})$ e un context, $l \to r \in R_E$ cu $Var(l) = Y$ și $\theta : Y \to T_\Sigma(X)$.
Definim $c' := \sigma(t_1, \ldots, c, \ldots, t_n)$. Rezultă că $\sigma(t_1, \ldots, t_k, \ldots, t_n)$ este $c'[z \leftarrow \theta(l)]$ și $\sigma(t_1, \ldots, t'_k, \ldots, t_n)$ este $c'[z \leftarrow \theta(r)]$.

(2) Demonstrăm prin inducție după $p \geq 1$ că
$t_k \stackrel{p}{\leftrightarrow}_E t'_k$ implică $\sigma(t_1, \ldots, t_k, \ldots, t_n) \stackrel{*}{\leftrightarrow}_E \sigma(t_1, \ldots, t'_k, \ldots, t_n)$
Pentru $p = 1$ aplicăm (1) și simetria.
Dacă $t_k \stackrel{p+1}{\leftrightarrow}_E t'_k$ atunci $t_k \stackrel{p}{\leftrightarrow}_E t''_k \leftrightarrow_E t'_k$. Din ip. de inducție
$\sigma(t_1, \ldots, t_k, \ldots, t_n) \stackrel{*}{\leftrightarrow}_E \sigma(t_1, \ldots, t''_k, \ldots, t_n) \stackrel{*}{\leftrightarrow}_E \sigma(t_1, \ldots, t'_k, \ldots, t_n)$
deci $\sigma(t_1, \ldots, t_k, \ldots, t_n) \stackrel{*}{\leftrightarrow}_E \sigma(t_1, \ldots, t'_k, \ldots, t_n)$.

Continuare

(3) Din (2), dacă $t_1 \stackrel{*}{\leftrightarrow}_E t'_1$, …, $t_n \stackrel{*}{\leftrightarrow}_E t'_n$ atunci
$\sigma(t_1, \ldots, t_k, \ldots, t_n) \stackrel{*}{\leftrightarrow}_E \sigma(t_1, \ldots, t'_k, \ldots, t_n)$
or. $k \in \{1, \ldots, n\}$

(4) $\sigma(t_1, \ldots, t_n) \stackrel{*}{\leftrightarrow}_E \sigma(t'_1, t_2 \ldots, t_n) \stackrel{*}{\leftrightarrow}_E \sigma(t'_1, t'_2, \ldots, t_n) \stackrel{*}{\leftrightarrow}_E \cdots$
$\cdots \stackrel{*}{\leftrightarrow}_E \sigma(t'_1, \ldots, t'_k, t_{k+1}, \ldots, t_n) \stackrel{*}{\leftrightarrow}_E \cdots \stackrel{*}{\leftrightarrow}_E \sigma(t'_1, \ldots, t'_n)$

În consecință, $\stackrel{*}{\leftrightarrow}_E$ este închisă la C_Σ și Sub_E, deci $\sim_{SR} = \equiv_E \subseteq \stackrel{*}{\leftrightarrow}_E$ din Propoziția 27. $\square$

Substituţii. Unificare

F. Baader, T. Nipkow, **Terms Rewriting and All That**, Cambridge University Press, 1998.

Substituţie

(S, Σ) signatură multisortată, X şi Y mulţimi de variabile
O substituţie a variabilelor din X cu termeni din $T_\Sigma(Y)$ este o funcţie $\nu : X \to T_\Sigma(Y)$.
Substituţia ν se extinde la o funcţie $\tilde{\nu} : T_\Sigma(X) \to T_\Sigma(Y)$ după cum urmează:

- $\tilde{\nu}_s(x) := \nu(x)$ or. $x \in X_s$,
- $\tilde{\nu}_s(\sigma) := \sigma$ or. $\sigma :\to s$,
- $\tilde{\nu}_s(\sigma(t_1, \ldots, t_n)) := \sigma(\tilde{\nu}_{s_1}(t_1), \ldots, \tilde{\nu}_{s_n}(t_n))$ or. $\sigma : s_1 \ldots s_n \to s$, or. $t_1 \in T_\Sigma(X)_{s_1}$, ..., $t_n \in T_\Sigma(X)_{s_n}$.

Observaţie: $\tilde{\nu} : T_\Sigma(X) \to T_\Sigma(Y)$ morfism de (S, Σ)-algebre.

Substituţie

X, Y şi Z mulţimi de variabile

- Dacă $\nu : X \to T_\Sigma(Y)$, $\mu : X \to T_\Sigma(Y)$ atunci
  $\nu = \mu \Leftrightarrow \tilde{\nu} = \tilde{\mu}$.
  - vom identifica uneori $\tilde{\nu}$ cu ν
- $\{x_1 \leftarrow t_1, \cdots, x_n \leftarrow t_n\}$ e notaţie pt. $\sigma : X \to T_\Sigma(X)$, $\sigma(x_i) := t_i$ or. $i = 1, \ldots, n$ şi $\sigma(x) := x$ pt. $x \neq x_i$
- compunerea substituţiilor $\nu : X \to T_\Sigma(Y)$, $\mu : Y \to T_\Sigma(Z)$ $\nu; \mu : X \to T_\Sigma(Z)$, $(\nu; \mu)_s(x) := \nu; \tilde{\mu}$
- compunerea substituţiilor este asociativă,
- compunerea substituţiilor nu este în general comutativă.

Exemplu

$S = \{s\}$, $\Sigma = \{a :\to s, p : sssss \to s, f : s \to s, g : s \to s\}$,
$X = \{x, y, z, u, v\}$

$t \ = \ p(u, v, x, y, z)$

$\nu = \{x \leftarrow f(y), y \leftarrow f(a), z \leftarrow u\}$

$\nu(t) \ = \ p(u, v, f(y), f(a), u)$

$\mu = \{y \leftarrow g(a), u \leftarrow z, v \leftarrow f(f(a))\}$

$\nu; \mu = \{x \leftarrow f(g(a)), y \leftarrow f(a), u \leftarrow z, v \leftarrow f(f(a))\}$

$(\nu; \mu)(t) \ = \ p(z, f(f(a)), f(g(a)), f(a), z)$

$(\mu; \nu)(t) \ = \ p(u, f(f(a)), f(y), g(a), u)$

Termeni. Cazul monosortat

(S, Σ) signatură monosortată, $S = \{s\}$
X mulțime de variabile,
$T_\Sigma(X)$ termenii cu variabile din X

O ecuație este o pereche de termeni $\langle t, t' \rangle$, unde t, $t' \in T_\Sigma(X)$.
Ecuația $\langle t, t' \rangle$ o vom nota $t \doteq t'$.

Egalitatea termenilor

Dacă $t = \sigma(t_1, \ldots, t_n)$ și $t' = \tau(t'_1, \ldots, t'_k)$ atunci

$$t = t' \Leftrightarrow \sigma = \tau,\ n = k,\ t_i = t'_i \text{ or. } i$$

$\doteq$ egalitate formală, $=$ egalitate efectivă

Unificare. Cazul monosortat.

$(S = \{s\}, \Sigma)$ signatură, X mulțime de variabile

- O problemă de unificare este o mulțime finită de ecuații
$$U = \{t_1 \doteq t'_1, \ldots, t_n \doteq t'_n\}$$
- Un unificator (o soluție) pentru U este o substituție $\nu : X \to T_\Sigma(X)$ a.î. $\nu(t_i) = \nu(t'_i)$ or. $i = 1, \ldots, n$.
Notăm cu $Unif(U)$ mulțimea unificatorilor lui U.
- Un unificator $\nu \in Unif(U)$ este un cel mai general unificator (cgu, mgu) dacă or. $\nu' \in Unif(U)$ ex. τ substituție a.î. $\nu' = \nu; \tau$.

Exemplu

$S = \{s\}$, $\Sigma = \{0 :\to s, + : ss \to s, * : ss \to s\}$, $X = \{x, y, z\}$
$t = x + (y * y) = +(x, *(y, y))$, $u = x + (y * x) = +(x, *(y, x))$

- $\nu(x) := y$, $\nu(y) := y$, $\nu(z) := z$,
- $\nu'(x) := 0$, $\nu'(y) := 0$, $\nu'(z) := z$, $\nu' = \nu; \{y \leftarrow 0\}$
- $\nu"(x) := z + 0$, $\nu"(y) := z + 0$, $\nu"(z) := z$,
$\nu" = \nu; \{y \leftarrow z + 0\}$
- $\mu(x) := z$, $\mu(y) := z$, $\mu(z) := z$, $\mu = \nu; \{y \leftarrow z\}$
- ν și μ sunt cgu
cgu nu este unic

Notații

- $Unif(U) :=$ mulțimea unificatorilor lui U
- $Var(U) := \bigcup_{i=1}^{n}(Var(t_i) \bigcup Var(t'_i))$, unde
$Var(t) :=$ mulțimea variabilelor care apar în $t \in T_\Sigma(X)$
- dacă $\nu = \{x_1 \leftarrow t_1, \cdots, x_n \leftarrow t_n\}$ atunci
$\{x_1 \leftarrow t_1, \cdots, x_n \leftarrow t_n\}U := \{\nu(t) \doteq \nu(t') \mid t \doteq t' \in U\}$

Unificare

$(S = \{*\}, \Sigma)$ signatură, X mulțime de variabile

Spunem că problema de unificare $R = \{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ este rezolvată dacă

- $x_i \in X$, $x_1 \neq x_j$ or. $i \neq j$
- $x_i \notin \bigcup_{i=1}^{n} Var(t_i)$ or. $i = 1, \ldots, n$.

O problemă rezolvată R definește o substituție ν_R

- $\nu_R(x_i) := \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$
- $\nu_R \in Unif(R)$

Algoritmul transformă o problemă de unificare U într-o problemă de unificare R. Dacă $R = \emptyset$ atunci U nu are unificatori. În caz contrar, R este rezovată, iar substituția determinată de R este cgu pentru U.

Unificare

Propoziția 35.

Dacă $R = \{x_1 \doteq t_1, \ldots, x_n \doteq t_n\}$ este o problemă de unificare rezolvată, atunci ν_R este cgu idempotent pentru R:

- $\nu' = \nu_R; \nu'$ or. $\nu' \in Unif(R)$,
- $\nu_R; \nu_R = \nu_R$.

Dem.

$$(\nu_R; \nu')(x) = \nu'(\nu_R(x)) = \begin{cases} \nu'(x) & x \notin \{x_1, \ldots, x_n\}, \\ \nu'(t_i) & x = x_i \end{cases}$$

Dar $\nu'(t_i) = \nu'(x_i)$ deoarece $\nu' \in Unif(R)$,
deci $(\nu_R; \nu')(x) = \nu'(x)$ or. $x \in X$.
Pentru $\nu' = \nu_R$, rezultă $\nu_R; \nu_R = \nu_R$. □

Algoritmul de unificare

$(S = \{*\}, \Sigma)$ signatură, X mulțime de variabile
Intrare: $U = \{t_1 \doteq t'_1, \ldots, t_n \doteq t'_n\}$
Inițializare: $R = U$
Se execută nedeterminist următorii pași, atâta timp cât este posibil:
(1) Șterge: $R \cup \{t \doteq t\} \Rightarrow R$
(2) Orientează: $R \cup \{t \doteq x\} \Rightarrow R \cup \{x \doteq t\}$ dc. $x \in X$, $t \notin X$
(3) Descompune:
▷ $R \cup \{\sigma(e_1, \ldots, e_n) \doteq \sigma(e'_1, \ldots, e'_n)\} \Rightarrow R \cup \{e_1 \doteq e'_1, \ldots, e_n \doteq e'_n\}$
▷ $R \cup \{\sigma(e_1, \ldots, e_n) \doteq \tau(e'_1, \ldots, e'_k)\} \Rightarrow \emptyset$ dc. $\sigma \neq \tau$
(4) Elimină:
▷ $R \cup \{x \doteq t\} \Rightarrow \{x \doteq t\} \cup \{x \leftarrow t\}R$ dc.. $x \in Var(R) \setminus Var(t)$
▷ $R \cup \{x \doteq t\} \Rightarrow \emptyset$ dc. $x \in Var(t)$ și $t \neq x$
Ieșire: daca $R = \emptyset$ atunci nu exită soluții pentru U
daca $R \neq \emptyset$ atunci R este cgu pentru U

$\Sigma\{g : s \to s, h : s \to s, f : sss \to s\}$, $X = \{x, y, z, w\}$

$U = R = \{g(y) \doteq x, f(x, h(x), y) \doteq f(g(z), w, z)\}$

(2) $R = \{x \doteq g(y), f(x, h(x), y) \doteq f(g(z), w, z)\}$

(4) $R = \{x \doteq g(y), f(g(y), h(g(y)), y) \doteq f(g(z), w, z)\}$

(3) $R = \{x \doteq g(y), g(y) \doteq g(z), h(g(y)) \doteq w, y \doteq z\}$

(3) $R = \{x \doteq g(y), y \doteq z, h(g(y)) \doteq w, y \doteq z\}$

(4) $R = \{x \doteq g(z), y \doteq z, h(g(z)) \doteq w, z \doteq z\}$

(2)(1) $R = \{x \doteq g(z), y \doteq z, w \doteq h(g(z))\}$

(4) $R = \{x \doteq g(z), y \doteq z, w \doteq h(g(z))\}$ cgu

$\Sigma\{a :\to s, g : s \to s, h : s \to s, f : sss \to s\}$, $X = \{x, z, w\}$

$U = R = \{g(a) \doteq x, f(x, h(x), a) \doteq f(g(z), w, z)\}$

(2) $R = \{x \doteq g(a), f(x, h(x), a) \doteq f(g(z), w, z)\}$

(4) $R = \{x \doteq g(a), f(g(a), h(g(a)), a) \doteq f(g(z), w, z)\}$

(3) $R = \{x \doteq g(a), g(a) \doteq g(z), h(g(a)) \doteq w, a \doteq z\}$

(3) $R = \{x \doteq g(a), a \doteq z, h(g(a)) \doteq w, a \doteq z\}$

(2) $R = \{x \doteq g(a), z \doteq a, w \doteq h(g(a)), z \doteq a\}$

(4) $R = \{x \doteq g(a), z \doteq a, w \doteq h(g(a)), a \doteq a\}$

(1) $R = \{x \doteq g(a), z \doteq a, w \doteq h(g(a))\}$ cgu

$\Sigma\{b :\to s, g : s \to s, h : s \to s, f : sss \to s\}$, $X = \{x, y, z\}$

$U = R = \{g(y) \doteq x, f(x, h(x), y) \doteq f(g(z), b, z)\}$

(2) $R = \{x \doteq g(y), f(x, h(x), y) \doteq f(g(z), b, z)\}$

(4) $R = \{x \doteq g(y), f(g(y), h(g(y)), y) \doteq f(g(z), b, z)\}$

(3) $R = \{x \doteq g(y), g(y) \doteq g(z), h(g(y)) \doteq b, y \doteq z\}$

(3) $R = \{x \doteq g(y), y \doteq z, h(g(y)) \doteq b, y \doteq z\}$

(4) $R = \{x \doteq g(z), y \doteq z, h(g(z)) \doteq b, z \doteq z\}$

(3) $R = \emptyset$ deoarece $\boxed{\mathbf{b} \neq \mathbf{h}}$

$\Sigma\{g : s \to s, h : s \to s, f : sss \to s\}$, $X = \{x, y, z, w\}$

$U = R = \{g(y) \doteq x, f(x, h(x), y) \doteq f(y, w, z)\}$

(2) $R = \{x \doteq g(y), f(x, h(x), y) \doteq f(y, w, z)\}$

(4) $R = \{x \doteq g(y), f(g(y), h(g(y)), y) \doteq f(y, w, z)\}$

(3) $R = \{x \doteq g(y), g(y) \doteq y, h(g(y)) \doteq w, y \doteq z\}$

(2) $R = \{x \doteq g(z), y \doteq g(y), h(g(y)) \doteq w, y \doteq z\}$

(4) $R = \emptyset$ deoarece $\boxed{y \in Var(g(y))}$

Terminare

Propoziția 36.

Algoritmul de unificare se termină.

Dem. Fie R problema de unificare. Notăm
$n_{sol} := |\{x \in Var(R) | R = R' \cup \{x \doteq t\}, x \notin Var(R') \cup Var(t)\}|$,
$n_1 := |Var(R)| - n_{sol}$, $n_2 := \sum_{s \doteq t \in R}(|s| + |t|)$,
$n_3 := |\{t \doteq x \in R | t \notin X, x \in X\}|$
Fiecare pas al algoritmului modifică n_1, n_2, n_3

	n_1	n_2	n_3
Șterge	$\geq$	$>$	
Orientează	$\geq$	$=$	$>$
Descompune	$\geq$	$>$	
Elimină	$>$		

Dacă la execuția unui pas (n_1, n_2, n_3) se schimbă în (n'_1, n'_2, n'_3), atunci $(n_1, n_2, n_3) >_{lex} (n'_1, n'_2, n'_3)$, adică (n_1, n_2, n_3) descrește strict în ordine lexicografică. □

Corectitudine

Propoziția 37.

Dacă $R \Rightarrow T$ și $T \neq \emptyset$, atunci $Unif(R) = Unif(T)$.

Dem. Evident adevărat pentru Ștergeși Orientează.

Descompune. $R = R' \cup \{\sigma(e_1, \ldots, e_n) \doteq \sigma(e'_1, \ldots, e'_n)\}$

$T = R' \cup \{e_1 \doteq e'_1, \ldots, e_n \doteq e'_n\}$

Dacă $\nu : X \to T_\Sigma(X)$ substituție, atunci

$\nu(\sigma(e_1, \ldots, e_n)) = \nu(\sigma(e'_1, \ldots, e'_n)) \Leftrightarrow \nu(e_i) = \nu(e'_i)$ or. i

Elimină:

$R = R' \cup \{x \doteq t\}$, $T = \{x \doteq t\} \cup \{x \leftarrow t\}R'$, $x \in Var(R') \setminus Var(t)$

$\nu \in Unif(R) \Leftrightarrow \nu \in Unif(R')$ și $\nu \in Unif(\{x \doteq t\})$

Dacă $\nu \in Unif(\{x \doteq t\})$ atunci $\{x \leftarrow t\}; \nu = \nu$

$$\begin{aligned}\nu \in Unif(R) &\Leftrightarrow \{x \leftarrow t\}; \nu \in Unif(R') \text{ și } \nu \in Unif(\{x \doteq t\})\\ &\Leftrightarrow \nu \in Unif(\{x \leftarrow t\}R') \text{ și } \nu \in Unif(\{x \doteq t\})\\ &\Leftrightarrow \nu \in Unif(T) \quad \square\end{aligned}$$

Completitudine

Propoziția 38.

Dacă $U \Rightarrow \cdots \Rightarrow R \Rightarrow \emptyset$ atunci U nu are soluții.

Dem. Distingem două cazuri.

- La pasul Descompune, există $\sigma(e_1, \ldots, e_n) \doteq \tau(e'_1, \ldots, e'_k) \in R$ cu $\sigma \neq \tau$. Termenii $\sigma(e_1, \ldots, e_n)$ și $\tau(e'_1, \ldots, e'_k)$ nu pot fi unificați deoarece încep cu caractere diferite.
- La pasul Elimină , există $x \doteq t \in R$ cu $x \in Var(t)$ și $t \neq x$. Dacă ν este o substituție, atunci $|\nu(t)| > |\nu(x)|$, deci x și t nu au unificator.

R conține o ecuație care nu poate fi unificată, deci R nu are unificator. Deoarece U și R au aceeasi unificatori, rezultă ca U nu are unificatori. $\square$

Complexitate

- Problema de unificare $U = \{x_1 \doteq f(x_0, x_0), x_2 \doteq f(x_1, x_1), \ldots, x_n \doteq f(x_{n-1}, x_{n-1})\}$ are cgu $R = \{x_1 \leftarrow f(x_0, x_0), x_2 \leftarrow f(f(x_0, x_0), f(x_0, x_0)), \ldots\}$.
- La pasul Elimină, pentru a verifica că o variabilă x_i nu apare în membrul drept al ecuației (occur check) facem 2^i comparații.
- Algoritmul de unificare prezentat anterior este exponențial. Complexitatea poate fi îmbunătățită printr-o reprezentare eficientă a termenilor.

K. Knight, Unification: A Multidisciplinary Survey, ACM Computing Surveys, Vol. 21, No. 1, 1989.

Matching problem

```
fmod MATCH is
sort Elem .
op a : -> Elem .
ops _+_ _*_   : Elem Elem -> Elem .
vars x y : Elem
eq x + (y * y) = x * y .
endfm
reduce in MATCH : (a + y) + (x * x) .
eq x + (y * y) = x * y .
x --> a + y
y --> x
(a + y) + (x * x)
--->
(a + y) * x
rewrites: 1 in 9146759363ms cpu (4ms real)
result Elem: (a + y) * x
```

Matching problem

- Fie p și t termeni cu variabile din X. Spunem că

  p matches t (t este o instanță a lui p)

  dacă există o substituție $\nu : X \to T_\Sigma(X)$ a. î. $\nu(p) = t$.
  Exemplu.
  $p = x + (y * y)$, $t = (a + y) + (x * x)$
  $\nu(x) := a + y$, $\nu(y) := x$
- Fie t' termenul obținut din t prin înlocuirea fiecărei variabile $x \in Var(T)$ cu o operație constantă c_x. Substituția ν poate fi determinată aplicînd algoritmul de unificare ecuației $p \doteq t'$.
  Exemplu.
  $x + (y * y) \doteq (a + c_y) + (c_x * c_x)$
  $\{x \doteq a + c_y,\ y \doteq c_x\}$
- O problemă de *matching* poate fi rezolvată prin unificare.

Sisteme de rescriere complete

$(T, \to)$ sistem de rescriere (T mulțime, $\to \subseteq T \times T$)

Definiții
$t \in T$ este reductibil dc. ex. $t' \in T$ a.î. $t \to t'$
$t_0 \to t_1 \to t_2 \to \cdots$ reducere
$t \in T$ este formă normală(ireductibil) dc. nu este reductibil
t_0 este o formă normală a lui t dc. $t \xrightarrow{*} t_0$ și
t_0 este formă normală
$t_1 \downarrow t_2$ dc. ex. $t \in T$ a.î. $t_1 \xrightarrow{*} t \xleftarrow{*} t_2$
(t_1 și t_2 se întàlnesc, $\downarrow$ relația de întâlnire)

- $T := \mathbb{N} \setminus \{0, 1\}$, $\to := \{(m, k) \mid k < m, k|m\}$
  k este formă normală dacă este număr prim
  $k_1 \downarrow k_2$ dacă nu sunt prime între ele
  k este formă normală a lui m dacă k este factor prim al lui m
- $T := \{a, b\}^*$, $\to := \{(ubav, uabv) | u, v \in T\}$
  $w \in T$ este formă normală dacă $w = a^n b^k$ cu n, $k \geq 0$
  $w_1 \downarrow w_2$ dacă $nr_a(w_1) = nr_a(w_2)$ și $nr_b(w_1) = nr_b(w_2)$

$(T, \to)$ sistem de rescriere (T mulțime, $\to \subseteq T \times T$)

Definiții $(T, \to)$ se numește
noetherian: nu există reduceri infinite $t_0 \to t_1 \to \cdots$
(orice rescriere se termină)
confluent: $t_1 \xleftarrow{*} t \xrightarrow{*} t_2 \Rightarrow t_1 \downarrow t_2$
local confluent: $t_1 \leftarrow t \rightarrow t_2 \Rightarrow t_1 \downarrow t_2$
Church-Rosser: $t_1 \xleftrightarrow{*} t_2 \Rightarrow t_1 \downarrow t_2$
normalizat: orice element are o formă normală
complet (convergent, canonic): confluent și noetherian

- $T := \mathbb{N} \setminus \{0, 1\}$, $\to := \{(k, m) \mid k < m, k|m\}$
  $(T, \to)$ e noetherian, nu e confluent

Proprietăți

Propoziția 39 *

$(T, \to)$ confluent $\Leftrightarrow$ $(T, \to)$ Church-Rosser

Propoziția 40

Dacă $(T, \to)$ este complet atunci orice termen are o singură formă normală. În acest caz vom nota $fn(t)$ forma normală a lui $t \in T$.

Dem. Existența formei normale rezultă din terminare. Fie t un termen si presupunem că t_1 și t_2 sunt forme normale distincte ale lui t.Deoarece $t_1 \stackrel{*}{\leftarrow} t \stackrel{*}{\to} t_2$, din confluență rezultă că exisă u a.î. $t_1 \to u \leftarrow t_2$.În consecință t_1 și t_2 sunt reductibile, ceea ce contrazice faptul că sunt forme normale. $\square$

Proprietăți

Propoziția 41 (Lema lui Newman)

Dacă $(T, \to)$ este local confluent și noetherian atunci este confluent.

Dem. Terminarea asigură faptul că orice termen are o formă normală. Pentru a demonstra confluența este suficient să arătăm că orice element are o formă normală unică.
$U := \{a \in T | a \stackrel{*}{\to} n_1,\ a \stackrel{*}{\to} n_2,\ n_1 \neq n_2 \text{forme normale}\}$
Demonstrăm că are loc următoarea proprietate:
(*) or. $a \in U$ ex. $b \in U$ a.î. $a \to b$
Dacă $U \neq \emptyset$, atunci există $(a_n)_n \subseteq U$ cu
$a_1 \to a_2 \to \cdots \to a_n \to$
ceea ce contrazice proprietatea de terminare. În consecință, $U = \emptyset$ și orice termen t are o unică formă normală $fn(t)$.
Dacă $t_1 \stackrel{*}{\leftarrow} t \stackrel{*}{\to} t_2$, atunci $t_1 \stackrel{*}{\to} fn(t) \stackrel{*}{\leftarrow} t_2$, deci T este confluent.

Continuare

(*) or. $a \in U$ ex. $b \in U$ a.î. $a \to b$

- $n_1 \leftarrow a \to n_2$ implică $n_1 = n_2$ (*imposibil*)
- $n_1 \leftarrow a \stackrel{*}{\to} n_2$ implică $n_1 \leftarrow a \to b \stackrel{*}{\to} n_2$ pentru un $b \in T$; din local confluență, $n_1 \stackrel{*}{\to} t \stackrel{*}{\leftarrow} b$, dar n_1 este formă normală, deci $t = n_1$; avem $b \to n_1$, $b \stackrel{*}{\to} n_2$, deci $b \in U$ și $a \to b$
- $n_1 \stackrel{*}{\leftarrow} a \to n_2$ implică $n_1 \stackrel{*}{\leftarrow} b \leftarrow a \to n_2$ pentru un $b \in T$;avem $b \stackrel{*}{\to} n_1$, $b \to n_2$, deci $b \in U$ și $a \to b$
- $n_1 \stackrel{*}{\leftarrow} b \leftarrow a \to c \stackrel{*}{\to} n_2$ implică $b \stackrel{*}{\to} n_3 \stackrel{*}{\leftarrow} c$, unde n_3 este formă normală; deoarece $n_1 \neq n_2$ avem $n_1 \neq n_3$ ($b \in U$) sau $n_2 \neq n_3$ $(c \in U)$. $\square$

Exercițiu

$T = \{a, b, c, d\}$

$\to = \{(a, b), (a, c), (b, a), (b, d)\}$

Putem descrie sistemul de rescriere $(T, \to)$ prin
$R = \{a \to b, a \to c, b \to a, b \to d\}$
Vom identifica $R = (T, \to)$

- arătați că R e local confluent
- arătați că R nu e confluent
- arătați că R nu e noetherian
- determinați formele normale ale lui R
- adăugați o regulă de rescriere a.î. R să devină confluent
- ștergeți o regulă de rescriere a.î. R să devină confluent

Teorema

(S, Σ) signatură, X mulțime de variabile,
E mulțime de ecuații necondiționate,
R_E sistemul de rescriere determinat de E, $\rightarrow_E \subseteq T_\Sigma(X) \times T_\Sigma(X)$ relația de rescriere
$t, t' \in T_\Sigma(X)_s$

Teoremă 42

Dacă R_E este complet atunci sunt echivalente:
(1) $E \models (\forall X) t \doteq_s t'$
(2) $E \vdash (\forall X) t \doteq_s t'$
(3) $t \overset{*}{\leftrightarrow}_E t'$
(4) $fn(t) = fn(t')$
(t și t' au aceeași formă normală)

Dem. (1)$\Leftrightarrow$(2)$\Leftrightarrow$(3) din Teorema 34.

Continuare

(4)$\Rightarrow$(3) $t \overset{*}{\rightarrow} fn(t) \overset{*}{\leftarrow} t'$

(3)$\Rightarrow$(4) Fie $t_1, \ldots, t_n$ a.î. $t \leftrightarrow t_1 \leftrightarrow \cdots \leftrightarrow t_n \leftrightarrow t'$

Demonstrăm că $fn(t) = fn(t')$ prin inducție după n.

$P(1)$ Avem $t \leftrightarrow t'$, adică $t \rightarrow t'$ sau $t' \rightarrow t$. Rezultă $fn(t) + fn(t')$.

$P(n) \Rightarrow P(n+1)$ Avem $t \overset{*}{\leftrightarrow} t_{n+1} \leftrightarrow t'$.

Din ipoteza de inducție, $fn(t) = fn(t_{n+1})$.

Din $P(1)$ rezultă $fn(t_{n+1}) = fn(t')$.

În consecință $fn(t) = fn(t')$. $\square$

Logica ecuațională și rescrierea

Terminarea unui sistem de rescriere este nedecidabilă.

Pentru sisteme de rescriere particulare putem decide asupra terminării.

Pentru sisteme de rescriere noetheriene, confluența este decidabilă.

Dacă E este o mulțime de ecuații a.î. R_E este un sistem de rescriere complet atunci deducția ecuațională $E \vdash (\forall X) t \doteq_s t'$ este decidabilă:

- $t \overset{*}{\rightarrow} fn(t)$
- $t' \overset{*}{\rightarrow} fn(t')$
- $E \vdash (\forall X) t \doteq_s t' \Leftrightarrow fn(t) = fn(t')$

Rescrierea termenilor
Terminare. Confluență. Completare

(S, Σ) signatură

- O regulă de rescriere este formată dintr-o mulțime de variabile Y și doi termeni l, r de același sort din $T_\Sigma(Y)$ a.î.
  - l nu este variabilă
  - $Var(r) \subseteq Var(l) = Y$.

  Vom nota $l \to r$ ($l \to_s r$).
- Un sistem de rescriere (TRS) este o mulțime finită de reguli de rescriere.
  Exemplu: $R = \{x + 0 \to x, x + succ(y) \to succ(x + y)\}$
- prin *lungimea* unui termen t vom înțelege numărul de simboluri din scrierea lui t în forma prefix.

(S, Σ) signatură, R un TRS

Propoziție 43

Dacă fiecărui termen t îi poate fi asociat un număr natural $t \mapsto \mu(t) \in \mathbb{N}$ astfel încât

$$t \to_R t' \Rightarrow \mu(t) > \mu(t')$$

oricare t și t', atunci R este noetherian.

Dem. $\mathbb{N}$ nu conține lanțuri infinite $n_1 > n_2 > \cdots > n_k > \cdots$.

Exemple

- $R = \{x - 0 \to x, succ(x) - succ(y) \to x - y\}$ noetherian
  $\mu(t) :=$ lungimea lui t
- $R = \{f(g(x), y) \to f(y, y)\}$ nu este noetherian
  $f(g(x), g(x)) \to_R f(g(x), g(x)) \to_R \cdots$

(S, Σ) signatură, R un sistem de rescriere (TRS)

- Arborele de reducere al termenului t este definit astfel:
  • rădăcina arborelui are eticheta t,
  • descendenții nodului cu eticheta u sunt etichetați cu termenii u' care verifică $u \to_R u'$.
- Orice nod al unui arbore de reducere are un număr finit de descendenți deoarece R este o mulțime finită.
- Dacă R se termină atunci
  $\mu(t) :=$ înălțimea arborelui de reducere asociat lui t.
  $t \to_R t' \Rightarrow \mu(t) > \mu(t')$

Arbore de reducere

$R = \{x + 0 \to x, x + succ(y) \to succ(x + y)\}$

$$\begin{tikzcd}
 & succ(0) + succ(0+0) \arrow[dl] \arrow[dr] & \\
succ(0) + succ(0) \arrow[d] & & succ(succ(0) + (0+0)) \arrow[d] \\
succ(succ(0) + 0) \arrow[d] & & succ(succ(0) + 0) \arrow[d] \\
succ(succ(0)) & & succ(succ(0))
\end{tikzcd}$$

Terminare

(S, Σ) signatură, R un TRS

Propoziția 44

Sunt echivalente:
(a) R este noetherian
(b) oricărui termen t îi poate fi asociat un număr natural $\mu(t) \in \mathbb{N}$ astfel încât $t \to_R t'$ implică $\mu(t) > \mu(t')$

Dem. (a)⇒(b) Din Lema lui König rezultă că într-un sistem de rescriere noetherian orice termen are un arbore de reducere finit și definim $\mu(t) =$ *adâncimea arborelui asociat lui* t.
(b)⇒(a) rezultă din Propoziția 43. □

Ordine de reducere: $t > t'$ ddacă $\mu(t) > \mu(t')$

Relația $>$ se numește **ordine de reducere** pentru R.

Terminare

(S, Σ) signatură, R un sistem de rescriere (TRS)

Propoziția 45 *

Fie A o (S, Σ)-algebră astfel încât:

- $A_s = \mathbb{N}$ or. $s \in S$,
- or. $\sigma : s_1 \dots s_n \to s$, dacă $k_i > k_i'$ atunci $A_\sigma(k_1, \dots, k_i, \dots k_n) > A_\sigma(k_1, \dots, k_i', \dots k_n)$,
- $\tilde{\mathbf{e}}(l) > \tilde{\mathbf{e}}(r)$ or. $l \to r \in R$ or. $\mathbf{e} : Var(l) \to A$.

Atunci R este noetherian

Dem. Pentru orice termen t definim
$\mu(t) = \tilde{\mathbf{e}}_0(t)$, unde $\mathbf{e}_0(x) = 0$ or. $x \in Var(t)$.
Se demonstrează că $t \to_R t'$ implică $\mu(t) > \mu(t')$.

Exemplu

$R = \{x + 0 \to 0, x + succ(y) \to succ(x + y)\}$
$A_0 := 1$, $A_{succ}(k) := k + 1$,
$A_+(k, m) := k + 2 * m$
$(k +_A m := k + 2 * m)$ or. k, $m \in \mathbb{N}$

$\mathbf{e}(x) := n$, $\mathbf{e}(y) := m$

$\tilde{\mathbf{e}}(x + 0) = n +_A 0 = n + 2 * A_0 = n + 2 > 1 = A_0 = \tilde{\mathbf{e}}(0)$

$\tilde{\mathbf{e}}(x + succ(y)) = n +_A A_{succ}(m) = n + 2 * (m + 1) =$
$n + 2 * m + 2 > n + 2 * m + 1 = A_{succ}(n +_A m) = \tilde{\mathbf{e}}(succ(x + y))$

Confluență. Perechi critice

(S, Σ) signatură, R un sistem de rescriere (TRS)

Fie $l_1 \to r_1$, $l_2 \to r_2 \in R$ astfel încât:

- $Var(l_1) \cap Var(l_2) = \emptyset$,
- $l_1 = c[z \leftarrow t]$ unde $nr_z(c) = 1$, t nu este variabilă (t este subtermen al lui l_1 care nu este variabilă),
- θ c.g.u pentru t și l_2 ($\Rightarrow \theta(t) = \theta(l_2)$).

Atunci $(\theta(r_1), \theta(c)[z \leftarrow \theta(r_2)])$ este pereche critică.

Confluență. Perechi critice

$$\begin{tikzcd}
 & \theta(l_1) \arrow[dl] \arrow[dr] & \\
\theta(r_1) & & \theta(c)[z \leftarrow \theta(r_2)]
\end{tikzcd}$$

$l_1 \to r_1$, $l_2 \to r_2 \in R$,
$Var(l_1) \cap Var(l_2) = \emptyset$,
$l_1 = c[z \to t]$,
$\theta(t) = \theta(l_2)$ c.g.u

*Teorema 46 (Teorema Perechilor Critice)**

Dacă R este noetherian atunci sunt echivalente:
(a) R este confluent,
(b) $t_1 \downarrow_R t_2$ oricare (t_1, t_2) pereche critică.

Exemplu

$R = \{f(f(x)) \to x\}$ este confluent.
Determinăm perechile critice $f(f(x)) \to x$ și $f(f(y)) \to y$
$l_1 := f(f(x))$, $r_1 := x$, $l_2 := f(f(y))$, $r_2 := y$
Subtermenii lui l_1 care nu sunt variabile sunt $f(f(x))$ și $f(x)$.

- $t := f(f(x))$, $c = z$, $\theta := \{x \leftarrow y\}$
  $\theta(r_1) = y$, $\theta(c)[z \leftarrow \theta(r_2)] = y$
- $t := f(x)$, $c = f(z)$, $\theta := \{x \leftarrow f(y)\}$
  $\theta(r_1) = f(y)$, $\theta(c)[z \leftarrow \theta(r_2)] = f(y)$

Perechile critice sunt (y, y) și $(f(y), f(y))$. Deoarece $y \downarrow y$ și $f(y) \downarrow f(y)$, sistemul de rescriere R este confluent.

Completarea unui TRS (Knuth-Bendix)

(S, Σ) signatură
Intrare. R un sistem de rescriere (TRS) noetherian.
Inițializare. $T := R$, $>$ ordine de reducere pentru T
Se execută următorii pași, cât timp este posibil:
(1) $CP := CP(T) = \{(t_1, t_2) | (t_1, t_2)$ pereche critică în $T\}$
(2) Dacă $t_1 \downarrow t_2$ oricare $(t_1, t_2) \in CP$,
  atunci STOP (T este completarea lui R).
(3) Dacă $(t_1, t_2) \in CP$, $t_1 \not\downarrow t_2$, atunci
  dacă $fn(t_1) > fn(t_2)$
  atunci $T := T \cup \{fn(t_1) \to fn(t_2)\}$,
  altfel dacă $fn(t_2) > fn(t_1)$
    atunci $T := T \cup \{fn(t_2) \to fn(t_1)\}$,
    altfel STOP (completarea eșuată).
Ieșire. T complet (completarea lui R) sau eșec.

Atenție: succesul completării depinde de relația $<$

Exemplu

$S := \{s\}$, $\Sigma := \{* : ss \to s\}$,
$E := \{\forall\{x, y, v\}(x * y) * (y * v) \stackrel{\cdot}{=} y)\}$
$T = R_E := \{(x * y) * (y * v) \to y\}$, $\mu(t) :=$ lungimea termenului t
Determinăm perechile critice pentru
$l_1 := (x * y) * (y * v)$, $r_1 := y$, $l_2 := (x' * y') * (y' * v')$, $r_2 := y'$
Subtermenii lui l_1 care nu sunt variabile sunt $(x * y)$ și $y * v$.

- $t := x * y$, $c = z * (y * v)$, $\theta := \{x \leftarrow x' * y', y \to y' * v'\}$
  $\theta(r_1) = y' * v'$, $\theta(c)[z \leftarrow \theta(r_2)] = y' * ((y' * v') * v)$
- $t := y * v$, $c = (x * y) * z$, $\theta := \{y \to x' * y', v \to y' * v'\}$
  $\theta(r_1) = x' * y'$, $\theta(c)[z \leftarrow \theta(r_2)] = (x * (x' * y')) * y'$

$CP = \{(y' * v', y' * ((y' * v') * v)), (x' * y', (x * (x' * y')) * y')\}$

Deoarece $y * ((y * x) * v) > y * x$ și $(x * (v * y)) * y > v * y$,
$T := T \cup \{y * ((y * x) * v) \to y * x, \ (x * (v * y)) * y \to v * y\}$

Demonstrați că T este complet.

Logica ecuatională locală


```
fmod MYNAT is
  sort    MyNat .
  op 0 : -> MyNat .
  op s : MyNat -> MyNat .
  op _+_ : MyNat MyNat -> MyNat .
  op n : -> MyNat .
  vars  x y z  : MyNat .
  eq x + 0 = x .
  eq x + s(y) = s(x + y) .
  eq (x + y) + z = x + ( y  + z ) .
  endfm

reduce in MYNAT : n + (0 + n) .
result MyNat: n + (0 + n)
```



```
fmod MYNAT1 is
  sort    MyNat .
  op 0 : -> MyNat .
  op s : MyNat -> MyNat .
  op _+_ : MyNat MyNat -> MyNat [assoc] .
  op n : -> MyNat .
  vars  x y  : MyNat .
  eq x + 0 = x .
  eq x + s(y) = s(x + y) .
  endfm

reduce in MYNAT1 : n + (0 + n) .
result MyNat: n + n
```

Termenii sunt unificați modulo asociativitate

- Semantically, declaring a set of equational attributes for an operator is equivalent to declaring the corresponding equations for the operator.
- Operationally, using equational attributes to declare such equations avoids termination problems and leads to much more efficient evaluation of terms containing such an operator.
- The effect of declaring equational attributes is to compute with equivalence classes modulo such equations.

Punctul de vedere local

(S,Σ) signatură multisortată

- O ecuatie $(\forall X)t \doteq_s t'$ este o pereche de termeni de acelasi sort din algebra $T_\Sigma(X)$.
- Fie F mulţime de ecuaţii necondiţionate. O ecuaţie modulo F are forma $(\forall X)[t]_F \doteq_s [t']_F$, unde $[t]_F$, $[t']_F \in T_\Sigma(X)/_{\equiv_F}$. În acest caz, o ecuatie este o pereche de termeni de acelasi sort din algebra cat $T_\Sigma(X)/_{\equiv_F}$.

  Pentru un modul in Maude, F este mulţimea ecuaţiilor declarate ca atribute.
- Putem inlocui termenii cu elemente dintr-o algebra fixata A. Astfel, o ecuatie(propozitie) din A va avea forma $a \doteq_s c$, unde a, $c \in A_s$. Acesta este punctul de vedere local al logicii ecuationale.

V.E. Căzănescu, Local Equational Logic,
FUNDAMENTALS OF COMPUTATION THEORY,
LNCS 710 (1993), 162-170.

Ecuaţiile(propoziţiile) din A

clasic	$(\forall X)t \doteq_s t'$	$t,\ t' \in T_\Sigma(X)_s$
local	$a \doteq_s c$ $(\forall A)a \doteq_s c$	$a,\ c \in A_s$, A fixata

În cele ce urmează

- A va fi o (S,Σ)-algebra fixata.
- $Sen(A) := \{a \doteq_s c | a, c \in A_s, s \in S\}$ propozitiile din A

Vom defini sintaxa si semantica logicii locale asociate lui A.

Regulile deducţiei ecuaţionale în A

$a,\ b,\ c,\ a_i,\ b_i \in A$

R $\quad \dfrac{}{a \doteq_s a}$

S $\quad \dfrac{a \doteq_s b}{b \doteq_s a}$

T $\quad \dfrac{a \doteq_s b,\ \ b \doteq_s c}{a \doteq_s c}$

C_Σ $\quad \dfrac{a_1 \doteq_{s_1} b_1, \ldots, a_n \doteq_{s_n} b_n}{A_\sigma(a_1,\ldots,a_n) \doteq_s A_\sigma(b_1,\ldots,b_n)}$ $\qquad \sigma : s_1 \cdots s_n \to s$

Sub_Γ $\quad \dfrac{\{h_{s'}(u) \doteq_{s'} h_{s'}(v) | u \doteq_{s'} v \in H\}}{h_s(t) \doteq_s h_s(t')}$ $\qquad h : T_\Sigma(X) \to A$, $(\forall X)t \doteq_s t'$ *if* $H \in \Gamma$

Deducţia sintactică in A

Spunem că ecuaţia $\epsilon := a \doteq_s a' \in Sen(A)$ se deduce sintactic din Γ dacă există o secvenţă de ecuaţii $\epsilon_1, \ldots, \epsilon_n$ a. î. $\epsilon_n = \epsilon$ şi pentru orice $i \in \{1,\ldots,n\}$

- $\epsilon_i \in \Gamma$ sau
- ϵ_i se obţine din ecuaţiile $\epsilon_1, \ldots, \epsilon_{i-1}$ aplicând una din regulile R, S, T, C_Σ, Sub_Γ.

În acest caz spunem că ϵ este Γ -teoremă a lui A. Secvenţa $\epsilon_1, \ldots, \epsilon_n = \epsilon$ este o demonstraţie în A pentru $\mathbf{e}$ din ipotezele Γ. Vom nota $\Gamma \vdash_A a \doteq_s a'$

Exemplu

$S = \{s\}$, $\Sigma := \{0 :\to s, + : ss \to s\}$
$A = (\mathbb{N}, 0, +)$
$\Gamma := \{(\forall\{x\})x + x \doteq 0\}$
Demonstrati ca $\Gamma \vdash_A 3 \doteq 1$

(1) $h(x+x) \doteq h(0)$ (Sub$_\Gamma$, $h : T_\Sigma(\{x\}) \to A$, $h(x) := 1$)
$2 \doteq 0$

(2) $1 \doteq 1$ (R)

(3) $2 + 1 \doteq 0 + 1$ ((1),(2), C$_\Sigma$)
$3 \doteq 1$

Închiderea la reguli de deducție

(S, Σ) o signatură multisortată, X o mulțime de variabile

Reg $\boxed{\dfrac{a_1 \doteq_{s_1} a'_1, \ldots, a_n \doteq_{s_n} a'_n}{a \doteq_s a'}}$

Relație binară $Q \subseteq A \times A$ este închisă la **Reg** dacă

$$(a_1, a'_1) \in Q_{s_1},\ \ldots,\ (a_n, a'_n) \in Q_{s_n} \Rightarrow (a, a') \in Q_s$$

Γ-tautologiile lui A

Γ o multime de ecuatii conditionate
(de forma $(\forall X)t \doteq_s t'$ *if* H).

O ecuație $a \doteq_s a' \in Sen(A)$ este Γ-tautologie dacă
$h_s(a) = h_s(a')$ or. $B \models \Gamma$, or. $h : A \to B$ morfism.

Vom nota $\Gamma \models_A a \doteq_s a'$

$\equiv^A_\Gamma := \bigcap\{\ker(h) | h : A \to B \models \Gamma\}$
congruenta semantica a lui A

Propozitia 47

$\equiv^A_\Gamma$ este cea mai mica congruenta pe A, inchisa la Γ-substitutie.

Dem. exercițiu

Teorema de completitudine

- $a \sim^A_\Gamma a' \Leftrightarrow \Gamma \vdash_A a \doteq_s a'$ (echiv. sintactică)
- $a \equiv^A_\Gamma a' \Leftrightarrow \Gamma \models_A a \doteq_s a'$ (echiv. semantica)
- Corectitudinea logicii locale: $\sim^A_\Gamma \subseteq \equiv^A_\Gamma$
- Completitudinea regulilor de deducție: $\equiv^A_\Gamma \subseteq \sim^A_\Gamma$

Teorema 48 (Teorema de completitudine a logicii locale)

$\equiv^A_\Gamma = \sim^A_\Gamma$

Dem. exercițiu

Contexte

(S,Σ)-signatura, A (S,Σ)-algebra
Fie $A[z] := T_\Sigma(A \cup \{z\})$, $z \notin A$
Un context este un termen $c \in A[z]$ cu $nr_z(c) = 1$, iar
$c[a] := c[z \leftarrow a]$

$A = (\mathbb{N}, 0, +)$
$c = 0 + z$
$c = 4 + (z + 1000)$

Inchiderea la contexte

A (S,Σ)-algebra, $Q \subseteq A \times A$

Q este închisă la contexte dacă
$(a, a') \in Q \Rightarrow (c[a], c[a']) \in Q$ or. $c \in A[z]$ context

Propoziția 49

Sunt echivalente:
(a) Q este închisă la contexte,
(b) Q este compatibilă pe componente, i.e.
or. $\sigma : s_1 \ldots s_n \to s$, $i \in \{1, \ldots, n\}$,
or. $a_1 \in A_{s_1}$, ..., $a_{i-1} \in A_{s_{i-1}}$, $a_{i+1} \in A_{s_{i+1}}$, ..., $a_n \in A_{s_n}$,
or. $(a, a') \in Q_{s_i}$,
$(A_\sigma(a_1, \ldots, a_{i-1}, a, a_{i+1}, \ldots, a_n), A_\sigma(a_1, \ldots, a_{i-1}, a', a_{i+1}, \ldots, a_n)) \in Q_s$.

Continuare

Dem. (a) $\Rightarrow$ (b) Fie $\sigma : s_1 \ldots s_n \to s$, $i \in \{1, \ldots, n\}$, $(a, a') \in Q_{s_i}$ și
$a_1 \in A_{s_1}$, ..., $a_{i-1} \in A_{s_{i-1}}$, $a_{i+1} \in A_{s_{i+1}}$, ..., $a_n \in A_{s_n}$.
Pentru $c := A_\sigma(a_1, \ldots, a_{i-1}, z, a_{i+1}, \ldots, a_n) \in A[z]$ se obține rezultatul dorit.
(b) $\Rightarrow$ (a) Fie $(a, a') \in Q$ și $c \in A[z]$ un context.
Demonstrăm $(c[a], c[a']) \in Q$ prin inducție după lungimea lui c (termen în $T_\Sigma(A \cup \{z\})$).
Dacă $|c| = 1$, atunci $c = z$, $c[a] = a$, $c[a'] = a'$.
Presupunem $(c'[a], c'[a']) \in Q$ pentru $c' \in A[z]$ context, $|c'| < |c|$.
Observăm că există c' context, $|c'| < |c|$ a.î.
$c = A_\sigma(a_1, \ldots, a_{i-1}, c', a_{i+1}, \ldots, a_n)$, deci
$c[a] = A_\sigma(a_1, \ldots, a_{i-1}, c'[a], a_{i+1}, \ldots, a_n)$ și
$c[a'] = A_\sigma(a_1, \ldots, a_{i-1}, c'[a'], a_{i+1}, \ldots, a_n)$.
Aplicăm compatibilitatea pe argumente și ipoteza de inducție. $\square$

Inchiderea la contexte

A (S,Σ)-algebra, $Q \subseteq A \times A$

$\to_Q := \{(c[a], c[a']) | (a, a') \in Q, c \in A[z] \text{ context}\}$

$\stackrel{*}{\to}_Q$ închiderea reflexivă și tranzitivă

$\downarrow_Q := \{(a, a') |$ ex. $b \in A$ a. î. $a \stackrel{*}{\to}_Q b$ și $a' \stackrel{*}{\to}_Q b\}$

Propoziția 50 *

(a) $\to_Q$ este cea mai mica relație închisă la contexte care include Q.
(b) $\stackrel{*}{\to}_Q$ este cea mai mica preordine închisă la C_Σ care include Q.

Rescrierea locala

A este (S,Σ)-algebra fixata
Γ multime de ecuatii conditionate

Definim $(Q_n)_n$:
$Q_n \subseteq A \times A$ or. n,
$Q_0 := \emptyset$,
$Q_{n+1} := \Big\{(h_s(l), h_s(r)) | (\forall Y) l \doteq_s r \textit{ if } H \in \Gamma,$
$h : T_\Sigma(Y) \to A \textit{ morfism},$
$h_{s'}(u) \downarrow_{Q_n} h_{s'}(v) \textit{ or. } u \doteq_{s'} v \in H\Big\}$
$Q := \bigcup_n Q_n$

Notam $\Rightarrow_{\Gamma,A} := \to_Q$, $\Downarrow_{\Gamma,A} := \downarrow_Q$

Rescrierea locala

A este (S,Σ)-algebra fixata
Γ multime de ecuatii conditionate, $a, a' \in A_s$

$a \to_{\Gamma,A} a' \quad \Leftrightarrow \quad$ ex. $c \in A[z]$, $nr_z(c) = 1$
a este $c[z \leftarrow h_s(l)]$ și b este $c[z \leftarrow h_s(r)]$,
$(\forall Y) l \doteq_s r \textit{ if } H \in \Gamma$,
$h : T_\Sigma(Y) \to A$ este un morfism,
$h_{s'}(u) \Downarrow_{\Gamma,A} h_{s'}(v)$ or. $u \doteq_{s'} v \in H$

Teorema 51 *

$\overset{*}{\to}_{\Gamma,A} \; = \; \overset{*}{\Rightarrow}_{\Gamma,A} \; \subseteq \; \equiv^A_\Gamma$

Rescrierea locala

A este (S,Σ)-algebra fixata

Relația $\succ A \times A$ este cofluentă dacă or. a, b, $c \in A$

$$a \succ b \text{ și } a \succ c \Rightarrow \text{ex. } d \in A \text{ a.î. } c \succ d \text{ și } b \succ d$$

Teorema 52 *

Fie Γ o multime de ecuatii conditionate a.î. $\overset{*}{\to}_\Gamma$ este confluenta.
Atunci $\downarrow_\Gamma \; = \; \equiv^A_\Gamma$, i.e.

$$a \equiv^A_\Gamma a' \Leftrightarrow a \downarrow_\Gamma a'$$

Maude

```
fmod RESLOC is
including  NAT .
op _++_ : Nat Nat -> Nat .
op rel : Nat Nat -> Bool .
vars x y :  Nat .
eq x  ++  y =  x + y + x * y .
ceq rel(x,y) = true if   x ++ y =   4 * x   .
ceq rel(x , y) = false if x ++ y =/=  4 * x   .
endfm

set trace on .
reduce rel (2 , 2) .
```

```
reduce in RESLOC : rel(2, 2) .
*********** trial #1
ceq rel(x, y) = true if x ++ y = x * 4 .
x --> 2
y --> 2
*********** solving condition fragment
x ++ y = x * 4
*********** equation
eq x ++ y = x + y + x * y .
..........
*********** success for condition fragment
x ++ y = x * 4
x --> 2
y --> 2
*********** success #1
```

$h(x++y) = 8,\ h(x*4) = 8,\ (8,8) \in Q_1$


```
*********** success for condition fragment
x ++ y = x * 4
x --> 2
y --> 2
*********** success #1
*********** equation
ceq rel(x, y) = true if x ++ y = x * 4 .
x --> 2
y --> 2
rel(2, 2)
--->
true
rewrites: 6 in 7967112414ms cpu (43ms real)
result Bool: true
```

$h(x++y) \downarrow_{Q_1} h(x*4)$, deci $(rel(2,2), true) = (h(rel(x,y)), true) \in Q_2$

Deductie si rescriere modulo axiome


```
fmod MYNAT1 is
  sort    MyNat .
  op 0 : -> MyNat .
  op s : MyNat -> MyNat .
  op _+_ : MyNat MyNat -> MyNat [assoc] .
  op n : -> MyNat .
  vars  x y  : MyNat .
  eq x + 0 = x .
  eq x + s(y) = s(x + y) .
  endfm
reduce in MYNAT1 : n + (0 + n) .
result MyNat: n + n
```

Termenii sunt unificați modulo asociativitate

(S,Σ) signatură multisortată, E mulțime de ecuații,
F mulțime de ecuații necondiționate (axiome)

Exemplu:

```
fmod MYNAT1 is
...
endfm
```

$E := \{(\forall\{x\})x + 0 \doteq_{MyNat} x,\ (\forall\{x,y\})x + s(y) \doteq_{MyNat} s(x+y)\}$
$F := \{\forall\{x,y,z\}(x+y)+z \doteq_{MyNat} x+(y+z)\}$

(S,Σ) signatură multisortată,
F mulțime de ecuații necondiționate (axiome)

Dacă X este o mulțime de variabile, definim
$\equiv_F := \bigcap\{Ker(h) \mid h : T_\Sigma(X) \to B \models F\}$
(echivalența semantică).

Pentru $t \in T_\Sigma(X)_s$ notăm
$[t]_F := [t]_{\equiv_F}$ clasa de echivalență a lui t.
$T_{\Sigma,F}(X) := T_\Sigma(X)/_{\equiv_F}$

F-algebra liber generată

Propoziția 53

$T_{\Sigma,F}(X)$ este F-algebra liber generată de X, i.e.

or. $B \models F$, or. $\mathbf{e} : X \to B$, există un unic morfism $\overline{\mathbf{e}} : T_{\Sigma,F}(X) \to B$ cu $\overline{\mathbf{e}}([x]_F) = \mathbf{e}(x)$ or. $x \in X$, și anume

$$\overline{\mathbf{e}}([t]_F) = \tilde{\mathbf{e}}(t) \text{ or. } t \in T_\Sigma(X)$$

Dem. Se aplică proprietatea de universalitate a mulțimii cat:

$$\begin{tikzcd} T_\Sigma(X) \arrow[r, "{[\cdot]_F}"] \arrow[d, "\tilde{\mathbf{e}}"] & T_{\Sigma,F}(X) \arrow[dl, dotted, "\overline{\mathbf{e}}"] \\ B & \end{tikzcd}$$

Exercițiu: clarificați detaliile acestei demonstrații. $\square$

Ecuații și satisfacere modulo F

(S, Σ) signatură multisortată,
F mulțime de axiome, E mulțime de ecuații

- O ecuație modulo F are forma $(\forall X)[t]_F \doteq_s [t']_F$, unde $t, t' \in T_\Sigma(X)_s$.
- Pentru B o F-algebră definim satisfacerea modulo F:

  $B \models_F (\forall X)[t]_F \doteq_s [t']_F \Leftrightarrow \overline{\mathbf{e}}([t]_F) = \overline{\mathbf{e}}([t']_F)$ or. $\mathbf{e} : X \to B$.

  In acest caz vom nota $B \models_F (\forall X)[t]_F \doteq_s [t']_F$.
- Notam $E \models_F (\forall X)[t]_F \doteq_s [t']_F$ daca or. B o F-algebra $B \models_F (\forall Y)[t_1]_F \doteq_{s'} [t_2]_F$ or. $(\forall Y)t_1 \doteq_{s'} t_2 \in E$ implică $B \models_F (\forall X)[t]_F \doteq_s [t']_F$

Deducție din E modulo F

(S, Σ) signatură multisortată, E mulțime de ecuații,
F mulțime de axiome

Propoziția 54

Dacă $B \models F$ atunci sunt echivalente:
(a) $B \models (\forall X)t \doteq_s t'$
(b) $B \models_F (\forall X)[t]_F \doteq_s [t']_F$.

Teoremă 55

Sunt echivalente:
(a) $E \cup F \models (\forall X)t \doteq_s t'$
(b) $E \models_F (\forall X)[t]_F \doteq_s [t']_F$

Dem. Consecință directă a Propoziției 54.

Continuare (Propoziția 54)

Dem. Fie $B \models F$.

(a) $\Rightarrow$ (b) Dacă $\mathbf{e} : X \to B$, atunci $\tilde{\mathbf{e}}(t) = \tilde{\mathbf{e}}(t')$ din ipoteză.

$$\overline{\mathbf{e}}([t]_F) = \tilde{\mathbf{e}}(t) = \tilde{\mathbf{e}}(t') = \overline{\mathbf{e}}([t']_F)$$

(b) $\Rightarrow$ (a) Dacă $\mathbf{e} : X \to B$, atunci $\overline{\mathbf{e}}([t]_F) = \overline{\mathbf{e}}([t']_F)$ din ipoteză.

$$\tilde{\mathbf{e}}(t) = \overline{\mathbf{e}}([t]_F) = \overline{\mathbf{e}}([t']_F) = \tilde{\mathbf{e}}(t') \qquad \square$$

Rescrierea modulo axiome

(S, Σ) signatură multisortată,
R sistem de rescriere,
F mulțime de axiome

Dacă $t,\ t' \in T_\Sigma(X)_s$ definim relația $[t]_F \to_{R/F} [t']_F$ astfel:

$$[t]_F \to_{R/F} [t']_F \quad \Leftrightarrow \quad \begin{array}{l} t \equiv_F c[z \leftarrow \theta(l)] \text{ și} \\ t' = c[z \leftarrow \theta(r)], \text{ unde} \\ c \in T_\Sigma(X \cup \{z\}),\ z \notin X,\ nr_z(c) = 1 \\ l \to r \in R \text{ cu } Var(l) = Y, \\ \theta : Y \to T_\Sigma(X) \text{ este o substituție.} \end{array}$$

$[t]_F \to_{R/F} [t']_F \Leftrightarrow$ ex. t_0 ($t \equiv_F t_0$ si $t_0 \to_R t'$)

Exemplu

```
fmod MYNAT1 is
  sort    MyNat .
  op 0 : -> MyNat .
  op s : MyNat -> MyNat .
  op _+_ : MyNat MyNat -> MyNat [assoc] .
  vars  x y  : MyNat .
  eq x + 0 = x .
  eq x + s(y) = s(x + y) .
  endfm
```

$R := \{x + 0 \to x,\ x + s(y) \to s(x + y)\}$
$F := \{\forall\{x, y, z\}(x + y) + z \doteq_{MyNat} x + (y + z)\}$

$s(0) + (0 + s(0)) \equiv_F (s(0) + 0) + s(0) \to_R s(0) + s(0) \to_R s(s(0))$

Rescrierea modulo axiome

(S, Σ) signatură multisortată, F mulțime de axiome,
E mulțime de ecuații, R_E sistemul de rescriere asociat,
Notăm $\to_{E/F} := \to_{R_E/F}$

Teorema 56 *

Sunt echivalente:
(1) $E \models_F (\forall X)[t]_F \doteq_s [t']_F$
(2) $t \overset{*}{\leftrightarrow}_{E \cup F} t'$
(3) $[t] \overset{*}{\leftrightarrow}_{E/F} [t']_F$

$(T_{\Sigma,F}(X)_s, \to_{E/F})$ este un sistem de rescriere abstract, pentru care proprietatile de confluenta, terminare si completare se definesc uzual. Algoritmul de completare necesita unificare modulo ecuatii, care este nedecidabilă în general.

Programare Logică

Semantica algebrei inițiale

Algebra inițială

(S, Σ) signatură

Teoremă 2

Pentru orice (S, Σ)-algebră A există un unic morfism $\mathcal{S}_A : T_\Sigma \to A$. $\mathcal{S}_A(t)$ este interpretarea termenului t în A.

Exemplu

- $D = (D_S, D_\Sigma)$, $D_s := \mathbb{N}$ or. $s \in S$
  dacă $\sigma :\to s$ atunci $D_\sigma := 0$,
  dacă $\sigma : s_1 \ldots s_n \to s$, $k_1, \ldots, k_n \in \mathbb{N}$
  $D_\sigma(k_1, \ldots, k_n) := 1 + max(k_1, \ldots, k_n)$
- $\mathcal{S}_D : T_\Sigma \to D$ unicul morfism

  dc. $t = \sigma(t_1, \ldots, t_n)$ at. $arb(t) :=$ $\begin{array}{ccc} & \sigma & \\ \swarrow & & \searrow \\ arb(t_1) & \cdots & arb(t_n) \end{array}$
- $\mathcal{S}_D(t) =$ adîncimea $arb(t)$

Gramatici independente de context

Vom considera gramatici independente de context neambigue.

- Gramaticii $G = (S_0, N, T, P)$ îi asociem signatura $\mathcal{G} = (S = N, \Sigma = P)$.
- Definim $\mathcal{G}$-algebra L_G astfel încât $\mathcal{S}_{G,S_0}(T_{\Sigma,S_0}) = L(G)$, unde $L(G)$ este limbajul definit de G, iar $\mathcal{S}_G : T_\Sigma \to L_G$ este unicul $\mathcal{G}$-morfism. Deoarece G este neambiguă, morfismul $\mathcal{S}_G$ este injectiv.
- Pentru $w \in L(G)$ exista un unic $t \in T_{\Sigma S_0}$ astfel încât $\mathcal{S}_G(t) = w$. Vom scrie $t_w = \mathcal{S}_G^{-1}(w)$.
- Pentru orice $\mathcal{G}$-algebră $\mathcal{A}$, unicul $\mathcal{G}$-morfism $\mathcal{S}_A : T_\Sigma \to \mathcal{A}$ îi asociază lui t_w o interpretare în $\mathcal{A}$, și anume $\mathcal{S}_A(t_w)$.
- $Sem(w) = \mathcal{S}_\mathcal{A}(t_w) = \mathcal{S}_\mathcal{A}(\mathcal{S}_G^{-1}(w))$ oricare $w \in L(G)$

Gramatici independente de context (neambigue)

- $G = (S_0, N, T, P)$
  - N este mulțimea neterminalelor
  - T este mulțimea terminalelor
  - S_0 este simbolul de start
  - $P \subseteq N \times (N \cup T)^*$ mulțimea producțiilor

$S_0 \in N$, $T \cap N = \emptyset$,
o producție $p = (A, \omega) \in P$ va fi descrisă prin

$$[p] \; A \to \omega$$

$G = (S_0, N, T, P)$

Definim signatura $\mathcal{G} = (S, \Sigma)$ astfel:

- neterminalele devin sorturi: $S = N$
- producțiile devin simboluri de operație:

$$p \in \Sigma_{n_1 \cdots n_k, n} \Leftrightarrow p = (n\,,\; t_0 n_1 t_1 \cdots n_k t_k) \in P,$$

  unde $n, n_1, \ldots, n_k \in N$ și $t_0, \ldots, t_k \in T^*$.

Observăm că

$$p : n_1 \cdots n_k \to n \Leftrightarrow [p]\ n \to t_0 n_1 t_1 \cdots n_k t_k.$$

Vom scrie simplu $\Sigma = P$.

$G = (S_0, N, T, P)$

Exemplu.

descrierea unui număr natural ca șir de cifre

Gramatica G	Signatura $\mathcal{G}$
$N = \{\langle cifra \rangle, \langle nat \rangle\}$	$S = N$
$T = \{0, \ldots, 9\}$, $S_0 = \langle nat \rangle$	
$P = \{c0, \cdots, c9, p1, p2\}$	$\Sigma = \{$
$[ci]\ \langle cifra \rangle \to i$, $i = 0, 9$	$ci :\to \langle cifra \rangle$, $i = 0, 9$
$[p1]\ \langle nat \rangle \to \langle cifra \rangle$	$p1 : \langle cifra \rangle \to \langle nat \rangle$
$[p2]\ \langle nat \rangle \to \langle nat \rangle \langle cifra \rangle$	$p2 : \langle nat \rangle \langle cifra \rangle \to \langle nat \rangle\ \}$

Algebra arborilor de derivare

Arbore de derivare pentru G:
• frunzele sunt terminale,
• nodurile interioare sunt neterminale,
• rădăcina este neterminal,
• dacă un nod are eticheta $n \in N$, iar succesorii săi sunt etichetați cu $x_1, \ldots, x_k \in N \cup T$, atunci $(n, x_1 \cdots x_k) \in P$

Observație

Mulțimea arborilor de derivare are o structura canonică de $\mathcal{G}$-algebră, *Arb*, care este izomorfă cu algebra termenilor T_Σ.

Algebra arborilor de derivare *Arb* este $\mathcal{G}$-algebră inițială.

V.E. Căzănescu, Algebră și rescriere

$G = (S_0, N, T, P)$

Am definit o signatură multisortată $\mathcal{G} = (S = N, \Sigma = P)$.

Construim o $\mathcal{G}$-algebră care va permite definirea limbajului generat de gramatică prin metoda algebrei inițiale.

- $L_G = (L_S, L_\Sigma)$ $\mathcal{G}$-algebră
  - $L_n = T^*$, oricare $n \in N$
  - $p : n_1 \cdots n_k \to n \in \Sigma$ și $(w_1, \ldots, w_k) \in (T^*)^k$

$$L_p(w_1, \ldots, w_k) = t_0 w_1 \cdots w_k t_k,$$

  unde $[p]\ n \to t_0 n_1 t_1 \cdots n_k t_k$ în P.

Limbajul $L(G)$

T_Σ este $\mathcal{G}$-algebră inițială.

Teorema 57 *

Dacă $\mathcal{S}_G : T_\Sigma \to L_G$ unicul $\mathcal{G}$-morfism atunci pentru orice neterminal $n \in N$

$$\{w \in T^* \mid n \overset{*}{\Rightarrow} w\} = \mathcal{S}_{G,n}(T_{\Sigma,n}).$$

In particular $L(G) = \mathcal{S}_{G,S_0}(T_{\Sigma,S_0})$.

Dem. $\subseteq$ inducție după lungimea derivării.
$\supseteq$ inducție structurală.

Semantica algebrei inițiale

pentru limbaje definite prin gramatici indepedente de context

$$\begin{tikzcd} T_\Sigma \arrow[r, "\mathcal{S}_G"] \arrow[d, "\mathcal{S}_A"] & L_G \\ \mathcal{A} & \end{tikzcd} \qquad \begin{tikzcd} t_w \arrow[d, "\mathcal{S}_A"] & w \arrow[l, dashed, "\mathcal{S}_G"'] \\ Sem(w) = \mathcal{S}_A(t_w) & \end{tikzcd}$$

- w sintaxa concretă, t_w sintaxa abstractă, A algebra semantică, A_s domeniu semantic or. $s \in S$ $\mathcal{S}_A(t)$ interpretarea (semantica, semnificație, denotația) termenului t
- Σ instrucțiunile, w (t_w) program, A mașină, $Sem(w) = \mathcal{S}_A(t_w)$ execuția programului w pe mașina A

Exemple

Vom prezenta următoarele aplicații ale metodei algebrei inițiale:

- semantica unui șir de cifre ca număr natural,
- semantica limbajului unui minicalculator,
- reprezentarea expresiilor în formă poloneză inversă,
- modelarea algebrică a compilării unei expresii aritmetice folosind o mașina cu stivă și acumulator.

Semantica unui șir de cifre

Semantica unui șir de cifre ca număr natural

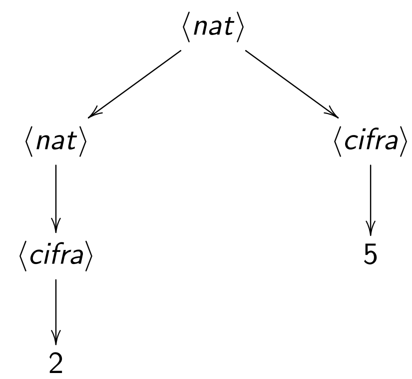
- Gramatica $G = (S_0, N, T, P)$,
$N = \{\langle cifra\rangle, \langle nat\rangle\}$, $S_0 = \langle nat\rangle$,
$T = \{0, \ldots, 9\}$, $P = \{c0, \cdots, c9, p1, p2\}$
- Signatura $\mathcal{G} = (S = N, \Sigma = P)$
$\Sigma = \{ci :\to \langle cifra\rangle \mid i = 0, 9\} \cup$
$\{p1 : \langle cifra\rangle \to \langle nat\rangle,$
$p2 : \langle nat\rangle\langle cifra\rangle \to \langle nat\rangle\}$

Semantica unui șir de cifre

Algebra semantică

- Definim $\mathcal{A} = (A_S, A_\Sigma)$ $\mathcal{G}$-algebră:
  $A_{\langle cifra \rangle} = \{0, \ldots, 9\} \subseteq \mathbb{N}$, $A_{\langle nat \rangle} = \mathbb{N}$,
  $A_{ci} = i \in A_{\langle cifra \rangle}$, $i = 0, 9$ (operație constantă)
  $A_{p1} : \{0, \ldots, 9\} \to \mathbb{N}$, $A_{p1}(i) = i$,
  $A_{p2} : \mathbb{N} \times \{0, \ldots, 9\} \to \mathbb{N}$, $A_{p2}(m, i) = m * 10 + i$
- Dacă $\mathcal{S}_A : T_\Sigma \to \mathcal{A}$ și $\mathcal{S}_G : T_\Sigma \to L_G$ sunt unicele morfisme definite pe T_Σ, atunci semantica unui sir de cifre $w \in L_G\langle nat \rangle$ este $Sem(w) = \mathcal{S}_A(\mathcal{S}_G^{-1}(w))$.

Exemplu: $Sem(\overline{25})$



$Sem(\overline{25}) = \mathcal{S}_A(T_{p2}(T_{p1}(T_{c2}), T_{c5})) =$
$A_{p2}(A_{p1}(A_{c2}), A_{c5}) = A_{p1}(A_{c2}) * 10 + 5 = A_{c2} * 10 + 5 =$
$2 * 10 + 5 = 25.$

Calculatorul de buzunar

0	1	2	3	4	+	(	IF	M	ON
5	6	7	8	9	*	)	,	E	OFF

IF 2+M,(M+45)+3,5+6 E

- M este celula de memorie
- E comanda de evaluare
- $val(IF\ e1, e2, e3) = val(e2)$ daca $val(e1) = 0$
  $val(IF\ e1, e2, e3) = val(e3)$ daca $val(e1) \neq 0$

Calculatorul de buzunar

$[ci]$ $\langle cifra \rangle \to i$, $i = 0, 9$
$[p1]$ $\langle nat \rangle \to \langle cifra \rangle$
$[p2]$ $\langle nat \rangle \to \langle nat \rangle \langle cifra \rangle$
$[r1]$ $\langle exp \rangle \to \langle nat \rangle$
$[r2]$ $\langle exp \rangle \to M$
$[r3]$ $\langle exp \rangle \to \langle exp \rangle + \langle exp \rangle$
$[r4]$ $\langle exp \rangle \to IF \langle exp \rangle, \langle exp \rangle, \langle exp \rangle$
$[r5]$ $\langle exp \rangle \to (\langle exp \rangle)$
$[I1]$ $\langle inst \rangle \to \langle exp \rangle E\ OFF$
$[I2]$ $\langle inst \rangle \to \langle exp \rangle E \langle inst \rangle$
$[Pr]$ $\langle prog \rangle \to ON \langle inst \rangle$

Calculatorul de buzunar

- La pornirea calculatorului memoria M este initializată cu 0. La apăsarea butonului E, expresia de pe ecran este evaluată folosind valoarea celulei M. Valoarea astfel obținută este afișată pe ecran și este introdusă în M.
- Un program este un șir de instrucțiuni

$$ON\ e1\ E\ e2\ E \cdots en\ E\ OFF$$

Semantica lui este șirul de numere care apare pe ecran, adică un element din $\mathbb{N}^+$.

  - $Sem(w) \in \mathbb{N}^+$, unde w este un program

Calculatorul de buzunar

Notație $A \to B := \{f \mid f : A \to B \text{ funcție}\}$

- O instrucțiune este o secvență

$$e1\ E\ e2\ E \cdots en\ E\ OFF$$

care se execută dintr-o anumită stare a memoriei. Semantica unei instrucțiuni va fi o funcție $I : \mathbb{N} \to \mathbb{N}^+$.

  - $Sem(\omega) \in \mathbb{N} \to \mathbb{N}^+$, unde ω este o instructiune

- Expresiile sunt termeni în variabila M. Pentru fiecare evaluare se folosește valoarea curentă a memoriei M.
Semantica unei expresii este o funcție $e : \mathbb{N} \to \mathbb{N}$
  - $Sem(\epsilon) \in \mathbb{N} \to \mathbb{N}$, unde ϵ este o expresie

Algebra semantică

A este o $\mathcal{G}$-algebra, unde $\mathcal{G} = (S, \Sigma)$,

$S = \{\langle cifra\rangle, \langle nat\rangle, \langle expr\rangle, \langle inst\rangle, \langle prog\rangle\}$

$A_{\langle cifra\rangle} = \{0, \ldots, 9\}$, $A_{\langle nat\rangle} = \mathbb{N}$,
$A_{\langle expr\rangle} = \mathbb{N} \to \mathbb{N} = \{e : \mathbb{N} \to \mathbb{N} \mid e \ \ functie\}$,
$A_{\langle inst\rangle} = \mathbb{N} \to \mathbb{N}^+ = \{I : \mathbb{N} \to \mathbb{N}^+ \mid I \ \ functie\}$,
$A_{\langle prog\rangle} = \mathbb{N}^+$

$\Sigma = \{c0, \ldots, c9, p1, p2, r1, \ldots, r5, I1, I2, Pr\}$

Definim operatiile A_p, cu $p \in \Sigma$.

Algebra semantică

- A_{ci}, A_{p1}, A_{p2}
- $[r1]\ \langle exp\rangle \to \langle nat\rangle$
$A_{r1} : A_{\langle nat\rangle} \to A_{\langle exp\rangle}$,
$A_{r1} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$
$A_{r1}(k) : \mathbb{N} \to \mathbb{N}$, $A_{r1}(k)(m) = k$
- $A_{r2} : \mathbb{N} \to \mathbb{N}$, $A_{r2}(m) = m$,
- $A_{r3} : A_{\langle expr\rangle} \times A_{\langle expr\rangle} \to A_{\langle expr\rangle}$,
$A_{r3}(e1, e2)(m) = e1(m) + e2(m)$,
- $A_{r4} : A_{\langle expr\rangle} \times A_{\langle expr\rangle} \times A_{\langle expr\rangle} \to A_{\langle expr\rangle}$,
$A_{r4}(e1, e2, e3)(m) = e2(m)$ daca $e1(m) = 0$,
$A_{r4}(e1, e2, e3)(m) = e3(m)$ daca $e1(m) \neq 0$,
- $A_{r5} : A_{\langle expr\rangle} \to A_{\langle expr\rangle}$, $A_{r5}(e) = e$,
- $A_{I1} : A_{\langle expr\rangle} \to A_{\langle inst\rangle}$, $A_{I1}(e)(m) = e(m)$,
- $A_{I2} : A_{\langle expr\rangle} \times A_{\langle inst\rangle} \to A_{\langle inst\rangle}$, $A_{I2}(e, I) = e(m)I(e(m))$,
- $A_{Pr} : A_{\langle inst\rangle} \to A_{\langle prog\rangle}$, $A_{Pr}(I) = I(0)$.

Semantica algebrei initiale

- $\mathcal{S}_A : T_\Sigma \to \mathcal{A}$ unicul morfism
- $\mathcal{S}_G : T_\Sigma \to L_G$ unicul morfism
- $w \in L(G)$, $t_w \in T_\Sigma$, $\mathcal{S}_G(t_w) = w$ (t este unic)

semantica lui w este $Sem(w) = \mathcal{S}_A(t_w)$

Exemplu:

$w = ON\ 5 + M\ E\ OFF$
$t_w = T_{\langle Pr\rangle}(T_{\langle I1\rangle}(T_{\langle r3\rangle}(T_{\langle r1\rangle}(T_{\langle p1\rangle}(T_{\langle c5\rangle})), T_{\langle r2\rangle})))$
$Sem(w) = \mathcal{S}_A(t) = A_{\langle Pr\rangle}(A_{\langle I1\rangle}(A_{\langle r3\rangle}(A_{\langle r1\rangle}(A_{\langle p1\rangle}(A_{\langle c5\rangle})), A_{\langle r2\rangle})))$

Semantica algebrei initiale

$w = ON\ 5 + M\ E\ OFF$
$Sem(w) = A_{\langle Pr\rangle}(A_{\langle I1\rangle}(A_{\langle r3\rangle}(A_{\langle r1\rangle}(A_{\langle p1\rangle}(A_{\langle c5\rangle})), A_{\langle r2\rangle}))) \overset{A_{Pr}(I)=I(0)}{=}$
$A_{\langle I1\rangle}(A_{\langle r3\rangle}(A_{\langle r1\rangle}(A_{\langle p1\rangle}(A_{\langle c5\rangle})), A_{\langle r2\rangle}))(0) =$
$A_{\langle r3\rangle}(A_{\langle r1\rangle}(A_{\langle p1\rangle}(A_{\langle c5\rangle})), A_{\langle r2\rangle})(0) =$
$A_{\langle r1\rangle}(A_{\langle p1\rangle}(A_{\langle c5\rangle}))(0) + A_{\langle r2\rangle}(0) =$
$A_{\langle p1\rangle}(A_{\langle c5\rangle}) + 0 =$
$A_{\langle c5\rangle} + 0 =$
$5 + 0 = 5$

Exemple

Vom prezenta următoarele aplicații ale metodei algebrei inițiale:

- reprezentarea expresiilor în formă poloneză inversă,
- compilării unei expresii aritmetice folosind o mașina cu stivă și acumulator.

$G = (S_0, N, T, P)$ pt. expresii

Definim o gramatica pentru expresii construite cu variabilele $X = \{x, y, z\}$.

- $G = (S_0, N, T, P)$
  $N = \{\langle prog\rangle, \langle exp\rangle, \langle term\rangle, \langle fact\rangle, \langle var\rangle\}$,
  $S_0 = \langle prog\rangle$,
  $T = \{x, y, z, (,), +, *\}$, $P = \{p0, \ldots, p9\}$,

$G = (S_0, N, T, P)$ pentru expresii aritmetice

[p0] $\langle prog \rangle \to \langle exp \rangle$
[p1] $\langle var \rangle \to x$
[p2] $\langle var \rangle \to y$
[p3] $\langle var \rangle \to z$
[p4] $\langle fact \rangle \to \langle var \rangle$
[p5] $\langle fact \rangle \to (\langle exp \rangle)$
[p6] $\langle term \rangle \to \langle fact \rangle$
[p7] $\langle term \rangle \to \langle fact \rangle * \langle term \rangle$
[p8] $\langle exp \rangle \to \langle term \rangle$
[p9] $\langle exp \rangle \to \langle exp \rangle + \langle term \rangle$

Semantica algebrei inițiale

$$\begin{tikzcd}
 & T_\Sigma \arrow[r, "\mathcal{S}_G"] \arrow[dl, dotted, "\mathcal{P}"'] \arrow[d, "\mathcal{S}_A"] \arrow[dr, dotted, "\mathcal{C}"] & L_G \\
Pol & \mathcal{A} & Cod
\end{tikzcd}$$

$$\begin{tikzcd}
t_w \arrow[rr, "\mathcal{S}_G"] \arrow[d, "\mathcal{S}_A"] & & w \arrow[ll, dashed] \\
Sem(w) = \mathcal{S}_A(t_w) & &
\end{tikzcd}$$

Algebra Pol - suportul

- Definim forma poloneză postfix a unei expresii din $L(G)$ prin metoda algebrei inițiale. Pentru aceasta construim $\mathcal{G}$-algebra semantică Pol, unde $\mathcal{G} = (S = N, \Sigma = P)$.
- $Pol_{\langle var \rangle} = X = \{x, y, z\}$

  $Pol_s = T^* = \bigcup_{k \geq 0} T^k$, $s \in N \setminus \{\langle var \rangle\}$

Algebra Pol - operațiile

- $Pol_p : T^* \to T^*$, $p \in \{p0, p5, p6, p8\}$,
  $Pol_p(w) = w$ oricare $w \in T^*$,
- Pol_{p1}, Pol_{p2}, $Pol_{p3} :\to X$,
  $Pol_{p1} = x$, $Pol_{p2} = y$, $Pol_{p3} = z$,
- $Pol_{p4} : X \to T^*$, $Pol_{p4}(v) = v$, oricare $v \in X$,
- $Pol_{p7} : T^* \times T^* \to T^*$, $Pol_{p7}(\alpha, \beta) = \alpha\beta*$,
- $Pol_{p9} : T^* \times T^* \to T^*$, $Pol_{p7}(\alpha, \beta) = \alpha\beta+$,

Forma poloneză

- $\mathcal{S}_G : T_\Sigma \to L_G$ unicul $\mathcal{G}$-morfism
  $\mathcal{P} : T_\Sigma \to Pol$ unicul $\mathcal{G}$-morfism
- Pentru orice $w \in L(G)$, forma poloneză postfix este $\mathcal{P}(t_w)$, unde $t_w \in T_\Sigma$ este unicul termen cu $\mathcal{S}_G(t_w) = w$.
- $w = y * (x + z)$
  $t_w = p0(p8(p7(p4(p2), p5(p9(p8(p6(p4(p1))), p6(p4(p3)))))))$

  $\mathcal{P}(t_e) = \mathcal{P}(T_{p0}(T_{p8}(T_{p7}(T_{p4}(T_{p2}), T_{p5}(T_{p9}(T_{p8}($
  $T_{p6}(T_{p4}(T_{p1}))), T_{p6}(T_{p4}(T_{p3})))))))) =$
  $Pol_{p0}(Pol_{p8}(Pol_{p7}(Pol_{p4}(Pol_{p2}), Pol_{p5}(Pol_{p9}(Pol_{p8}($
  $Pol_{p6}(Pol_{p4}(Pol_{p1}))), Pol_{p6}(Pol_{p4}(Pol_{p3}))))))) =$
  $Pol_{p7}(y, Pol_{p9}(x, z)) = Pol_{p7}(y, xz+) = yxz + *$

Compilarea unei expresii

- Folosim pentru compilare o mașină cu stivă și acumulator (MSA)
- Definim limbajul MSA și algebra semantică $Cod = (\{Cod_s\}_{s \in S}, \{C_{pi}\}_{pi \in \Sigma})$.
- Compilarea expresiei $w \in L(G)$ în limbajul MSA este $\mathcal{C}(t_w)$, unde $t_w \in T_\Sigma$ este unicul termen cu $\mathcal{S}_G(t_w) = w$, iar $\mathcal{C} : T_\Sigma \to Cod$ este unicul $\mathcal{G}$-morfism.

Funcționarea SMA

- La efectuarea operației $op(e1, \ldots, e_n)$, valorile $e_1, \ldots, e_{n-1}$ vor fi în stivă, iar valoarea lui e_n în registru.
- Rezultatul evaluării oricărei expresii este depus în registru. Registrul este vârful *real* al stivei. Evaluarea unei expresii lasă stiva neschimbată.

e_n	$op(e1, \ldots, e_n)$
e_{n-1}	$\ldots$
$\ldots$	$\ldots$
e_1	$\ldots$

Instrucțiunile SMA

- R este registrul (acumulatorul)
- P este adresa primei poziții libere din stivă
- Instrucțiunile *SMA*:
  inc P (incrementează P)
  dec P (decrementează P)
  Ad R (adună valoarea din vârful stivei cu cea din R, iar rezultatul este depus în R)
  Mu R (înmulțește valoarea din vârful stivei cu cea din R, iar rezultatul este depus în R)
  ld *v* (încarcă în R valoarea din *v*)
  st R (mută în stivă valoarea din R)
  print (afișează conținutul lui R)

Algebra Cod - suportul

- $Inst(MSA)$ = instructiunile definite anterior
- codul unei expresii este un șir de instrucțiuni separate prin ";"
- $Cod_s = (Inst(SMA) \cup \{;\})^*$, oricare $s \in S = N$
- Notăm $SI := Cod_s$, $s \in S$

Algebra Cod - operațiile

- $C_{p0} : SI \to SI$,
  $C_{p0}(\alpha) = \alpha\,;$ print
- $C_p : SI \to SI$, $p \in \{p4, p5, p6, p8\}$
  $C_p(\alpha) = \alpha$
- C_{p1}, C_{p2}, $C_{p3} :\to SI$,
  $C_{p1} =$ld x, $C_{p2} =$ld y, $C_{p3} =$ld z
- $C_{p7} : SI \times SI \to SI$,
  $C_{p7}(\alpha, \beta) = \alpha\,;$ st R ; inc P ; $\beta\,;$ Mu R ; dec P
- $C_{p9} : SI \times SI \to SI$,
  $C_{p9}(\alpha, \beta) = \alpha\,;$ st R ; inc P ; $\beta\,;$ Ad R ; dec P

Compilarea

- Pentru orice $w \in L(G)$, codul evaluării expresiei w este $\mathcal{C}(t_w)$, unde $t_w \in T_\Sigma$ este unicul termen cu $\mathcal{C}_G(t_w) = w$.
- $e = y * (x + z)$
  $t_e = p0(p8(p7(p4(p2), p5(p9(p8(p6(p4(p1))), p6(p4(p3)))))))$

  $$\mathcal{C}(t) = \mathcal{C}(T_{p0}(T_{p8}(T_{p7}(T_{p4}(T_{p2}), T_{p5}(T_{p9}(T_{p8}(T_{p6}(T_{p4}(T_{p1}))), T_{p6}(T_{p4}(T_{p3})))))))) =$$
  $$C_{p0}(C_{p8}(C_{p7}(C_{p4}(C_{p2}), C_{p5}(C_{p9}(C_{p8}(C_{p6}(C_{p4}(C_{p1})))), C_{p6}(C_{p4}(C_{p3})))))) =$$

Compilarea

$C_{p0}(C_{p7}(C_{p2}, C_{p9}(C_{p1}, C_{p3}))) =$

$C_{p7}(C_{p2}, C_{p9}(C_{p1}, C_{p3}))\,;$ print=

$C_{p2}\,;$ st R ; inc P ; $C_{p9}(C_{p1}, C_{p3})\,;$
Mu R ; dec P ; print=

$C_{p2}\,;$ st R ; inc P ; $C_{p1}\,;$ st R ; inc P ; $C_{p3}\,;$
Ad R ; dec P ; Mu R ; dec P ; print=

ldy ; st R ; inc P ; ld x ; st R ; inc P ; ld z ;
Ad R ; dec P ; Mu R ; dec P ; print

$e = y * (x + z)$, $x = 3$, $y = 7$, $z = 2$

Cod	R	Stiva(←)	P
-	-	-	1
ld y;	7	-	1
st R; inc P;	7	7	2
ld x;	3	7	2
st R; inc P;	3	3 7	3
ld z;	2	3 7	3
Ad R; dec P;	5	7	2
Mu R; dec P;	35	-	1
print	35		

Clauze Horn. Rezolutie

Signatura multisortata de ordinul I

O signatura multisortata de ordinul I este un triplet (S, Σ, Π) unde:

- (S, Σ) este o signatura multisortata,
- $\Pi = \{\Pi_w\}_{w \in S^+}$ familie de simboluri de predicate.

Exemplu:

$NATPRED = (S, \Sigma, \Pi)$
$S := \{nat\}$, $\Sigma := \{0 :\to nat, succ : nat \to nat\}$,
$\Pi_{nat} := \{pos\}$, $\Pi_{nat\ nat} := \{<\}$

Modele si morfisme

Un (S, Σ, Π)-model este $A = (A_S, A_\Sigma, A_\Pi)$ unde:

- (A_S, A_Σ) este o (S, Σ)-algebra,
- $A_\Pi = \{A_\pi \subseteq A^w \mid \pi \in \Pi_w\}$.

Notatie: $A^w := A_{s_1} \times \cdots \times A_{s_n}$ daca $w = s_1 \cdots s_n$.

Fie A si B doua (S, Σ, Π)-modele. O functie $f : A \to B$ este (S, Σ, Π)-morfism daca:

- f este morfism de (S, Σ)-algebre,
- $(a_1, \ldots, a_n) \in A_\pi \Rightarrow (f_{s_1}(a_1), \ldots, f_{s_n}(a_n)) \in B_\pi$ oricare $\pi \in \Pi_{s_1 \cdots s_n}$

Exemplu

- *NATPRED*-modelul A:
  $A_{nat} := \mathbb{N}$, $A_0 := 0$, $A_{succ}(n) := n + 1$
  $A_{pos} := \{n | n > 0\}$, $A_< := \{(k, n) | k \leq n\}$
- *NATPRED*-modelul B:
  $B_{nat} := \{2^n | n \in \mathbb{N}\}$, $B_0 := 1$, $B_{succ}(2^n) := 2^{n+1}$
  $B_{pos} := \{2^n | n > 0\}$, $B_< := \{(2^k, 2^n) | k \leq n\}$
- $f : A \to B$, $f(n) := 2^n$ or. $n \in \mathbb{N}$
  morfism de *NATPRED*-modele

Universul Herbrand

(S, Σ, Π) signatura multisortata de ordinul I

Universul Herbrand este T_Σ, multimea termenilor fara variabile.

- Definim (S, Σ, Π)-modelul $T_{\Sigma,\Pi} := (T_\Sigma, T_\Pi)$, unde
  T_Σ este (S, Σ)-algebra termenilor fara variabile,
  $T_\pi := \emptyset$ oricare $\pi \in \Pi$.
- $T_{\Sigma,\Pi}$ este (S, Σ, Π)-model initial.

Clauze Horn

Fie (Σ, Π) limbaj de ordinul I .
Definim formulele de ordinul I peste (Σ, Π).

- formula atomica: $\pi(t_1, \cdots, t_n)$ cu $t_i \in T_\Sigma(X)$ or. i
- literal: $\pi(t_1, \cdots, t_n)$ sau $\neg\pi(t_1, \cdots, t_n)$
  (literal pozitiv) (literal negativ)
- clauză: $L_1 \vee \cdots \vee L_m$ unde L_i literali or. i
- clauza vida: $\square$ (disjunctie indexata de $\emptyset$)

Satisfiabilitate

Fie A un (Σ, Π)- model, X o multime de variabile si
$t_1, \ldots, t_n \in T_\Sigma(X)$.

- $A \models \pi(t_1, \ldots, t_n) \Leftrightarrow (h(t_1), \cdots, h(t_n)) \in A_\pi$
  oricare $h : T_\Sigma(X) \to A$ morfism
- $A \models \neg\pi(t_1, \ldots, t_n) \Leftrightarrow A \not\models \pi(t_1, \ldots, t_n)$
- $A \models L_1 \vee \cdots \vee L_m \Leftrightarrow A \models L_1$ sau $\cdots$ sau $A \models L_m$
- Daca Γ este o multime de clauze, atunci
  $A \models \Gamma \Leftrightarrow A \models \gamma$ or. $\gamma \in \Gamma$
  (spunem ca A este model pentru Γ)
- O multime de clauze Γ se numeste satisfiabilă daca are un model. Clauza vidă $\square$ nu este satisfiabilă.

Consecinta semantica si satisfiabilitatea

Γ multime de clauze,
$P_1, \cdots, P_n$ formule atomice cu variabile din X

Teorema. Sunt echivalente

- $\Gamma \models \exists X(P_1 \wedge \cdots \wedge P_n)$,
- $\Gamma \cup \{\neg(\exists X(P_1 \wedge \cdots \wedge P_n))\}$ nu e satisfiabila,
- $\Gamma \cup \{(\forall X(\neg P_1 \vee \cdots \vee \neg P_n))\}$ nu e satisfiabila.

Rezolutia este o metoda prin care verificam daca o multime de clauze nu este satisfiabila.

Limbajul PROLOG are la bază rezoluția SLD pentru clauze Horn.

Clauze Horn

O clauză Horn are cel mult un literal pozitiv

Clauzele Horn au trei forme:

rule (clauză definită)	$\neg P_1 \vee \cdots \neg P_n \vee P$
fact (clauza definită)	P
query (clauza scop)	$\neg P_1 \vee \cdots \neg P_n$

unde $P_1, \ldots, P_n, P$ sunt formule atomice.

Notatia Prolog este:

- $P : -P_1, \cdots, P_n$
- P
- $: -P_1, \cdots, P_n$

In aceasta notatie toate variabilele sunt cuantificate universal.

Rezolutia SLD

Γ multime de clauze definite

SLD $\boxed{\dfrac{\neg P_1 \vee \cdots \vee \neg P_i \vee \cdots \neg P_n}{(\neg P_1 \vee \cdots \vee \neg Q_1 \vee \cdots \vee \neg Q_m \vee \cdots \neg P_n)\theta}}$

unde $Q \vee \neg Q_1 \vee \cdots \vee \neg Q_m$ este o clauza definita din Γ
(in care toate variabilele au fost redenumite) si
θ este c.g.u pentru P_i si Q.

Notatia Prolog pentru rezolutia SLD:
G_i este $: -P_1, \cdots, P_i, \cdots, P_n$
C_i este $Q : -Q_1, \cdots, Q_m$
$\theta_i(Q) = \theta_i(P_i)$
G_{i+1} este $: -(P_1, \cdots, Q_1, \cdots, Q_m, \cdots, P_n)\theta_i$

unde $P\theta := \theta(P)$, $\theta(p(t)) := p(\theta(t))$ cu $p \in \Pi$

Rezolutia SLD

Γ multime de clauze definite, G clauza scop

O derivare din Γ prin rezolutie SLD este o secventa
$G_0 := G, G_1, \ldots, G_k, \ldots$
in care G_{i+1} se obtine din G_i prin regula SLD.
Dacă exista un k cu $G_k = \Box$ atunci derivarea se numeste
SLD-respingere.

Completitdinea SLD-rezolutiei

Sunt echivalente:

- exista o SLD-respingere a lui G din Γ,
- $\Gamma \models \exists X(P_1 \wedge \cdots \wedge P_n)$,
  unde G este $: -P_1, \cdots, P_n$.

Exemplu

Fie Γ urmatoarea multime de clauze Horn (bază de date):

(1) $stramos(X, Y) : -parinte(X, Y)$
(2) $stramos(X, Y) : -parinte(X, Z), stramos(Z, Y)$
(3) $parinte(dan, bogdan)$
(4) $parinte(bogdan, ana)$

Gasiti o respingere din Γ pentru

$: -stramos(Y, bogdan), stramos(bogdan, Z)$

(există Y și Z astfel încât
Y este *stramos* al lui *bogdan* și
bogdan este *stramos* al lui Z)

Exemplu

Vom nota $p := parinte$, $q := stramos$ (predicate binare)
$b := bogdan$, $d := dan$, $a := ana$ (constante)
$G_0 : q(Y, b), q(b, Z)$

---

(1) $q(X', Y') : -p(X', Y')$, $\theta_1 := \{X' \leftarrow Y, Y' \leftarrow b\}$
$G_1 : p(Y, b), q(b, Z)$

---

(3) $p(d, b)$, $\theta_2 := \{ Y \leftarrow d \}$
$G_2 : q(b, Z)$

---

(1) $q(X'', Y'') : -p(X'', Y'')$, $\theta_3 := \{X'' \leftarrow b, Y'' \leftarrow Z\}$
$G_3 : p(b, Z)$

---

(4) $p(b, a)$, $\theta_4 := \{ Z \leftarrow a \}$
$G_4 : \square$

Signaturi multisortate de ordinul I

(S, Σ, Π) signatura multisortata de ordinul I,

O clauza Horn are forma $(\forall X)p \; if \{p_1, \ldots, p_n\}$
unde $p_1, \ldots, p_n, p$ formule atomice
(p_i este $\pi(t_1, \ldots, t_{m_i})$ cu $\pi \in \Pi$,
$t_1, \ldots, t_{m_i} \in T_\Sigma(X)$)

Daca $n = 0$ clauza are forma $(\forall X)p$

$(\forall X)p \; if \{p_1, \ldots, p_n\}$ este o notatie pt. $p : -p_1, \ldots, p_n$

O *clauza scop* este o multime G de formule atomice.

Fie Γ multime de clauze Horn. Regula rezolutiei este

$$\boxed{\frac{G \cup \{p(t)\}}{G\theta \cup C\theta}}$$

G este o multime de formule atomice,
$(\forall X)p(s) \; if \; C \in \Gamma$ (varianta noua)
$p \in \Pi$, $\theta(s) = \theta(t)$, θ c.g.u. pentru s si t

Clauze Horn ecuationale

Unei signaturi multisortate de ordinul I (S, Σ, Π) ii asociem o signatura multisortata $(S^b, \Sigma^b \cup \Pi^b)$ astfel:

- $S^b := S \cup \{b\}$, unde $b \notin S$,
- $\Sigma^b := \Sigma \cup \{true :\to b\}$,
- $\Pi^b_{w,b} := \Pi_w$
  (predicatele devin operatii cu rezultat boolean)

Exemplu

$NATPRED = (S, \Sigma, \Pi)$, $S := \{nat\}$,
$\Sigma := \{0 :\to nat, succ : nat \to nat\}$,
$\Pi_{nat} := \{pos\}$, $\Pi_{nat\ nat} := \{<\}$
$NATPRED^b = (S^b, \Sigma^b \cup \Pi^b)$, $S^b := \{nat, b\}$,
$\Sigma^b := \{0 :\to nat, succ : nat \to nat, true :\to b\}$,
$\Pi_{nat,b} := \{pos : \; nat \to b\}$, $\Pi_{nat\ nat,b} := \{<: \; nat \; nat \to b\}$

Clauze Horn ecuationale

(S, Σ, Π) signatura multisortata de ordinul I,
$(S^b, \Sigma^b \cup \Pi^b)$ signatura multisortată asociată

- Unei clauza Horn $(\forall X)p \; if \{p_1, \ldots, p_n\}$ îi asociem ecuatia conditionată $(\forall X)p \doteq_b true \; if \{p_1 \doteq_b true, \ldots, p_n \doteq_b true\}$.
- Unui (S, Σ, Π)-model $\overline{A}$ ii asociem o $(S^b, \Sigma^b \cup \Pi^b)$-algebră A

$$(a_1, \ldots, a_m) \in \overline{A}_\pi \Leftrightarrow A_\pi(a_1, \ldots, a_m) = A_{true}$$

- Sunt echivalente:
  - $\overline{A} \models_{\Sigma,\Pi} (\forall X)p \; if \{p_1, \ldots, p_n\}$
  - $A \models_{\Sigma^b,\Pi^b} (\forall X)p \doteq_b true \; if \{p_1 \doteq_b true, \ldots, p_n \doteq_b true\}$,