Curs 4

Din cursul trecut

- Sistem de deducție *backchain*:
  - Toate clauzele din T sunt axiome.
  - Regula de deducție *backchain*:

$$\frac{\theta(p_1) \quad \theta(p_2) \quad \ldots \quad \theta(p_n) \quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q)}{\theta(q')}$$

unde $p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q \in T$, iar θ este cgu pentru q' și q.

- Sistem de deducție corect și complet:

$$T \vdash_B Q \quad \text{ddacă} \quad T \models Q$$

Cuprins

Rezoluție SLD

Regula *backchain* și rezoluția SLD

- Regula *backchain* este implementată în programarea logică prin rezoluția SLD (Selected, Linear, Definite).
- Prolog are la bază rezoluția SLD.

Clauze definite

- Singurele formule admise sunt de forma:
  - formule atomice: $P(t_1, \ldots, t_n)$
  - $P_1 \land \ldots \land P_n \to Q$, unde toate P_i, Q sunt formule atomice.
- O clauză definită $P_1 \land \ldots \land P_n \to Q$ poate fi gândită ca formula

$$Q \lor \neg P_1 \lor \ldots \lor \neg P_n$$

Rezoluția SLD

Fie T o mulțime de clauze definite.

$$\text{SLD} \quad \boxed{\frac{\neg P_1 \vee \cdots \vee \neg P_i \vee \cdots \vee \neg P_n}{(\neg P_1 \vee \cdots \vee \neg Q_1 \vee \cdots \vee \neg Q_m \vee \cdots \vee \neg P_n)\theta}}$$

unde

- $Q \vee \neg Q_1 \vee \cdots \vee \neg Q_m$ este o clauză definită din T (în care toate variabilele au fost redenumite) și
- θ este c.g.u pentru P_i și Q.

Rezoluția SLD

Exemplu

```
father(eddard,sansa).
father(eddard,jonSnow).

stark(eddard).
stark(catelyn).

stark(X) :- father(Y,X),
stark(Y).
```

```
?- stark(jonSnow)
```

$$\text{SLD} \quad \boxed{\frac{\neg P_1 \vee \cdots \vee \neg P_i \vee \cdots \vee \neg P_n}{(\neg P_1 \vee \cdots \vee \neg Q_1 \vee \cdots \vee \neg Q_m \vee \cdots \vee \neg P_n)\theta}}$$

- $Q \vee \neg Q_1 \vee \cdots \vee \neg Q_m$ este o clauză definită din T
- θ este c.g.u pentru P_i și Q.

Rezoluția SLD

Exemplu

$father(eddard, sansa)$
$father(eddard, jonSnow)$

$stark(eddard)$
$stark(catelyn)$

$stark(X) \vee \neg father(Y, X) \vee \neg stark(Y)$

$$\frac{\neg stark(jonSnow)}{}$$

$$\theta(X) = jonSnow$$

$$\text{SLD} \quad \boxed{\frac{\neg P_1 \vee \cdots \vee \neg P_i \vee \cdots \vee \neg P_n}{(\neg P_1 \vee \cdots \vee \neg Q_1 \vee \cdots \vee \neg Q_m \vee \cdots \vee \neg P_n)\theta}}$$

- $Q \vee \neg Q_1 \vee \cdots \vee \neg Q_m$ este o clauză definită din T
- θ este c.g.u pentru P_i și Q.

Rezoluția SLD

Exemplu

$father(eddard, sansa)$
$father(eddard, jonSnow)$

$stark(eddard)$
$stark(catelyn)$

$stark(X) \lor \neg father(Y, X) \lor \neg stark(Y)$

$$\frac{\neg stark(jonSnow)}{\neg father(Y, jonSnow) \lor \neg stark(Y)}$$

$$\theta(X) = jonSnow$$

$$\text{SLD} \quad \boxed{\frac{\neg P_1 \lor \cdots \lor \neg P_i \lor \cdots \lor \neg P_n}{(\neg P_1 \lor \cdots \lor \neg Q_1 \lor \cdots \lor \neg Q_m \lor \cdots \lor \neg P_n)\theta}}$$

- $Q \lor \neg Q_1 \lor \cdots \lor \neg Q_m$ este o clauză definită din T
- θ este c.g.u pentru P_i și Q.

Rezoluția SLD

Fie T o mulțime de clauze definite și $P_1 \wedge \ldots \wedge P_m$ o întrebare, unde P_i sunt formule atomice.

- O derivare din T prin rezoluție SLD este o secvență

  $$G_0 := \neg P_1 \vee \ldots \vee \neg P_m, \quad G_1, \quad \ldots, \quad G_k, \ldots$$

  în care G_{i+1} se obține din G_i prin regula SLD.
- Dacă există un k cu $G_k = \square$ (clauza vidă), atunci derivarea se numește SLD-respingere.

Rezoluția SLD

Teoremă (Completitudinea SLD-rezoluției)

Sunt echivalente:

- *există o SLD-respingere a lui $P_1 \wedge \ldots \wedge P_m$ din T,*
- $T \vdash_b P_1 \wedge \ldots \wedge P_m$,
- $T \models P_1 \wedge \cdots \wedge P_m$.

Demonstrație

Rezultă din completitudinea sistemului de deducție backchain și din faptul că:

există o SLD-respingere a lui $P_1 \wedge \ldots \wedge P_m$ din T

ddacă

$$T \vdash_b P_1 \wedge \ldots \wedge P_m$$

$\square$

Rezoluția SLD - arbori de căutare

Arbori SLD

- Presupunem că avem o mulțime de clauze definite T și o țintă $G_0 = \neg P_1 \vee \ldots \vee \neg P_m$

- Construim un arbore de căutare (arbore SLD) astfel:
  - Fiecare nod al arborelui este o țintă (posibil vidă)
  - Rădăcina este G_0
  - Dacă arborele are un nod G_i, iar G_{i+1} se obține din G_i folosind regula SLD folosind o clauză $C_i \in T$, atunci nodul G_i are copilul G_{i+1}. Muchia dintre G_i și G_{i+1} este etichetată cu C_i.

- Dacă un arbore SLD cu rădăcina G_0 are o frunză $\square$ (clauza vidă), atunci există o SLD-respingere a lui G_0 din T.

Rezoluția SLD

Exemplu

- Fie T următoarea mulțime de clauze definite:
  1. $grandfather(X, Z) :- father(X, Y), parent(Y, Z)$
  2. $parent(X, Y) :- father(X, Y)$
  3. $parent(X, Y) :- mother(X, Y)$
  4. $father(ken, diana)$
  5. $mother(diana, brian)$

- Găsiți o respingere din T pentru

$$:- grandfather(ken, Y)$$

Rezoluția SLD

Exemplu

- Fie T următoarea mulțime de clauze definite:
  1. $grandfather(X, Z) \vee \neg father(X, Y) \vee \neg parent(Y, Z)$
  2. $parent(X, Y) \vee \neg father(X, Y)$
  3. $parent(X, Y) \vee \neg mother(X, Y)$
  4. $father(ken, diana)$
  5. $mother(diana, brian)$

- Găsiți o respingere din T pentru

$$\neg grandfather(ken, Y)$$

Rezoluția SLD

Exemplu

- Fie T următoarea mulțime de clauze definite:
  1. $grandfather(X, Z), \neg father(X, Y), \neg parent(Y, Z)$
  2. $parent(X, Y), \neg father(X, Y)$
  3. $parent(X, Y), \neg mother(X, Y)$
  4. $father(ken, diana)$
  5. $mother(diana, brian)$

- Găsiți o respingere din T pentru

$$\neg grandfather(ken, Y)$$

Rezoluția SLD

Exemplu

1. $grandfather(X, Z), \neg father(X, Y), \neg parent(Y, Z)$
2. $parent(X, Y), \neg father(X, Y)$
3. $parent(X, Y), \neg mother(X, Y)$
4. $father(ken, diana)$
5. $mother(diana, brian)$

$$\neg grandfather(ken, Y)$$

$$\downarrow 1$$

$$\neg father(ken, V), \neg parent(V, Y)$$

$$\downarrow 4$$

$$\neg parent(diana, Y)$$

$$\swarrow 2 \qquad \searrow 3$$

$$\neg father(diana, Y) \qquad \neg mother(diana, Y)$$

$$\downarrow 5$$

$$\square$$

Rezoluția SLD

Exemplu

1. $grandfather(X, Z), \neg father(X, Y), \neg parent(Y, Z)$
2. $parent(X, Y), \neg father(X, Y)$
3. $parent(X, Y), \neg mother(X, Y)$
4. $father(ken, diana)$
5. $mother(diana, brian)$

$$\neg grandfather(ken, Y)$$

$$\downarrow 1$$

$$\neg father(ken, V), \ \neg parent(V, Y)$$

$$\downarrow 4$$

$$\neg parent(diana, Y)$$

$$\swarrow 2 \qquad \searrow 3$$

$$\neg father(diana, Y) \qquad \neg mother(diana, Y)$$

$$\downarrow 5$$

$$\square$$

Ce lipsește?

Limbajul Prolog

- Am arătat că sistemul de inferență din spatele Prolog-ului este complet.
  - Dacă o întrebare este consecință logică a unei mulțimi de clauze, atunci există o derivare a întrebării.
- Totuși, strategia de căutate din Prolog este incompletă!
  - Chiar dacă o întrebare este consecință logică a unei mulțimi de clauze, Prolog nu găsește mereu o derivare a întrebării.

Limbajul Prolog

Exemplu

```
warmerClimate :- albedoDecrease.
warmerClimate :- carbonIncrease.
iceMelts :- warmerClimate.
albedoDecrease :- iceMelts.
carbonIncrease.

?- iceMelts.
! Out of local stack
```

Exemplu (cont.)

Există o derivare a lui *iceMelts* în sistemul de deducție din clauzele:

$$\begin{array}{rcl} albedoDecrease & \to & warmerClimate \\ carbonIncrease & \to & warmerClimate \\ warmerClimate & \to & iceMelts \\ iceMelts & \to & albedoDecrease \\ & & carbonIncrease \end{array}$$

$$\dfrac{\dfrac{carbonInc. \qquad carbonInc. \to warmerClim.}{warmerClim.} \qquad warmerClim. \to iceMelts}{iceMelts}$$

Spre logica ecuațională multisortată

In cursurile trecute

Programare logica – cazul logicii de ordinul I

Logica clauzelor definite/Logica Horn

- Un fragment al logicii de ordinul intai
- Singurele formule admise sunt clauze definite:
  - formule atomice: $P(t_1, \ldots, t_n)$
  - $A_1 \land \ldots \land A_n \to B$, unde toate A_i, B sunt formule atomice.

Problema programarii logice: $T \models A_1 \land \ldots \land A_n$

- T multime de clauze definite
- toate A_i sunt formule atomice

Diverse probleme

Ce se intamplă dacă vrem să avem egalitate?

Cum putem deosebi obiecte de naturi diferite?

Vom investiga
logica ecuațională multisortată.

Algebre multisortate

Signaturi multisortate

Definiție

O signatură multisortată este o pereche (S, Σ), unde

Signaturi multisortate

Definiție

O signatură multisortată este o pereche (S, Σ), unde

- $S \neq \emptyset$ este o mulțime de sorturi.

Signaturi multisortate

Definiție

O signatură multisortată este o pereche (S, Σ), unde

- $S \neq \emptyset$ este o mulțime de sorturi.
- Σ este o mulțime de simboluri de operații de forma

$$\sigma : s_1 s_2 \ldots s_n \to s.$$

Signaturi multisortate

Definiție

O signatură multisortată este o pereche (S, Σ), unde

- $S \neq \emptyset$ este o mulțime de sorturi.
- Σ este o mulțime de simboluri de operații de forma

$$\sigma : s_1 s_2 \ldots s_n \to s.$$

  - σ este numele operației
  - $s_1, \ldots, s_n, s \in S$
  - $\langle s_1 s_2 \ldots s_n, s \rangle$ este aritatea operației
  - $s_1, \ldots, s_n$ sunt sorturile argumentelor
  - s este sortul rezultatului
  - dacă $n = 0$, atunci $\sigma :\to s$ este simbolul unei constante

Observații

- S^* mulțimea cuvintelor finite cu elemente din S.

Observații

- S^* mulțimea cuvintelor finite cu elemente din S.
- Notații alternative pentru o signatură (S, Σ):

Observații

- S^* mulțimea cuvintelor finite cu elemente din S.
- Notații alternative pentru o signatură (S, Σ):
  - $(S, \{\Sigma_{s_1 s_2 \ldots s_n, s}\}_{s_1 s_2 \ldots s_n \in S^*, s \in S})$

Observații

- S^* mulțimea cuvintelor finite cu elemente din S.
- Notații alternative pentru o signatură (S, Σ):
  - $(S, \{\Sigma_{s_1 s_2 \ldots s_n, s}\}_{s_1 s_2 \ldots s_n \in S^*, s \in S})$
  - $(S, \{\Sigma_{w,s}\}_{w \in S^*, s \in S})$
    - $\sigma \in \Sigma_{w,s}$ ddacă $\sigma : w \to s$
    - $w = s_1 \ldots s_n \in S^*$

Observații

- S^* mulțimea cuvintelor finite cu elemente din S.
- Notații alternative pentru o signatură (S, Σ):
  - $(S, \{\Sigma_{s_1 s_2 \ldots s_n, s}\}_{s_1 s_2 \ldots s_n \in S^*, s \in S})$
  - $(S, \{\Sigma_{w,s}\}_{w \in S^*, s \in S})$
    - $\sigma \in \Sigma_{w,s}$ ddacă $\sigma : w \to s$
    - $w = s_1 \ldots s_n \in S^*$
  - Σ

Observații

- S^* mulțimea cuvintelor finite cu elemente din S.
- Notații alternative pentru o signatură (S, Σ):
  - $(S, \{\Sigma_{s_1 s_2 \ldots s_n, s}\}_{s_1 s_2 \ldots s_n \in S^*, s \in S})$
  - $(S, \{\Sigma_{w,s}\}_{w \in S^*, s \in S})$
    - $\sigma \in \Sigma_{w,s}$ ddacă $\sigma : w \to s$
    - $w = s_1 \ldots s_n \in S^*$
  - Σ
- Este permisă supraîncărcarea operațiilor:
  - σ este supraîncărcată dacă $\sigma \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$ și $\langle w_1, s_1 \rangle \neq \langle w_2, s_2 \rangle$

Exemple

Exemplu

$NAT = (S, \Sigma)$

- $S = \{nat\}$
- $\Sigma = \{0 : \to nat,\ succ : nat \to nat\}$

Exemple

Exemplu

$NAT = (S, \Sigma)$

- $S = \{nat\}$
- $\Sigma = \{0 : \to nat, \ succ : nat \to nat\}$

Cum arată $\Sigma = \{\Sigma_{w,s}\}_{w \in S^*, s \in S}$ în acest caz?

Exemple

Exemplu

$NAT = (S, \Sigma)$

- $S = \{nat\}$
- $\Sigma = \{0 : \to nat, \; succ : nat \to nat\}$

Cum arată $\Sigma = \{\Sigma_{w,s}\}_{w \in S^*, s \in S}$ în acest caz?

- $\Sigma_{\lambda,nat} = \{0\}$ și $\Sigma_{nat,nat} = \{succ\}$
- $\Sigma_{w,s} = \emptyset$, pentru orice alt $w \in S^*$ și $s \in S$

Exemple

Exemplu

$NAT = (S, \Sigma)$

- $S = \{nat\}$
- $\Sigma = \{0 : \to nat,\ succ : nat \to nat\}$

Cum arată $\Sigma = \{\Sigma_{w,s}\}_{w \in S^*, s \in S}$ în acest caz?

- $\Sigma_{\lambda,nat} = \{0\}$ și $\Sigma_{nat,nat} = \{succ\}$
- $\Sigma_{w,s} = \emptyset$, pentru orice alt $w \in S^*$ și $s \in S$

Modul în Maude:

```
fmod MYNAT-SIMPLE is
   sort Nat .
   op 0 : -> Nat .
   op succ : Nat -> Nat .
endfm
```

Exemple

Exemplu

$BOOL = (S, \Sigma)$

- $S = \{bool\}$
- $\Sigma = \{T : \to bool,\ F : \to bool,$
  $\neg : bool \to bool,$
  $\vee : bool\ bool \to bool,\ \wedge : bool\ bool \to bool\}$

Cum arată $\Sigma = \{\Sigma_{w,s}\}_{w \in S^*, s \in S}$ în acest caz?

- $\Sigma_{\lambda, bool} = \{T, F\}$
- $\Sigma_{bool, bool} = \{\neg\}$
- $\Sigma_{bool\ bool, bool} = \{\vee, \wedge\}$
- $\Sigma_{w,s} = \emptyset$, pentru orice alt $w \in S^*$ și $s \in S$

Exemple

Exemplu

$NATBOOL = (S, \Sigma)$

- $S = \{bool, nat\}$
- $\Sigma = \{T : \to bool,\ \ F : \to bool,$
  $0 : \to nat, succ : nat \to nat,$
  $\leq : nat\ nat \to bool\}$

Cum arată $\Sigma = \{\Sigma_{w,s}\}_{w \in S^*, s \in S}$ în acest caz?

- $\Sigma_{\lambda,bool} = \{T, F\}$
- $\Sigma_{\lambda,nat} = \{0\}$
- $\Sigma_{nat,nat} = \{succ\}$
- $\Sigma_{nat\ nat,bool} = \{\leq\}$
- $\Sigma_{w,s} = \emptyset$, pentru orice alt $w \in S^*$ și $s \in S$

Exemple

Exemplu

$STIVA = (S, \Sigma)$

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

Signaturi ordonat-sortate

Definiție

O signatură ordonat-sortată este un triplet $(S, \leq, \Sigma)$ unde

- (S, Σ) signatură multisortată
- $(S, \leq)$ mulțime parțial ordonată
- condiția de monotonie:

$$\text{dacă } \sigma \in \Sigma_{w_1, s_1} \cap \Sigma_{w_2, s_2} \text{ atunci } w_1 \leq w_2 \Rightarrow s_1 \leq s_2$$

Exemple

Exemplu

$NAT = (S, \leq, \Sigma)$

- $S = \{zero, nznat, nat\}$
- $zero \leq nat,\ nznat \leq nat$
- $\Sigma = \{0 : \to zero,\ \ succ : nat \to nznat\}$

Modul în Maude:

```
fmod MYNAT is
   sorts Zero NzNat Nat .
   subsort Zero NzNat < Nat .
   op 0 : -> Zero .
   op succ : Nat -> NzNat .
endfm
```

Exemple

Exemplu

$STIVA = (S, \leq, \Sigma)$

- $S = \{elem, stiva, nvstiva\}$
- $elem \leq stiva,\ nvstiva \leq stiva$
- $\Sigma = \{0 : \to elem,\ \ empty : \to stiva,$
  $push : elem\ stiva \to nvstiva,$
  $pop : nvstiva \to stiva,$
  $top : nvstiva \to elem\}$

În practică se folosesc signaturi ordonat-sortate.

Mulțimi multisortate

Fixăm o mulțime de sorturi S.

Definiție

O mulțime S-sortată este o familie de mulțimi $A = \{A_s\}_{s \in S}$ indexată după S.

- Pentru orice sort $s \in S$, avem o mulțime A_s.
- Pentru orice sort $s \in S$ și $a \in A_s$, spunem că a este de sort s.

Exemplu

Exemplu

- Mulțime de sorturi: $S = \{nat, bool\}$
- Mulțime S-sortată: $A = \{A_{nat}, A_{bool}\}$, unde
  - $A_{nat} = \mathbb{N}$
  - $A_{bool} = \{true, false\}$
- 1 este de sort *nat*
- *false* este de sort *bool*

- sorturi $=$ tipuri
- elemente de sort s $=$ date de tip s

Operații

Operațiile uzuale pe mulțimi sunt extinse la cazul multisortat pe componente:

- $\{A_s\}_{s\in S} \subseteq \{B_s\}_{s\in S}$ ddacă $A_s \subseteq B_s$, or. $s \in S$
- $\{A_s\}_{s\in S} \cup \{B_s\}_{s\in S} = \{A_s \cup B_s\}_{s\in S}$
- $\{A_s\}_{s\in S} \cap \{B_s\}_{s\in S} = \{A_s \cap B_s\}_{s\in S}$
- $\{A_s\}_{s\in S} \times \{B_s\}_{s\in S} = \{A_s \times B_s\}_{s\in S}$

Funcții

Fixăm $A = \{A_s\}_{s\in S}$, $B = \{B_s\}_{s\in S}$, $C = \{C_s\}_{s\in S}$ mulțimi S-sortate.

Definiție

O funcție S-sortată $f : A \to B$ este o familie de funcții $f = \{f_s\}_{s\in S}$, unde $f_s : A_s \to B_s$, pt. or. $s \in S$.

- Dacă $f : A \to B$ și $g : B \to C$, definim compunerea
$$f;g = \{(f;g)_s\}_{s\in S} : A \to C$$
$$(f;g)_s(a) = (f_s;g_s)(a) = g_s(f_s(a)), \text{ or. } a \in A.$$
- Compunerea este asociativă: $(f;g);h = f;(g;h)$.
- Funcția identitate: $1_A : A \to A$, $1_A = \{1_{A_s}\}_{s\in S}$.
- Observați că $f;1_B = f$, $1_A;f = f$, or. $f : A \to B$.

Funcții

- O funcție S-sortată $f = \{f_s\}_{s \in S} : A \to B$ se numește injectivă, (surjectivă, bijectivă) dacă f_s este injectivă, (surjectivă, bijectivă), or. $s \in S$.
- O funcție S-sortată $f = \{f_s\}_{s \in S} : A \to B$ se numește inversabilă dacă există $g : B \to A$ astfel încât $f; g = 1_A$ si $g; f = 1_B$.

Funcții

- O funcție S-sortată $f = \{f_s\}_{s \in S} : A \to B$ se numește injectivă, (surjectivă, bijectivă) dacă f_s este injectivă, (surjectivă, bijectivă), or. $s \in S$.
- O funcție S-sortată $f = \{f_s\}_{s \in S} : A \to B$ se numește inversabilă dacă există $g : B \to A$ astfel încât $f ; g = 1_A$ si $g ; f = 1_B$.

Propoziție

O funcție S-sortată $f : A \to B$ este inversabilă $\Leftrightarrow$ este bijectivă.

Demonstrație

Exercițiu!

Algebre multisortate

Fixăm o signatură multisortată (S, Σ).

Definiție

O algebră multisortată de tip (S, Σ) este o structură $\mathcal{A} = (A_S, A_\Sigma)$ unde

Algebre multisortate

Fixăm o signatură multisortată (S, Σ).

Definiție

O algebră multisortată de tip (S, Σ) este o structură $\mathcal{A} = (A_S, A_\Sigma)$ unde

- $A_S = \{A_s\}_{s \in S}$ este o mulțime S-sortată (mulțimea suport).

Algebre multisortate

Fixăm o signatură multisortată (S, Σ).

Definiție

O algebră multisortată de tip (S, Σ) este o structură $\mathcal{A} = (A_S, A_\Sigma)$ unde

- $A_S = \{A_s\}_{s \in S}$ este o mulțime S-sortată (mulțimea suport).
- $A_\Sigma = \{A_\sigma\}_{\sigma \in \Sigma}$ este o familie de operații astfel încât
  - dacă $\sigma : s_1 \ldots s_n \to s$ în Σ, atunci $A_\sigma : A_{s_1} \times \ldots \times A_{s_n} \to A_s$ (operație).
  - dacă $\sigma : \to s$ în Σ, atunci $A_\sigma \in A_s$ (constantă).

Algebre multisortate

Fixăm o signatură multisortată (S, Σ).

Definiție

O algebră multisortată de tip (S, Σ) este o structură $\mathcal{A} = (A_S, A_\Sigma)$ unde

- $A_S = \{A_s\}_{s \in S}$ este o mulțime S-sortată (mulțimea suport).
- $A_\Sigma = \{A_\sigma\}_{\sigma \in \Sigma}$ este o familie de operații astfel încât
  - dacă $\sigma : s_1 \ldots s_n \to s$ în Σ, atunci $A_\sigma : A_{s_1} \times \ldots \times A_{s_n} \to A_s$ (operație).
  - dacă $\sigma : \to s$ în Σ, atunci $A_\sigma \in A_s$ (constantă).

- Exprimări alternative:
  - $\mathcal{A} = (A_S, A_\Sigma)$ este o (S, Σ)-algebră sau o Σ-algebră
  - $\mathcal{A}$ este o Σ-algebră
  - $\mathcal{A}$ este o algebră
- Notație: $A_{s_1 \ldots s_n} = A_{s_1} \times \ldots \times A_{s_n}$

Exemple

Exemplu

$NAT = (S, \Sigma)$

- $S = \{nat\}$
- $\Sigma = \{0 : \to nat,\ succ : nat \to nat\}$

Exemple

Exemplu

$NAT = (S, \Sigma)$

- $S = \{nat\}$
- $\Sigma = \{0 : \to nat,\ succ : nat \to nat\}$

NAT-algebra $\mathcal{A}$

- Mulțimea suport: $A_{nat} := \mathbb{N}$
- Operații: $A_0 := 0$, $A_{succ}(x) := x + 1$

Exemple

Exemplu

$NAT = (S, \Sigma)$

- $S = \{nat\}$
- $\Sigma = \{0 : \to nat,\ succ : nat \to nat\}$

NAT-algebra $\mathcal{A}$

- Mulțimea suport: $A_{nat} := \mathbb{N}$
- Operații: $A_0 := 0$, $A_{succ}(x) := x + 1$

NAT-algebra $\mathcal{B}$

- Mulțimea suport: $B_{nat} := \{0, 1\}$
- Operații: $B_0 := 0$, $B_{succ}(x) := 1 - x$

Exemple

Exemplu

$NAT = (S, \Sigma)$

- $S = \{nat\}$
- $\Sigma = \{0 : \to nat,\ succ : nat \to nat\}$

NAT-algebra $\mathcal{A}$

- Mulțimea suport: $A_{nat} := \mathbb{N}$
- Operații: $A_0 := 0$, $A_{succ}(x) := x + 1$

NAT-algebra $\mathcal{B}$

- Mulțimea suport: $B_{nat} := \{0, 1\}$
- Operații: $B_0 := 0$, $B_{succ}(x) := 1 - x$

NAT-algebra $\mathcal{C}$

- Mulțimea suport: $C_{nat} := \{2^n \mid n \in \mathbb{N}\}$
- Operații: $C_0 := 1$, $C_{succ}(2^n) := 2^{n+1}$

Exemple

Exemplu

$BOOL = (S, \Sigma)$

- $S = \{bool\}$
- $\Sigma = \{T : \to bool,\ F : \to bool, \neg : bool \to bool,$
  $\vee : bool\ bool \to bool,\ \wedge : bool\ bool \to bool\}$

Exemple

Exemplu

$BOOL = (S, \Sigma)$

- $S = \{bool\}$
- $\Sigma = \{T : \to bool,\ F : \to bool, \neg : bool \to bool,$
  $\vee : bool\ bool \to bool,\ \wedge : bool\ bool \to bool\}$

$BOOL$-algebra $\mathcal{A}$

- Mulțimea suport: $A_{bool} := \{0, 1\}$
- Operații:
  - $A_T := 1,$
  - $A_F := 0,$
  - $A_{\neg}(x) := 1 - x$
  - $A_{\vee}(x, y) := \max(x, y),$
  - $A_{\wedge}(x, y) := \min(x, y)$

Exemple

Exemplu

$BOOL = (S, \Sigma)$

- $S = \{bool\}$
- $\Sigma = \{T : \to bool,\ F : \to bool, \neg : bool \to bool,$
  $\vee : bool\ bool \to bool,\ \wedge : bool\ bool \to bool\}$

Exemple

Exemplu

$BOOL = (S, \Sigma)$

- $S = \{bool\}$
- $\Sigma = \{T : \to bool,\ F : \to bool, \neg : bool \to bool,$
  $\vee : bool\ bool \to bool,\ \wedge : bool\ bool \to bool\}$

BOOL-algebra $\mathcal{B}$

- Mulțimea suport: $B_{bool} := \mathcal{P}(\mathbb{N})$
- Operații:
  - $B_T := \mathbb{N}$,
  - $B_F := \emptyset$,
  - $B_\neg(X) := \mathbb{N} \setminus X$
  - $B_\vee(X, Y) := X \cup Y$,
  - $B_\wedge(x, y) := X \cap Y$

Exemple

Exemplu

$STIVA = (S, \Sigma)$

- $S = \{elem, stiva\}$
- $\Sigma = \{0 : \rightarrow elem,\ empty : \rightarrow stiva, push : elem\ stiva \rightarrow stiva,$
  $pop : stiva \rightarrow stiva, top : stiva \rightarrow elem\}$

Exemple

Exemplu

$STIVA = (S, \Sigma)$

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

$STIVA$-algebra $\mathcal{A}$

- Mulțimea suport:
  - $A_{elem} := \mathbb{N}$,
  - $A_{stiva} := \mathbb{N}^*$

Exemple

Exemplu

$STIVA = (S, \Sigma)$

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

$STIVA$-algebra $\mathcal{A}$

- Mulțimea suport:
  - $A_{elem} := \mathbb{N}$,
  - $A_{stiva} := \mathbb{N}^*$
- Operații:
  - $A_0 := 0$,
  - $A_{empty} := \lambda$,
  - $A_{push}(n, n_1 \ldots n_k) := nn_1 \ldots n_k$,
  - $A_{pop}(\lambda) := \lambda$, $A_{pop}(n) := \lambda$, $A_{pop}(n_1 n_2 \ldots n_k) := n_2 \ldots n_k$, pt $k \geq 2$
  - $A_{top}(\lambda) := 0$, $A_{top}(n_1 \ldots n_k) := n_1$, pt. $k \geq 1$

Exemple

Exemplu

$STIVA = (S, \Sigma)$

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

$STIVA$-algebra $\mathcal{B}$

- Mulțimea suport:
  - $B_{elem} := \{0\}$,
  - $B_{stiva} := \mathbb{N}$

Exemple

Exemplu

$STIVA = (S, \Sigma)$

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

$STIVA$-algebra $\mathcal{B}$

- Mulțimea suport:
  - $B_{elem} := \{0\}$,
  - $B_{stiva} := \mathbb{N}$
- Operații:
  - $B_0 := 0$,
  - $B_{empty} := 0$,
  - $B_{push}(0, n) := n + 1$,
  - $B_{pop}(0) := 0,\ B_{pop}(n) := n - 1$, pt. $n \geq 1$,
  - $B_{top}(n) := 0$

Privire de ansamblu

signatură (multisortată)	Σ	simboluri
Σ-algebră	$\mathcal{A}$	"înțeles" pentru simboluri

Algebre ordonat-sortate

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

Definiție

O algebră ordonat-sortată de tip $(S, \leq, \Sigma)$ este o (S, Σ)-algebră $\mathcal{A} = (A_S, A_\Sigma)$ astfel încât

- dacă $s_1 \leq s_2$, atunci $A_{s_1} \subseteq A_{s_2}$.
- dacă $\sigma \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$ si $w_1 \leq w_2$, atunci
$$A_\sigma^{w_2,s_2}(x) = A_\sigma^{w_1,s_1}(x),$$
oricare $x \in A_{w_1}$.

Semantica unui modul în **Maude** este o algebră ordonat-sortată.

Pe săptămâna viitoare!