



Programare logică

Tipuri abstracte de date

Algebra de termeni

Algebra inițială

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 să fie modele ale același tip de date. Aceasta asigură **independența de implementare**.

ADT

(S, Σ) semnătură multisortată

$Alg(S, \Sigma)$ clasa tuturor (S, Σ) -algebrelor

- Un **tip abstract de date** este o clasă \mathcal{C} de (S, Σ) -algebre închisă la izomorfism:

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

ADT

(S, Σ) semnatură multisortată

$Alg(S, \Sigma)$ clasa tuturor (S, Σ) -algebrelor

- Un tip abstract de date este o clasă \mathcal{C} de (S, Σ) -algebre închisă la izomorfism:

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

- Un tip abstract de date \mathcal{C} se numește **monomorfic** dacă

$$A \simeq B \text{ oricare ar fi } A, B \in \mathcal{C}.$$

În caz contrar, \mathcal{C} se numește **polimorfic**.

ADT

- Orice tip abstract de date monomorfic este de forma $[A]_{\mathcal{K}} = \{B \in \mathcal{K} \mid A \simeq B\}$, unde \mathcal{K} este o clasă de (S, Σ) -algebre și $A \in \mathcal{K}$ fixată.

ADT

- Orice tip abstract de date monomorfic este de forma $[A]_{\mathcal{K}} = \{B \in \mathcal{K} \mid A \simeq B\}$, unde \mathcal{K} este o clasă de (S, Σ) -algebre și $A \in \mathcal{K}$ fixată.
- Orice tip abstract de date \mathcal{C} se decompune în tipuri monomorfe $\mathcal{C} = \bigcup \{[A]_{\mathcal{C}} \mid A \in \mathcal{C}\}$.

ADT

- Orice tip abstract de date monomorfic este de forma $[A]_{\mathcal{K}} = \{B \in \mathcal{K} \mid A \simeq B\}$,
unde \mathcal{K} este o clasă de (S, Σ) -algebre și $A \in \mathcal{K}$ fixată.
- Orice tip abstract de date \mathcal{C} se decompune în tipuri monomorfe $\mathcal{C} = \bigcup \{[A]_{\mathcal{C}} \mid A \in \mathcal{C}\}$.
- $Alg(S, \Sigma)$ este un tip abstract de date polimorfic.

ADT

- Orice tip abstract de date monomorfic este de forma $[A]_{\mathcal{K}} = \{B \in \mathcal{K} \mid A \simeq B\}$, unde \mathcal{K} este o clasă de (S, Σ) -algebre și $A \in \mathcal{K}$ fixată.
- Orice tip abstract de date \mathcal{C} se decompune în tipuri monomorfe $\mathcal{C} = \bigcup \{[A]_{\mathcal{C}} \mid A \in \mathcal{C}\}$.
- $Alg(S, \Sigma)$ este un tip abstract de date polimorfic.
- În CafeObj modulele `mod*` specifică un tip abstract de date polimorfic.

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* $f : I \rightarrow B$.

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* $f : I \rightarrow B$.
- Dacă I inițială în \mathcal{K} , $A \in \mathcal{K}$ și $A \simeq I$ atunci A inițială în \mathcal{K} .

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* $f : I \rightarrow B$.
- Dacă I inițială în \mathcal{K} , $A \in \mathcal{K}$ și $A \simeq I$ atunci A inițială în \mathcal{K} .
- Dacă A_1 și A_2 sunt inițiale în \mathcal{K} atunci $A_1 \simeq A_2$.

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* $f : I \rightarrow B$.
- Dacă I inițială în \mathcal{K} , $A \in \mathcal{K}$ și $A \simeq I$ atunci A inițială în \mathcal{K} .
- Dacă A_1 și A_2 sunt inițiale în \mathcal{K} atunci $A_1 \simeq A_2$.
- $\mathcal{I}_{\mathcal{K}} = \{I \mid I \text{ inițială în } \mathcal{K}\}$
este un tip abstract de date monomorfic. În CafeObj
modulele `mod!` specifică un astfel de tip monomorfic.

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* $f : I \rightarrow B$.
- Dacă I inițială în \mathcal{K} , $A \in \mathcal{K}$ și $A \simeq I$ atunci A inițială în \mathcal{K} .
- Dacă A_1 și A_2 sunt inițiale în \mathcal{K} atunci $A_1 \simeq A_2$.
- $\mathcal{I}_{\mathcal{K}} = \{I \mid I \text{ inițială în } \mathcal{K}\}$
este un tip abstract de date monomorfic. În CafeObj modulele `mod!` specifică un astfel de tip monomorfic.
- Vom construi o algebră inițială în $\mathcal{K} = \text{Alg}(S, \Sigma)$, și anume și anume *algebra termenilor fără variabile*.

Mulțime de variabile

(S, Σ) semnătură multisortată

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

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

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



(S, Σ) semnătură, 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

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

care verifică următoarele proprietăți:

(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

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

care verifică următoarele proprietăți:

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

(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

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

care verifică următoarele proprietăți:

■(T1) $X_s \subseteq T_\Sigma(X)_s$

■(T2) dc. $\sigma \rightarrow s$, at. $\sigma \in T_\Sigma(X)_s$,

■(T3) dc. $\sigma : s_1 \cdots s_n \rightarrow s$ și $t_i \in T_\Sigma(X)_{s_i}$ or. $i = 1, \dots, n$
at. $\sigma(t_1, \dots, t_n) \in T_\Sigma(X)_s$.

• $Var(t) :=$ mulțimea variabilelor care apar în $t \in T_\Sigma$ p.12

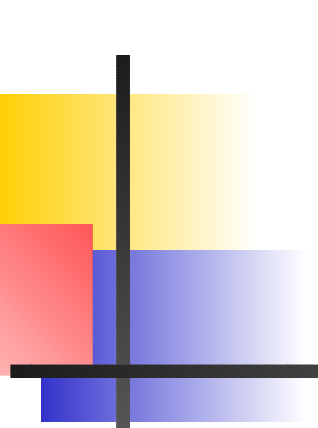
(S, Σ) signatură multisortată

Mulțimea S -sortată **termenilor fără variabile**, T_Σ , este cea *mai mică* mulțime de *șiruri finite* peste alfabetul

$$L = \bigcup_{w,s} \Sigma_{w,s} \cup \{ (,) \} \cup \{ , \}$$

care verifică următoarele proprietăți:

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



■ $STIVA = (S, \Sigma)$

$S = \{elem, stiva\}, X_{elem} = \{x, y\}, X_{stiva} = \emptyset,$

$\Sigma = \{0 : \rightarrow elem, empty : \rightarrow stiva,$

$push : elem\ stiva \rightarrow stiva,$

$pop : stiva \rightarrow stiva,$

$top : stiva \rightarrow elem\}$

Example

■ $STIVA = (S, \Sigma)$

$$S = \{elem, stiva\}, X_{elem} = \{x, y\}, X_{stiva} = \emptyset,$$

$$\Sigma = \{0 : \rightarrow elem, empty : \rightarrow stiva,$$

$$push : elem\ stiva \rightarrow stiva,$$

$$pop : stiva \rightarrow stiva,$$

$$top : stiva \rightarrow elem\}$$

■ $T_{\Sigma}(X)_{elem} = \{0, x, y, top(pop(empty)),$
 $top(push(x, empty)), \dots\}$

$$T_{\Sigma}(X)_{stiva} = \{empty, push(y, empty), pop(empty),$$
$$push(top(empty), empty), \dots\}$$

Exemple

■ $STIVA = (S, \Sigma)$

$$S = \{elem, stiva\}, X_{elem} = \{x, y\}, X_{stiva} = \emptyset,$$

$$\Sigma = \{0 : \rightarrow elem, empty : \rightarrow stiva,$$

$$push : elem\ stiva \rightarrow stiva,$$

$$pop : stiva \rightarrow stiva,$$

$$top : stiva \rightarrow elem\}$$

■ $T_{\Sigma}(X)_{elem} = \{0, x, y, top(pop(empty)),$
 $top(push(x, empty)), \dots\}$

$$T_{\Sigma}(X)_{stiva} = \{empty, push(y, empty), pop(empty),$$
$$push(top(empty), empty), \dots\}$$

■ şiruri care nu sunt termeni

$$pop(0), (pop)top(empty), empty(y), \dots$$

Exemple

■ $NATBOOL = (S, \Sigma)$

$$S = \{bool, nat\}$$

$$\Sigma = \{T : \rightarrow bool, F : \rightarrow bool, 0 : \rightarrow nat,$$

$$s : nat \rightarrow nat,$$

$$\leq : nat \ nat \rightarrow bool\}$$

Exemple

- $NATBOOL = (S, \Sigma)$

$$S = \{bool, nat\}$$

$$\Sigma = \{T : \rightarrow bool, F : \rightarrow bool, 0 : \rightarrow nat,$$

$$s : nat \rightarrow nat,$$

$$\leq : nat \ nat \rightarrow bool\}$$

- $T_{NATBOOL, nat} = \{0, s(0), s(s(0)), \dots\}$

$$T_{NATBOOL, bool} = \{T, F, \leq (0, 0), \leq (0, s(0)), \dots\}$$

Exemple

- $NATBOOL = (S, \Sigma)$

$$S = \{bool, nat\}$$

$$\Sigma = \{T : \rightarrow bool, F : \rightarrow bool, 0 : \rightarrow nat,$$

$$s : nat \rightarrow nat,$$

$$\leq : nat \ nat \rightarrow bool\}$$

- $T_{NATBOOL, nat} = \{0, s(0), s(s(0)), \dots\}$

$$T_{NATBOOL, bool} = \{T, F, \leq (0, 0), \leq (0, s(0)), \dots\}$$

- şiruri care nu sunt termeni

$$\leq (T, F), s \leq (0), Ts(0), \dots$$

Exemple

- $AUTOMAT = (S, \Sigma)$, $S = \{intrare, stare, iesire\}$,
 $\Sigma = \{s0 : \rightarrow stare,$
 $f : intrare\ stare \rightarrow stare,$
 $g : stare \rightarrow iesire\}$
 $T_{AUTOMAT, stare} = \{s0\}$, $T_{AUTOMAT, intrare} = \emptyset$,
 $T_{AUTOMAT, iesire} = \{g(s0)\}$

Exemple

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

Exemple

- $AUTOMAT = (S, \Sigma)$, $S = \{intrare, stare, iesire\}$,
 $\Sigma = \{s0 : \rightarrow stare,$
 $f : intrare\ stare \rightarrow stare,$
 $g : stare \rightarrow iesire\}$
 $T_{AUTOMAT, stare} = \{s0\}$, $T_{AUTOMAT, intrare} = \emptyset$,
 $T_{AUTOMAT, iesire} = \{g(s0)\}$
- $GRAF = (S, \Sigma)$, $S = \{arc, nod\}$
 $\Sigma = \{v0 : arc \rightarrow nod, v1 : arc \rightarrow nod\}$
 $T_{GRAF, arc} = T_{GRAF, nod} = \emptyset$
- Atunci când semnatura definește un modul mod^* ,
mulțimea termenilor fără variabile poate fi neinteresantă.



Expresii aritmetice

- $NATEXP = (S = \{nat\}, \Sigma)$

Expresii aritmetice

- $NATEXP = (S = \{nat\}, \Sigma)$
- $\Sigma = \{0 : \rightarrow nat, s : nat \rightarrow nat,$
 $\quad + : nat \ nat \rightarrow nat, * : nat \ nat \rightarrow nat\}$

Expresii aritmetice

- $NATEXP = (S = \{nat\}, \Sigma)$
- $\Sigma = \{0 : \rightarrow nat, s : nat \rightarrow nat, \\ + : nat \ nat \rightarrow nat, * : nat \ nat \rightarrow nat\}$
- $T_{NATEXP} = \{0, s(0), s(s(0)), \dots \\ +(0, 0), *(0, +(s(0), 0)), *(s(0), s(s(0))), \dots\}$

Expresii aritmetice

- $NATEXP = (S = \{nat\}, \Sigma)$
- $\Sigma = \{0 : \rightarrow nat, s : nat \rightarrow nat, \\ + : nat \ nat \rightarrow nat, * : nat \ nat \rightarrow nat\}$
- $T_{NATEXP} = \{0, s(0), s(s(0)), \dots \\ +(0, 0), *(0, +(s(0), 0)), *(s(0), s(s(0))), \dots\}$
- şiruri care nu sunt termeni
 $+(0), 0(s)s(0), *(0), \dots$

Expresii aritmetice

- $NATEXP = (S = \{nat\}, \Sigma)$
- $\Sigma = \{0 : \rightarrow nat, s : nat \rightarrow nat, \\ + : nat \ nat \rightarrow nat, * : nat \ nat \rightarrow nat\}$
- $T_{NATEXP} = \{0, s(0), s(s(0)), \dots \\ +(0, 0), *(0, +(s(0), 0)), *(s(0), s(s(0))), \dots\}$
- şiruri care nu sunt termeni
 $+(0), 0(s)s(0), *(0), \dots$
- T_{NATEXP} este mulţimea expresiilor aritmetice peste \mathbb{N} .

Inducția pe termeni

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

■ pasul inițial:

$$P(x) = \text{true} \text{ or. } x \in X, P(\sigma) = \text{true} \text{ or. } \sigma \rightarrow s,$$

Inducția pe termeni

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

■ pasul inițial:

$P(x) = \text{true}$ or. $x \in X$, $P(\sigma) = \text{true}$ or. $\sigma \rightarrow s$,

■ pasul de inducție:

dacă $t_1 \in T_\Sigma(X)_{s_1}, \dots, t_n \in T_\Sigma(X)_{s_n}$ și

$P(t_1) = \dots = P(t_n) = \text{true}$ atunci

$P(\sigma(t_1, \dots, t_n)) = \text{true}$ or. $\sigma : s_1 \dots s_n \rightarrow s$.

Inducția pe termeni

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

■ pasul inițial:

$P(x) = \text{true}$ or. $x \in X$, $P(\sigma) = \text{true}$ or. $\sigma \rightarrow s$,

■ pasul de inducție:

dacă $t_1 \in T_\Sigma(X)_{s_1}, \dots, t_n \in T_\Sigma(X)_{s_n}$ și

$P(t_1) = \dots = P(t_n) = \text{true}$ atunci

$P(\sigma(t_1, \dots, t_n)) = \text{true}$ or. $\sigma : s_1 \dots s_n \rightarrow s$.

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

Algebra de termeni

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

$T_\Sigma(X)$ algebra termenilor cu variabile din X

T_Σ algebra termenilor fără variabile ($X = \emptyset$)

Inițialitate

(S, Σ) semnătură multisortată

Teoremă.

Algebra T_Σ este inițială în $Alg(S, \Sigma)$.

■ or. (S, Σ) -algebra B , ex. un unic morfism $f : T_\Sigma \rightarrow B$.

■ $\mathcal{I}_\Sigma := \{A \mid A \in Alg(S, \Sigma), A \simeq T_\Sigma\} = [T_\Sigma]$
este un tip abstract de date monomorfic.

Acesta reprezintă semantica unui modul `mod`! în `CafeObj` care conține numai declarații de sorturi și operații.