Programare logică

Signaturi, Algebre, Morfisme

Mulţimi S-sortate

$S \neq \emptyset$

- O mulţime S- sortată A este o familie $A = \{A_s\}_{s \in S}$.

Mulţimi S-sortate

$S \neq \emptyset$

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

Mulţimi S-sortate

$S \neq \emptyset$

- O mulţime S- sortată A este o familie $A = \{A_s\}_{s \in S}$.
- Dacă $A = \{A_s\}_{s \in S}$ şi $B = \{B_s\}_{s \in S}$ 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

Funcţii S-sortate

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

- O funcţie S- sortată $f : A \to B$ este o familie de funcţii $f = \{f_s\}_{s\in S}$, unde $f_s : A_s \to B_s$ oricare $s \in S$.

Funcţii S-sortate

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

- O funcţie S- sortată $f : A \to B$ este o familie de funcţii $f = \{f_s\}_{s\in S}$, unde $f_s : A_s \to B_s$ oricare $s \in S$.
- Dacă $f : A \to B$ şi $g : B \to C$, definim $f;g : A \to C$, $f;g = \{f_s;g_s\}_{s\in S}$

Funcţii S-sortate

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

- O funcţie S- sortată $f : A \to B$ este o familie de funcţii $f = \{f_s\}_{s \in S}$, unde $f_s : A_s \to B_s$ oricare $s \in S$.
- Dacă $f : A \to B$ şi $g : B \to C$, 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. $s \in S$, $a \in A_s$

Funcţii S-sortate

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

- O funcţie S- sortată $f : A \to B$ este o familie de funcţii $f = \{f_s\}_{s\in S}$, unde $f_s : A_s \to B_s$ oricare $s \in S$.
- Dacă $f : A \to B$ şi $g : B \to C$, 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. $s \in S$, $a \in A_s$
- $1_A : A \to A$, $1_A = \{1_{A_s}\}_{s\in S}$

Funcţii S-sortate

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

- O funcţie S- sortată $f : A \to B$ este o familie de funcţii $f = \{f_s\}_{s\in S}$, unde $f_s : A_s \to B_s$ oricare $s \in S$.
- Dacă $f : A \to B$ şi $g : B \to C$, 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)

Funcţii S-sortate

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

- O funcţie S- sortată $f : A \to B$ este o familie de funcţii $f = \{f_s\}_{s\in S}$, unde $f_s : A_s \to B_s$ oricare $s \in S$.
- Dacă $f : A \to B$ şi $g : B \to C$, 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. $f : A \to B$

Funcţii S-sortate

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

- O funcţie S-sortată $f : A \to B$ se numeşte injectivă, (surjectivă, bijectivă) dacă f_s este injectivă, (surjectivă, bijectivă) oricare $s \in S$.

Funcţii S-sortate

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

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

Funcţii S-sortate

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

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

Propoziţie. O funcţie S-sortată $f : A \to B$ este inversabilă dacă şi numai dacă este bijectivă (f_s este bijectivă oricare $s \in S$).

Signaturi multisortate

(S, Σ) signatură multisortată

- S mulţimea sorturilor

Signaturi multisortate

(S, Σ) signatură multisortată

- S mulţimea sorturilor
- Σ mulţimea simbolurilor de operaţii $\sigma : s_1 \cdots s_n \to s$

Signaturi multisortate

(S, Σ) signatură multisortată

- S mulţimea sorturilor
- Σ mulţimea simbolurilor de operaţii $\sigma : s_1 \cdots s_n \to s$
  - σ este simbolul (numele) operaţiei

Signaturi multisortate

(S, Σ) signatură multisortată

- S mulţimea sorturilor
- Σ mulţimea simbolurilor de operaţii $\sigma : s_1 \cdots s_n \to s$
  - σ este simbolul (numele) operaţiei
  - $s_1, \cdots, s_n, s \in S$

    $s_1, \cdots, s_n$ sorturile argumentelor

    s sortul rezultatului

    $< s_1 \cdots s_n, s >$ aritatea operaţiei

Signaturi multisortate

(S, Σ) signatură multisortată

- S mulţimea sorturilor
- Σ mulţimea simbolurilor de operaţii $\sigma : s_1 \cdots s_n \to s$
  - σ este simbolul (numele) operaţiei
  - $s_1, \cdots, s_n$,$s \in S$
    $s_1, \cdots, s_n$ sorturile argumentelor
    s sortul rezultatului
    $< s_1 \cdots s_n, s >$ aritatea operaţiei
  - dacă $n = 0$ atunci $\sigma :\to s$ este simbolul unei operaţii *constante*

Signaturi

(S, Σ) signatură multisortată

- $S^* := \bigcup_{n \geq 0} S^n$
  $S^0 := \{\lambda\}$, $S^n := \{s_1 \cdots s_n | s_i \in S$ or. $i\}$

Signaturi

(S, Σ) signatură multisortată

- $S^* := \bigcup_{n \geq 0} S^n$
  $S^0 := \{\lambda\}$, $S^n := \{s_1 \cdots s_n | s_i \in S$ or. $i\}$
- $\Sigma = (\Sigma_{w,s})_{w \in S^*, s \in S}$
  $\sigma \in \Sigma_{w,s} \Leftrightarrow \sigma : w \to s$
  $w = s_1 \cdots s_n \in S^*$

Signaturi

(S, Σ) signatură multisortată

- $S^* := \bigcup_{n \geq 0} S^n$
  $S^0 := \{\lambda\}$, $S^n := \{s_1 \cdots s_n | s_i \in S$ 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
  $\sigma \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$ şi $< w_1, s_1 > \neq < w_2, s_2 >$

Signaturi

(S, Σ) signatură multisortată

- $S^* := \bigcup_{n \geq 0} S^n$

  $S^0 := \{\lambda\}$, $S^n := \{s_1 \cdots s_n | s_i \in S$ 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

  $\sigma \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$ şi $< w_1, s_1 > \neq < w_2, s_2 >$

- este permisă supraîncărcarea operaţiilor

Exemple

- $BOOL = (S, \Sigma)$
  - $S = \{bool\}$
  - $\Sigma = \{T :\rightarrow bool, F :\rightarrow bool,$
    $\neg : bool \rightarrow bool,$
    $\vee : bool\ bool \rightarrow bool,$
    $\wedge : bool\ bool \rightarrow bool\}$

Exemple

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

Exemple

- $NATBOOL = (S, \Sigma)$
  - $S = \{bool, nat\}$
  - $\Sigma = \{T :\rightarrow bool, F :\rightarrow bool, 0 :\rightarrow nat,$
    $succ : nat \rightarrow nat,$
    $\leq : nat\ nat \rightarrow bool\}$

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

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

Exemple

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

Exemple

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

    $f : intrare\ stare \rightarrow stare,$

    $g : stare \rightarrow iesire\}$
- $GRAF = (S, \Sigma)$
  - $S = \{arc, nod\}$
  - $\Sigma = \{v0 : arc \rightarrow nod, \quad v1 : arc \rightarrow nod\}$

Signaturi ordonat-sortate

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

- (S, Σ) signatură multisortată
- $(S, \leq)$ mulţime parţial ordonată
- condiţia 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:

$S = \{elem, stiva, nvstiva\}$, $elem \leq stiva$, $nvstiva \leq stiva$

$\Sigma = \{empty :\to stiva,\ \ push : elem\ stiva \to nvstiva,$
$\quad pop : nvstiva \to stiva, top : nvstiva \to elem\}$.

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

- $A_S = \{A_s\}_{s \in S}$ (mulţimea suport)
- $A_\Sigma = \{A_\sigma\}_{\sigma \in \Sigma}$ (familie de operaţii) 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

- $BOOL = (S = \{bool\}, \Sigma)$

  $$\Sigma = \{T :\rightarrow bool, F :\rightarrow bool, \neg : bool \rightarrow bool, \\ \vee : bool\ bool \rightarrow bool, \wedge : bool\ bool \rightarrow bool\}$$

Exemple

- $BOOL = (S = \{bool\}, \Sigma)$

  $\Sigma = \{T :\to bool, F :\to bool, \neg : bool \to bool,$

  $\qquad \vee : bool\ bool \to bool, \wedge : bool\ bool \to bool\}$

- $BOOL$-algebra A:

  $A_{bool} := \{0, 1\}$

  $A_T := 1$, $A_F := 0$, $A_\neg(x) := 1 - x$,

  $A_\vee(x, y) := max(x, y)$, $A_\wedge(x, y) := min(x, y)$

Exemple

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

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

Exemple

- $NAT = (S = \{nat\}, \Sigma)$
  $\Sigma = \{0 :\to nat, succ : nat \to nat\}$
- NAT-algebra A:
  $A_{nat} := \mathbb{N}$
  $A_0 := 0$, $A_{succ}(x) := x + 1$

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

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

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

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. $k \geq 1$.

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: $A_{elem} := \mathbb{N}$, $A_{stiva} := \mathbb{N}^*$
  $A_0 := 0$, $A_{empty} := \lambda$, $A_{push}(n, n_1 \cdots n_k) := n\, n_1 \cdots n_k$,
  $A_{pop}(\lambda) = A_{pop}(n) := \lambda$,
  $A_{pop}(n_1 n_2 \cdots n_k) := n_2 \cdots n_k$ 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

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

Exemple

- $AUTOMAT = (S = \{intrare, stare, iesire\}, \Sigma)$
  $\Sigma = \{s0 :\to stare, f : intrare\ stare \to stare,$
  $g : stare \to iesire\}$
- $AUTOMAT$-algebra A:
  $A_{intrare} = \{x, y\}$, $A_{stare} = \{s0, s1\}$, $A_{iesire} := \{T, F\}$
  $A_{s0} := s0$, $A_g(s0) := F$, $A_g(s1) := T$,
  $A_f(x, s0) := s0$, $A_f(y, s0) := s1$,
  $A_f(x, s1) := s0$, $A_f(y, s1) := s1$

Exemple

- $AUTOMAT = (S = \{intrare, stare, iesire\}, \Sigma)$
  $$\Sigma = \{s0 :\to stare, f : intrare\ stare \to stare,$$
  $$g : stare \to iesire\}$$
- $AUTOMAT$-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:
  $B_{intrare} = B_{stare} = B_{iesire} := \mathbb{N}$
  $B_{s0} := 0$, $B_f(m, n) := m + n$, $B_g(n) := n + 1$

Subalgebre

(S, Σ) - signatură multisortată

(A_S, A_Σ), (B_S, B_Σ) (S, Σ)-algebre

(B_S, B_Σ) este (S, Σ)-subalgebră a lui (A_S, A_Σ) dacă

- $B_s \subseteq A_s$ or. $s \in S$,
- $B_\sigma = A_\sigma$ 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. $(b_1, \ldots, b_n) \in B_{s_1} \times \cdots \times B_{s_n}$.

$B = \{B_s\}_{s \in S}$ este parte stabilă a lui $A = \{A_s\}_{s \in S}$

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 parte stabilă.

- $B_2 = \{\emptyset, \mathbb{N}\} \cup \{\{n\}, \mathbb{N} \setminus \{n\}\}$
  este parte stabilă ($n \in \mathbb{N}$ 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 parte stabilă a lui A

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 proprietăţi:

- $s_1 \leq s_2 \Rightarrow A_{s_1} \subseteq A_{s_2}$
- $\sigma \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$, $w_1 \leq w_2 \Rightarrow A_\sigma^{w_2,s_2}(\mathbf{x}) = A_\sigma^{w_1,s_1}(\mathbf{x})$ oricare $\mathbf{x} \in A_{w_1}$.
- Semantica unui modul in CafeObj este o algebră ordonat-sortată (`mod!`) sau o clasă de algebre ordonat-sortate (`mod*`).