Lecţia 2:

Sisteme clasice: Limbajul de specificare Z

v1.0.1

G Ştefănescu — Universitatea Bucureşti

Metode de Dezvoltare Software, Sem.2
Februarie 2007— Iunie 2007

Sisteme clasice: Limbajul de specificare Z

Cuprins:

- *Introducere*
- Mulţimi, tipuri
- Relaţii, funcţii
- Scheme
- Concluzii, diverse, etc.

Introducere

Fundamente Este bazat pe

- Logica de ordinul întâi (cu predicate)
- Teoria mulţimilor în axiomatizarea Zermelo-Fraenkel
- Notaţii auxiliare

Origine

- J.-R. Abrial, Oxford University Computing Laboratory (1980)

Standard International

- ISO/IEC 13568 (2002)

..Introducere

Referinţe

- Mike Spivey, The Z Reference Manual, Prentice Hall 1992
- Jim Woodcock, Jim Davies, Using Z, Prentice Hall 1996
- Noi folosim un rezumat: “Bernhard Beckert, Formal Specifcation of Software: The Z Specification Language”

Variatii

- Object-Z - extensie pentru specificaţii OO
- HOL-Z - includere în Isabelle, un demonstrator de teoreme bazat pe HOL (High Order Logic)

..Introducere

Tool-uri

- Stil Latex
- Verificator de tipuri
- Sistem de deducţie

Notă: Nu sunt "tool"-uri pentru simulare/execuţie/testare [limbaj de specificare].

Cuprins:

- Introducere
- *Mulţimi, tipuri*
- Relaţii, funcţii
- Scheme
- Concluzii, diverse, etc.

Operatori incluşi

Operatori logici

- $\neg$ (negaţie), $\wedge$ (conjuncţie), $\vee$ (disjuncţie), $\Rightarrow$ (implicaţie), $\Leftrightarrow$ (echivalenţă)

Egalitate

- $=$ (egalitate) - pe toate tipurile

Cuantificare

- $Q\ x_1 : S_1; \ldots x_n : S_n \mid p \bullet q$, unde Q este $\forall, \exists, \exists_1$
- Sensul este:
  $\forall\ x_1 : S_1; \ldots x_n : S_n(p \Rightarrow q)$, resp. $\exists\ x_1 : S_1; \ldots x_n : S_n(p \wedge q)$
- Abreviem: $\forall\ x : T \bullet q$ pentru $\forall\ x : T \mid true \bullet q$

Mulţimi

Mulţimi definite prin enumerare

- $\{e_1, \ldots, e_n\}$
- elementele de mai sus trebuie să aibă tipuri compatibile
- Exemplu: $\{2, 0, 1, 3\}$

Mulţimi definite prin proprietăţi

- $\{x : T \mid pred(x) \bullet expr(x)\}$
- reprezintă toate elementele care rezultă evaluând $expr(x)$ pentru toţi x de tip T care satisfac $pred(x)$
- *Exemplu* $\{x : \mathbb{N} \mid prime(x) \bullet x * x\}$ (pătrate de numere prime)

..Mulţimi

Abrevieri

- $\{x : T \mid pred(x)\}$ este o abreviere pentru $\{x : T \mid pred(x) \bullet x\}$
- Exemplu: $\mathbb{N} = \{x : \mathbb{Z} \mid x \geq 0\}$

Mulţimea vidă

- $\varnothing = \{x : T \mid \texttt{false}\}$
- de notat că este *tipizată* (mulţimea vidă *din* T)

Operaţii cu mulţimi

Apartenenţă $\in$

Sbmulţime $\subseteq$

- $S_1 \subseteq S_2 \Leftrightarrow \{\forall x : S_1 \mid x : S_2\}$
- S_1 şi S_2 trebuie să aibă acelaşi tip

Operatorul de generare a submulţimilor (power-set) $\mathbb{P}$

- $S' \in \mathbb{P} S \Leftrightarrow S' \subseteq S$

Produs cartezian $\times$

- $(x_1, \ldots, x_n) \in S_1 \times \ldots \times S_n \Leftrightarrow (x_1 \in S_1 \wedge \ldots \wedge x_n \in S_n)$

..Operaţii cu mulţimi

Reuniune $\cup$, $\bigcup$

- mulţimile trebuie să aibă acelaşi tip T
- $x \in S_1 \cup S_2 \Leftrightarrow (x \in S_1 \vee x \in S_2)$
- $x \in \bigcup S \Leftrightarrow (\exists S' : S \bullet x \in S')$

Intersecţie $\cap$, $\bigcap$ - similar

Diferenţa mulţimilor $\setminus$ - uzual

Tipuri

Tipuri predefinite

- *Exemplu* $\mathbb{Z}$ cu:
  — constantele 0,1,2,3,...
  — funcţiile $+, -, *, /$
  — predicatele $<, \leq, >, \geq$

Mulţimi

- Orice mulţime poate fi utilizată ca tip de dată

Tipuri de bază

- mulţimi particulare date

- *Exemplu* $[Person]$

Definiţii de noi tipuri

Definiţii de noi tipuri

Exemplu

- $weekDay ::= mon \mid tue \mid wed \mid thu \mid fri \mid sat \mid sun$

Exemplu

- $Tree ::= leaf\langle\langle \mathbb{Z} \rangle\rangle \mid node\langle\langle Tree \times Tree \rangle\rangle$
- $[Tree]$ este generat de $leaf, node$
- proprietăţi:
  $\forall x_1, y_1, x_2, y_2 : Tree \mid$
  $\quad node(x_1, y_1) = node(x_2, y_2) \bullet (x_1 = x_2 \wedge y_1 = y_2)$
  $\forall x_1, x_2 : Z \mid leaf(x_1) = leaf(x_2) \bullet x_1 = x_2$
  $\forall x : Z;\ y, z \in Tree \bullet leaf(x) \neq node(y, z)$

Tipuri compuse

Tipul mulţimi de tip dat $\mathbb{P}T$

- Toate mulţimile de tip T

Tipul produs cartezian $T_1 \times \ldots \times T_n$

- Tipul tuplelor $(t_1, \ldots, t_n)$ cu $t_i \in T_i$

Sumar al tipurilor

$T = \mathbb{Z}$

$T = [Type]$

$T ::= \ldots$ (tipuri liber definite)

$T ::= \mathbb{P}T'$

$T ::= T_1 \times \ldots \times T_n$

Variabile

Declarări de variabile

- *Exemple* $x : \mathbb{Z}$, $sold : \mathbb{P}\,Seat$
- Variabilele se pot referi la tipuri ori mulţimi

Abrevieri sintactice

Abrevieri

- trebuie să fie nerecursive
- pot fi generice
- *Exemple*

$$numberPairs == \mathbb{Z} \times \mathbb{Z}$$

$$pairWithNumber[S] == \mathbb{Z} \times S$$

Abrevieri vs. Tipuri generate

- $weekDay2 == \{mon, tue, wed, thu, fri, sat, sun\}$ (abreviere)
- $weekDay2 ::= mon \mid tue \mid wed \mid thu \mid fri \mid sat \mid sun$ (tip nou)

Definiţii axiomatice

Format

$$
\begin{array}{|l}
\mathit{SymbolDeclarations} \\
\hline
\mathit{ConstrainingPredicates}
\end{array}
$$

Exemplu

$$
\begin{array}{|l}
\mathbb{N}_1 : \mathbb{P}\,\mathbb{Z} \\
\hline
\forall z \in \mathbb{Z} \bullet (z \in \mathbb{N}_1 \leftrightarrow z \geq 1)
\end{array}
$$

Sisteme clasice: Limbajul de specificare Z

Cuprins:

- Introducere
- Mulţimi, tipuri
- *Relaţii, funcţii*
- Scheme
- Concluzii, diverse, etc.

Relaţii

Relaţii pe tipuri/mulţimi

- $S \leftrightarrow T$ este tipul (mulţimea) *relaţiilor* între tipurile (mulţimile) S şi T
- $S \leftrightarrow T = \mathbb{P}(S \times T)$

Notaţie

- $a \rightarrowtail b$ denotă perechea (a,b), dacă $(a,b) \in S \leftrightarrow T$

Notă: In exemplele de mai jos, dacă nu se specifică altfel, se presupune ca relaţia R este de tipul $S \times T$.

Operaţii pe relaţii

Domeniu $\operatorname{dom} R$

- $\operatorname{dom} R = \{a : S;\ b : T \mid a \rightarrowtail b \in R \bullet a\}$

Codomeniu (range) $\operatorname{ran} R$

- $\operatorname{ran} R = \{a : S;\ b : T \mid a \rightarrowtail b \in R \bullet b\}$

Restricţii de relaţii

- $S' \lhd R = \{a : S;\ b : T \mid a \rightarrowtail b \in R \wedge a \in S' \bullet a \rightarrowtail b\}$
- $R \rhd T' = \{a : S;\ b : T \mid a \rightarrowtail b \in R \wedge b \in T' \bullet a \rightarrowtail b\}$
- $S' \unlhd R = \{a : S;\ b : T \mid a \rightarrowtail b \in R \wedge a \notin S' \bullet a \rightarrowtail b\}$
- $R \unrhd T' = \{a : S;\ b : T \mid a \rightarrowtail b \in R \wedge b \notin T' \bullet a \rightarrowtail b\}$

..Operaţii pe relaţii

Relaţia inversă R^{-1}

- $R^{-1} = \{a : S;\ b : T \mid a \rightarrowtail b \in R \bullet b \rightarrowtail a\}$

Compunerea $R \mathbin{⨾} R'$ pentru $R : S \leftrightarrow T$ şi $R' : T \leftrightarrow U$

- $R \mathbin{⨾} R' = \{a : S;\ b : T;\ c : U \mid a \rightarrowtail b \in R \wedge b \rightarrowtail c \in R' \bullet a \rightarrowtail c\}$

Inchideri pentru $R : S \leftrightarrow S$

iterare: $R^n = R \mathbin{⨾} R^{n-1}$
identitate: $R^0 = \{a : S \mid true \bullet a \rightarrowtail a\}$
reflexiv-tranzitivă: $R^* = \cup\{n : \mathbb{N} \mid true \bullet R^n\}$
tranzitivă: $R^+ = \cup\{n : \mathbb{N} \mid n \geq 1 \bullet R^n\}$
simetrică: $R^s = R \cup R^{-1}$
reflexivă: $R^r = R \cup R^0$

Funcţii

Funcţiile ca relaţii

- funcţiile sunt relaţii speciale

Notaţie

- In loc de $\leftrightarrow$ folosim notaţiile:
  $\rightarrow$ - funcţii (totale)
  $\nrightarrow$ - funcţii parţiale

Funcţii parţiale

- $f \in S \nrightarrow T \Leftrightarrow f \in S \leftrightarrow T \wedge$
  $\forall a : S, b : T, b' : T \mid (a \rightarrowtail b \in f \wedge a \rightarrowtail b' \in f) \bullet b = b'$

Funcţii totale

- $f \in S \rightarrow T \Leftrightarrow f \in S \nrightarrow T \wedge$
  $\forall a : S \mid \exists b : T \bullet a \rightarrowtail b \in f$

Notaţii λ pentru funcţii

Forma generală (p - precondiţie; e - expresie)

- $\lambda a : S \mid p \bullet e$

Exemplu

$$\begin{array}{|l}
double : \mathbb{Z} \pfun \mathbb{Z} \\
\hline
double = \lambda n : \mathbb{Z} \mid n \geq 0 \bullet n + n
\end{array}$$

care este echivalent cu

$$\begin{array}{|l}
double : \mathbb{Z} \pfun \mathbb{Z} \\
\hline
double = \{n : \mathbb{N} \mid true \bullet n + n\}
\end{array}$$

Notaţii infixate şi prefixate

Sintaxă

- Relaţiile şi funcţiile se pot defini prefixat ori infixat
- poziţiile parametrilor se indică cu “_”

Exemplu (relaţii)

$$
\begin{array}{|l}
even\ \_ : \mathbb{P}\,\mathbb{Z} \\
\hline
\forall x : \mathbb{Z} \bullet (even\ x \Leftrightarrow (\exists y : \mathbb{Z} \bullet x = y + y))
\end{array}
$$

care este echivalent cu

$$
\begin{array}{|l}
even\ \_ : \mathbb{P}\,\mathbb{Z} \\
\hline
even = \{x : \mathbb{Z} \mid (\exists y : \mathbb{Z} \bullet x = y + y)\}
\end{array}
$$

Clase de funcţii

Notaţii

$\rightarrowtail\!\!\!\!\!+\;$ - funcţii parţiale injective

$\rightarrowtail$ - funcţii totale injective

$\twoheadrightarrow\!\!\!\!\!\!\!+\;$ - funcţii parţiale surjective

$\twoheadrightarrow$ - funcţii totale surjective

$\rightarrowtail\!\!\!\!\!\twoheadrightarrow$ - funcţii totale bijective

Trei definiţii pentru *abs*

Relaţie (in formă infixată)

$$
\begin{array}{|l}
\_\ abs\ \_ : \mathbb{Z} \leftrightarrow \mathbb{N} \\
\hline
\forall m \in \mathbb{Z}, n \in \mathbb{N} \bullet (m\ abs\ n) \leftrightarrow (m = n \vee -m = n)
\end{array}
$$

Funcţie (in formă infixată)

$$
\begin{array}{|l}
abs : \mathbb{Z} \rightarrow \mathbb{Z} \\
\hline
abs = (\lambda m \in \mathbb{Z} \mid m \leq 0 \bullet -m) \cup (\lambda m \in \mathbb{Z} \mid m \geq 0 \bullet m)
\end{array}
$$

Funcţie (in formă prefixată)

$$
\begin{array}{|l}
abs\_ : \mathbb{Z} \nrightarrow \mathbb{Z} \\
\hline
\forall x \in \mathbb{Z} \mid x \leq 0 \bullet x = -(abs\ x) \\
\forall x \in \mathbb{Z} \mid x \geq 0 \bullet x = abs\ x
\end{array}
$$

Construcţii finite

Submulţimi finite în $\mathbb{Z}$

- $m..n = \{n' : \mathbb{N} \mid m \leq n' \wedge n' \leq n\}$

Mulţimi finite

- $\mathbb{F}\,T$ constă din mulţimile finite din $\mathbb{P}\,T$

$$
\begin{array}{|l}
\hline
[S] \\
\mathbb{F}\,\mathbb{P}(\mathbb{P}\,S)) \\
\hline
\mathbb{F} = \{s \in \mathbb{P}\,S \mid (\exists n \in \mathbb{N} \bullet (\exists f : 1..n \rightarrowtail\!\!\!\!\twoheadrightarrow s \bullet \mathit{true}))\} \\
\hline
\end{array}
$$

Mulţimi finite

Cardinalitate

- $\mathbb{F}\, T$ constă din mulţimile finite din $\mathbb{P}\, T$

$$
\begin{array}{|l}
\hline
[S] \\
\# : \mathbb{F}\, S \longrightarrow \mathbb{N} \\
\hline
\forall s \in \mathbb{F}\, S;\ n : \mathbb{N} \bullet (n = \# s \leftrightarrow (\exists f : 1 \mathinner{..} n \rightarrowtail\!\!\twoheadrightarrow s \bullet \mathit{true})) \\
\hline
\end{array}
$$

Funcţii finite

Notaţii

- ⇻ - funcţii finite
  $S ⇻ T = \{f : S \longrightarrow T \mid \mathrm{dom} f \in \mathbb{F} S\}$
- ⇻ - funcţii injective finite
  $S ⤕ T = \{f : S \rightarrowtail T \mid \mathrm{dom} f \in \mathbb{F} S\}$
- etc.

Secvenţe

Definiţie

- $\mathrm{seq}\, T == \{s : \mathbb{Z} \nrightarrow T \mid \mathrm{dom}\, s = 1..\#s\}$
- secvenţele sunt funcţii (deci, iterativ, relaţii, mulţimi)
- lungimea lui s este $\#s$

Notaţie

- Secvenţa $\{1 \rightarrowtail x_1, 2 \rightarrowtail x_2, \ldots, n \rightarrowtail x_n\}$
  se scrie $\langle x_1, x_2, \ldots, x_n \rangle$

Exemplu concatenarea secvenţelor

- $s \frown t == s \cup (\lambda n \in \mathbb{Z} \mid n \in \#s + 1..\#s + \#t \bullet n - \#s) \,⨾\, t$

Sisteme clasice: Limbajul de specificare Z

Cuprins:

- Introducere
- Mulţimi, tipuri
- Relaţii, funcţii
- *Scheme*
- Concluzii, diverse, etc.

Scheme

Forma generală

$$\begin{array}{|l}
\hline \textit{Name} \\
\textit{SymbolDeclarations} \\
\hline \textit{ConstrainingPredicates} \\
\hline
\end{array}$$

Notaţie liniară

- $Name \mathrel{\widehat{=}} [SymbolDeclarations \mid ConstrainingPredicates]$

Formă particulară (fară predicate)

$$\begin{array}{|l}\hline \multicolumn{1}{l}{\textit{Name}} \\ \quad \textit{SymbolDeclarations} \\ \hline \end{array}$$

Notaţia liniară

- $Name \mathrel{\widehat{=}} [SymbolDeclarations]$

..Scheme

Exemplu: Bilete de teatru

$[Seat]$

$[Person]$

TicketsForPerformance0

$$seating : \mathbb{P}\, Seat$$
$$sold : Seat \pfun Person$$

$$\operatorname{dom} sold \subseteq seating$$

Schemele ca tipuri/mulţimi

Schema

$$
\begin{array}{|l}
\hline
\textit{Name} \\
x_1 : T_1 \\
\ldots \\
x_n : T_n \\
\hline
\textit{ConstrainingPredicates} \\
\hline
\end{array}
$$

poate fi privită ca mulţimea/tipul următoarelor tuple

$$
\begin{aligned}
&\textit{Name} = \\
&\{x_1 : T_1;\ \ldots;\ x_n : T_n \mid \textit{ConstrainingPredicates} \bullet (x_1, \ldots, x_n)\}
\end{aligned}
$$

Incluziunea schemelor

Incluziune

- schemele pot fi folosite în:
  - alte scheme;
  - mulţimi definite cu proprietăţi;
  - cunatificări
- se introduce numele schemei în partea de declaraţie

Sens

- se adaugă *declaraţiile* şi *predicatele de constrângere* la părţile corespunzatoare ale structurii în care se include

..Incluziune schemelor

Exemplu

$$\begin{array}{|l}
\hline
\textit{NumberInSet} \\
a : \mathbb{Z} \\
c : \mathbb{P}\,\mathbb{Z} \\
\hline
a \in c \\
\hline
\end{array}$$

$$\{NumberInSet \mid a = 0 \bullet c\}$$

este acelaşi lucru cu

$$\{a : \mathbb{Z}, c : \mathbb{P}\,\mathbb{Z} \mid a \in c \land a = 0 \bullet c\}$$

(toate mulţimile de întregi care conţin pe 0)

Schemele ca predicate

Incluziune

- schemele pot fi folosite ca predicate în:
  - alte scheme;
  - mulţimi definite cu proprietăţi;
  - cunatificări
- se introduce numele schemei în partea de predicate

Sens

- se adaugă *predicatele de constrângere* (nu şi partea de declaraţie) la parţile corespunzatoare ale structurii în care se include

..Schemele ca predicate

Exemplu

*NumberIn*01

$a : \mathbb{Z}$

$c : \mathbb{P}\,\mathbb{Z}$

$a \in c$

$c \subseteq \{0, 1\}$

$$\forall\, a : \mathbb{Z};\; c : \mathbb{P}\,\mathbb{Z} \mid \mathit{NumberIn}01 \bullet \mathit{NumberInSet}$$

este acelaşi lucru cu

$$\forall\, a : \mathbb{Z};\; c : \mathbb{P}\,\mathbb{Z} \mid a \in c \land c \subseteq \{0, 1\} \bullet a \in c$$

Scheme generice

In definiţia schemelor se pot folosi variabile pentru tipuri ori mulţimi.

Exemplu

$$\begin{array}{|l}
\hline
\multicolumn{1}{l}{\mathit{NumberInSetGeneric}[X]} \\
a : X \\
c : \mathbb{P}X \\
\hline
a \in c \\
\hline
\end{array}$$

Atunci

$$\mathit{NumberInSetGeneric}[\mathbb{Z}] = \mathit{NumberInSet}$$

Redenumirea variabilelor în scheme

Variabilele din scheme pot fi redenumite

Exemplu

$$numberInSet[a/q, c/s]$$

este echivalent cu

$$\begin{array}{|l}
\hline
\textit{NumberInSet} \\
q : \mathbb{Z} \\
s : \mathbb{P}\mathbb{Z} \\
\hline
q \in s \\
\hline
\end{array}$$

Conjuncţii de scheme

Schemele pot fi *compuse conjunctiv*.

Exemplu Dacă sunt date două scheme

$ConDis1$	$ConDis2$
$a : A;\ b : B$	$b : B;\ c : C$
P	Q

atunci următoarele formulări sunt echivalente

$ConDis1 \wedge ConDis2$

$a : A;\ b : B;\ c : C$
P Q

Disjuncţii de scheme

Schemele pot fi *compuse disjunctiv*.

Exemplu Similar, dacă sunt date două scheme

$$\begin{array}{|l}\hline \textit{ConDis}1 \\ a : A;\ b : B \\ \hline P \\ \hline\end{array} \qquad \begin{array}{|l}\hline \textit{ConDis}2 \\ b : B;\ c : C \\ \hline Q \\ \hline\end{array}$$

atunci următoarele formulări sunt echivalente

$$\textit{ConDis}1 \vee \textit{ConDis}2 \qquad \begin{array}{|l}\hline a : A;\ b : B;\ c : C \\ \hline P \vee Q \\ \hline\end{array}$$

Exemplu: Bilete la teatru

Specificare informală

Teatru: Biletele pentru premieră se vând doar prietenilor

Specificare în Z

$$Status \ ::= \ standard \mid firstNight$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad Friends \quad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}} \\
friends : \mathbb{P}\, Person \\
status : Status \\
sold : Seat \pfun Person \\
\hline
status = firstNight \Rightarrow \operatorname{ran} sold \subseteq friends \\
\hline
\end{array}$$

..Exemplu: Bilete la teatru

$$TicketsForPerformance1 \mathrel{\widehat{=}} TicketsForPerformance0 \land Friends$$

si

$$\begin{array}{|l}
\hline
\textit{TicketsForPerformance1} \\
\textit{Friends} \\
\textit{TicketsForPerformance0} \\
\hline
\end{array}$$

sunt echivalente cu formularea

$$\begin{array}{|l}
\hline
\textit{TicketsForPerformance1} \\
friends : \mathbb{P}\, Person;\ status : Status \\
sold : Seat \pfun Person;\ seating : \mathbb{P}\, Seat \\
\hline
status = firstNight \Rightarrow \operatorname{ran} sold \subseteq friends \\
\operatorname{dom} sold \subseteq seating \\
\hline
\end{array}$$

Scheme normalizate

Normalizare

O schemă este normalizată dacă

- variabile sunt tipizate
- ... *dar nu* restrânse la submulţimi de tipuri

Exemplu O schemă (stânga) şi normalizata ei (dreapta)

$x : \mathbb{N}$
P

$x : \mathbb{Z}$
$x \geq 0$ P

Negaţia unei scheme

Negaţia unei scheme se obţine *negând partea de predicate* în *schema normalizată*.

Exemplu O schemă (stânga) şi normalizata ei (dreapta)

$$\begin{array}{|l} x : \mathbb{N} \\ \hline P \end{array} \qquad \begin{array}{|l} x : \mathbb{Z} \\ \hline x \in \mathbb{N} \\ P \end{array}$$

Ultima negată este

$$\begin{array}{|l} x : \mathbb{Z} \\ \hline \neg(x \in \mathbb{N} \wedge P) \end{array}$$

Schemele ca operaţii

Stări

- o stare este o asignare de valori variabilelor
- o schemă descrie o mulţime de stări

Operaţii

- pentru a descrie o operaţie, o schemă trebuie să descrie perechi de stări (pre/post)

Notaţie

- variabilele se *decorează* cu prim $'$ pentru a specifica că se referă la stările de după (post)
- scheme intregi pot fi decorate

..Schemele ca operaţii

Exemplu

$$NumberInSet'$$

este tot una cu

$$\begin{array}{|l}
\hline
\textit{NumberInSet}' \\
a' \in \mathbb{Z} \\
c' : \mathbb{P}\,\mathbb{Z} \\
\hline
a' \in c' \\
\hline
\end{array}$$

Alte decoraţii

- variabilele de intrare se decorează cu "?"
- variabilele de ieşire se decorează cu "!"

Exemplu: Bilete la teatru

Informal

Teatru: Vânzare de bilete

$$
\begin{array}{|l}
\hline
\textit{Purchase}0 \\
\hline
\textit{TicketsForPerformance}0 \\
\textit{TicketsForPerformance}0' \\
s? : \textit{Seat} \\
p? : \textit{Person} \\
\hline
s? \in \textit{seating} \setminus \operatorname{dom} \textit{sold} \\
\textit{sold}' = \textit{sold} \cup \{s? \mapsto p?\} \\
\textit{seating}' = \textit{seating} \\
\hline
\end{array}
$$

(schema nu are variabile de ieşire)

..Exemplu: Bilete la teatru

$$Response ::= okey \mid sorry$$

$$\begin{array}{|l}
\hline
\textit{Success} \\
r! : Response \\
\hline
r! = okey \\
\hline
\end{array}$$

In fine,

$$Purchase0 \wedge Success$$

este o schema care returnează un rezultat de vânzare de bilete cu succes.

..Schemele ca operaţii

Forma generală

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\textit{StateSpace}} \\
x_1 : T_1;\ \ldots;\ x_n : T_n \\
\hline
inv(x_1, \ldots, x_n) \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\textit{Operation}} \\
\textit{StateSpace} \\
\textit{StateSpace}' \\
i_1? : U_1;\ \ldots;\ i_m? : U_m \\
o_1! : V_1;\ \ldots;\ o_p! : V_p \\
\hline
pre(i_1?, \ldots, i_m?, x_1, \ldots, x_n) \\
op(i_1?, \ldots, i_m?, x_1, \ldots, x_n, x'_1, \ldots, x'_n, o_1!, \ldots, o_p!) \\
\hline
\end{array}
$$

Operatorul Δ

Operatorul Δ

- $\Delta Schema$ abreviază $Schema \wedge Schema'$

Forma generală cu Δ

$$
\begin{array}{|l}
\hline
\textit{Operation} \\
\Delta StateSpace \\
i_1? : U_1;\ \ldots;\ i_m? : U_m \\
o_1! : V_1;\ \ldots;\ o_p! : V_p \\
\hline
pre(i_1?, \ldots, i_m?, x_1, \ldots, x_n) \\
op(i_1?, \ldots, i_m?, x_1, \ldots, x_n, x'_1, \ldots, x'_n, o_1!, \ldots, o_p!) \\
\hline
\end{array}
$$

Operatorul Ξ

Operatorul Ξ

- $\Xi Schema$ abreviază $\Delta Schema \wedge (x_1 = x_1' \wedge \ldots \wedge x_n = x_n')$ unde $x_1, \ldots, x_n$ sunt variabilele declarate în *Schema*

Forma generală cu Ξ

$$
\begin{array}{|l}
\hline
\textit{Operation} \\
\Xi StateSpace \\
i_1? : U_1;\ \ldots;\ i_m? : U_m \\
o_1! : V_1;\ \ldots;\ o_p! : V_p \\
\hline
pre(i_1?, \ldots, i_m?, x_1, \ldots, x_n) \\
op(i_1?, \ldots, i_m?, x_1, \ldots, x_n, o_1!, \ldots, o_p!) \\
\hline
\end{array}
$$

Notă: Operatorul Ξ indică faptul că operaţia nu schimbă starea.

Exemple Δ, Ξ

Exemplu

- $\Xi NumberInSet$

este echivalent cu

$$
\begin{array}{|l}
\hline
NumberInSet \\
NumberInSet' \\
\hline
a = a' \\
c = c' \\
\hline
\end{array}
$$

..Exemple Δ, Ξ

Exemplu

- Teatru: Vanzare de tichete, dar doar prietenilor la premieră

$$
\begin{array}{|l}
\hline
\textit{Purchase}1 \\
\Delta \textit{TicketsForPerformance}1 \\
s? : \textit{Seat} \\
p? : \textit{Person} \\
\hline
s? \in \textit{seating} \setminus \operatorname{dom} \textit{sold} \\
\textit{status} = \textit{firstNight} \Rightarrow (p? \in \textit{friends}) \\
\textit{sold}' = \textit{sold} \cup \{s? \mapsto p?\} \\
\textit{seating}' = \textit{seating} \\
\textit{status}' = \textit{status} \\
\textit{friends}' = \textit{friends} \\
\hline
\end{array}
$$

..Exemple Δ, Ξ

$$\begin{array}{|l}
\hline
\textit{NotAvailable} \\
\Xi \textit{ticketsForPerformance}1 \\
s? : \textit{Seat} \\
p? : \textit{Person} \\
\hline
s? \in \operatorname{dom} \textit{sold} \lor (\textit{status} = \textit{firstNight} \land \neg p? \in \textit{friends}) \\
\hline
\end{array}$$

$$\begin{array}{|l}
\hline
\textit{Failure} \\
r! : \textit{Response} \\
\hline
r! = \textit{sorry} \\
\hline
\end{array}$$

In fine,

- $\textit{TicketsForPerformance} \mathrel{\widehat=}$
  $(\textit{Purchase}1 \land \textit{Success}) \lor$
  $(\textit{NotAvailable} \land \textit{Failure})$

Cuantificarea (ascunderea) variabileleor in scheme

Cuantificarea schemelor

$$\forall x : S \bullet Schema$$

$$\exists x : S \bullet Schema$$

operatorul de quantificare existentială se mai numeşte ascundere de varaibile *(variable hiding)*

Exemplu

$$\exists a : \mathbb{Z} \bullet NumberInSet$$

este echivalent cu

$$c \in \mathbb{P}\mathbb{Z}$$

$$\exists a : \mathbb{Z} \bullet a \in c$$

Compunerea schemelor de operaţii

Definiţie:

- Schemele de operaţii se pot compune folosind $\fatsemi$, unde
  - fiecare variabilă cu $'$ din prima schemă trebuie sa apară fară $'$ în a doua
  - aceste variabile se identifică şi se ascund (nu mai sunt vizibile în exterior)

Compunerea schemelor de operaţii

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{Op1} \\
x_1 : T_1; \ \ldots; \ x_p : T_p \\
z_1 : V_1; \ \ldots; \ z_n : V_n \\
z'_1 : V_1; \ \ldots; \ z'_n : V_n \\
\hline
op1(x_1, \ldots, x_p, \\
z_1, \ldots, z_n, z'_1, \ldots, z'_n) \\
\hline
\end{array}
\qquad
\begin{array}{|l}
\hline
\multicolumn{1}{l}{Op2} \\
y_1 : U_1; \ \ldots; \ y_q : U_q \\
z_1 : V_1; \ \ldots; \ z_n : V_n \\
z'_1 : V_1; \ \ldots; \ z'_n : V_n \\
\hline
op2(y_1, \ldots, y_q, \\
z_1, \ldots, z_n, z'_1, \ldots, z'_n) \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{Op1 \fatsemi Op2} \\
x_1 : T_1; \ \ldots; \ x_p : T_p \\
y_1 : U_1; \ \ldots; \ y_q : U_q \\
z_1 : V_1; \ \ldots; \ z_n : V_n \\
z'_1 : V_1; \ \ldots; \ z'_n : V_n \\
\hline
\exists z''_1 : V_1; \ \ldots; \ z''_n : V_n \\
op1(x_1, \ldots, x_p, z_1, \ldots, z_n, z''_1, \ldots, z''_n) \\
op2(y_1, \ldots, y_q, z''_1, \ldots, z''_n, z'_1, \ldots, z'_n) \\
\hline
\end{array}
$$

Exemplu

$$Purchase1 \fatsemi Purchase1[s?/s2?]$$

este echivalent cu

$$
\begin{array}{|l}
\hline
\Delta TicketsForPerformance1 \\
s? : Seat;\ s2? : Seat;\ p? : Person \\
\hline
s? \in seating \setminus \operatorname{dom} sold \\
s2? \in seating \setminus \operatorname{dom}(sold \cup \{s? \mapsto p?\}) \\
status = fisrtNight \Rightarrow (p? \in friends) \\
sold' = sold \cup \{s? \mapsto p?, s2 \mapsto p?\} \\
seating' = seating \\
status' = status \\
friends' = friends \\
\hline
\end{array}
$$

Sisteme clasice: Limbajul de specificare Z

Cuprins:

- Introducere
- Mulţimi, tipuri
- Relaţii, funcţii
- Scheme
- *Concluzii, diverse, etc.*

Concluzii, diverse, etc.

a se insera...