

CS-21xx:MetDezvSoft

Lecţia 2:

Sisteme clasice: Limbajul de specificare Z

v1.0.1

G Ştefănescu — Universitatea București

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
- Notații auxiliare

Origine

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

Standard International

• ISO/IEC 13568 (2002)



..Introducere

Referințe

- 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 Specification of Software: The Z Specification Language"

Variatii

- Object-Z extensie pentru specificații 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 deducție

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



Sisteme clasice: Limbajul de specificare Z

Cuprins:

- Introducere
- Mulţimi, tipuri
- Relații, funcții
- Scheme
- Concluzii, diverse, etc.



Operatori incluşi

Operatori logici

 ¬ (negaţie), ∧ (conjuncţie), ∨ (disjuncţie), ⇒ (implicaţie), ⇔ (echivalenţă)

Egalitate

• = (egalitate) - pe toate tipurile

Cuantificare

- $Q x_1 : S_1; \dots x_n : S_n \mid p \bullet q$, unde Q este $\forall, \exists, \exists_1$
- Sensul este:

$$\forall x_1: S_1; \ldots x_n: S_n(p \Rightarrow q), \text{ resp. } \exists x_1: S_1; \ldots x_n: S_n(p \land q)$$

• Abreviem: $\forall x : T \bullet q$ pentru $\forall x : T \mid true \bullet q$

Mulţimi

Mulțimi definite prin enumerare

- $\bullet \ \{e_1,\ldots,e_n\}$
- 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 toți x de tip T care satisfac pred(x)
- *Exemplu* $\{x : \mathbb{N} \mid prime(x) \bullet x * x\}$ (pătrate de numere prime)

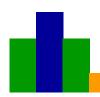
..Mulţimi

Abrevieri

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

Mulţimea vidă

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



Operații cu mulțimi

Apartenență ∈

Sbmulţime ⊆

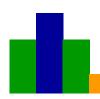
- $S_1 \subseteq S_2 \Leftrightarrow \{ \forall x : S_1 \mid x : S_2 \}$
- S_1 şi S_2 trebuie să aibă acelaşi tip

Operatorul de generare a submulțimilor (power-set) $\mathbb P$

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

Produs cartezian ×

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



.. Operații cu mulțimi

Reuniune \cup , \bigcup

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

Intersecție \cap , \cap - similar

Diferența mulțimilor \ - uzual

Tipuri

Tipuri predefinite

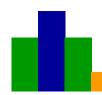
- $Exemplu \mathbb{Z}$ cu:
 - constantele 0,1,2,3,...
 - funcțiile +, -, *, /
 - predicatele $<, \le, >, \ge$

Mulţimi

• Orice mulțime poate fi utilizată ca tip de dată

Tipuri de bază

- mulţimi particulare date
- Exemplu [Person]



Definiții de noi tipuri

Definiții de noi tipuri

Exemplu

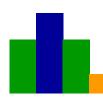
• $weekDay ::= mon \mid tue \mid wed \mid thu \mid fri \mid sat \mid sun$

Exemplu

- $Tree ::= leaf \langle \langle \mathbb{Z} \rangle \rangle \mid node \langle \langle Tree \times Tree \rangle \rangle$
- [Tree] este generat de leaf, node
- proprietăți:

$$\forall x_1, y_1, x_2, y_2 : Tree \mid$$

$$node(x_1, y_1) = node(x_2, y_2) \bullet (x_1 = x_2 \land y_1 = y_2)$$
 $\forall x_1, x_2 : Z \mid leaf(x_1) = leaf(x_2) \bullet x_1 = x_2$
 $\forall x : Z; y, z \in Tree \bullet leaf(x) \neq node(y, z)$



Tipuri compuse

Tipul mulțimi 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

$$T = \mathbb{Z}$$
 $T = [Type]$
 $T ::= \dots$ (tipuri liber definite)
 $T ::= \mathbb{P} T'$
 $T ::= T_1 \times \dots \times T_n$

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



Definiții axiomatice

Format

SymbolDeclarations
ConstrainingPredicates

Exemplu



Sisteme clasice: Limbajul de specificare Z

Cuprins:

- Introducere
- Mulţimi, tipuri
- Relații, funcții
- Scheme
- Concluzii, diverse, etc.

Relații

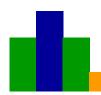
Relații pe tipuri/mulțimi

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

Notație

• $a \rightarrow b$ 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 relația R este de tipul $S \times T$.



Operații pe relații

Domeniu dom R

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

Codomeniu (range) ran R

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

Restricții de relații

•
$$S' \triangleleft R = \{a : S; b : T \mid a \rightarrowtail b \in R \land a \in S' \bullet a \rightarrowtail b\}$$

$$\bullet \ R \triangleright T' = \{a : S; \ b : T \mid a \rightarrowtail b \in R \land b \in T' \bullet a \rightarrowtail b\}$$

•
$$S' \triangleleft R = \{a : S; b : T \mid a \rightarrowtail b \in R \land a \notin S' \bullet a \rightarrowtail b\}$$

•
$$R \triangleright T' = \{a : S; b : T \mid a \rightarrowtail b \in R \land b \notin T' \bullet a \rightarrowtail b\}$$

.. Operații pe relații

Relația inversă R^{-1}

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

Compunerea $R \circ R'$ pentru $R : S \leftrightarrow T$ şi $R' : T \leftrightarrow U$

 $\bullet \ R_{9}^{\circ}R' = \{a: S; \ b: T; \ c: U \mid a \rightarrowtail b \in R \land b \rightarrowtail c \in R' \bullet a \rightarrowtail c\}$

Inchideri pentru $R: S \leftrightarrow S$

iterare: $R^n = R \circ R^{n-1}$

identitate: $R^0 = \{a : S \mid true \bullet a \rightarrowtail a\}$

reflexiv-tranzitivă: $R^* = \bigcup \{n : \mathbb{N} \mid true \bullet R^n \}$

tranzitivă: $R^+ = \bigcup \{n : \mathbb{N} \mid n \ge 1 \bullet R^n \}$

simetrică: $R^s = R \cup R^{-1}$

reflexivă: $R^r = R \cup R^0$

Funcții

Funcțiile ca relații

• funcțiile sunt relații speciale

Notație

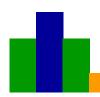
- In loc de ← folosim notaţiile:
 - → funcții (totale)
 - → funcții parțiale

Funcții parțiale

•
$$f \in S \rightarrow T \Leftrightarrow f \in S \hookrightarrow T \land \forall a : S, b : T, b' : T \mid (a \rightarrowtail b \in f \land a \rightarrowtail b' \in f) \bullet b = b'$$

Funcții totale

$$\bullet f \in S \to T \Leftrightarrow f \in S \to T \land \\ \forall a : S \mid \exists b : T \bullet a \rightarrowtail b \in f$$



Notații λ pentru funcții

Forma generală (p - precondiție; e - expresie)

•
$$\lambda a : S \mid p \bullet e$$

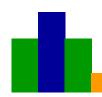
Exemplu

$$\frac{double : \mathbb{Z} \to \mathbb{Z}}{double = \lambda n : \mathbb{Z} \mid n \ge 0 \bullet n + n}$$

care este echivalent cu

$$double : \mathbb{Z} \to \mathbb{Z}$$

$$double = \{n : \mathbb{N} \mid true \bullet n + n\}$$



Notații infixate și prefixate

Sintaxă

- Relațiile și funcțiile se pot defini prefixat ori infixat
- pozițiile parametrilor se indică cu "_"

Exemplu (relații)

$$even_{-}: \mathbb{P}\mathbb{Z}$$

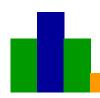
$$\forall x : \mathbb{Z} \bullet (even \ x \Leftrightarrow (\exists y : \mathbb{Z} \bullet x = y + y))$$

care este echivalent cu

$$even_{-}: \mathbb{P}\mathbb{Z}$$

$$even_{-}: \mathbb{P}\mathbb{Z}$$

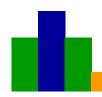
$$even_{-}: \mathbb{Z} \mid (\exists y : \mathbb{Z} \bullet x = y + y) \}$$



Clase de funcții

Notații

- >--- funcții parțiale injective
- → funcții totale injective
- ---- funcții parțiale surjective
- → funcţii totale surjective
- >---> funcții totale bijective



Trei definiții pentru abs

Relație (in formă infixată)

Funcție (in formă infixată)

$$abs: \mathbb{Z} \to \mathbb{Z}$$

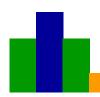
$$abs = (\lambda m \in \mathbb{Z} \mid m \le 0 \bullet -m) \cup (\lambda m \in \mathbb{Z} \mid m \ge 0 \bullet m)$$

Funcție (in formă prefixată)

$$abs_{-}: \mathbb{Z} \to \mathbb{Z}$$

$$\forall x \in \mathbb{Z} \mid x \leq 0 \bullet x = -(abs \ x)$$

$$\forall x \in \mathbb{Z} \mid x \geq 0 \bullet x = abs \ x$$



Construcții finite

Submulțimi finite în $\mathbb Z$

• $m..n = \{n' : \mathbb{N} \mid m \le n' \land n' \le n\}$

Mulţimi finite

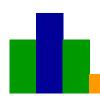
• $\mathbb{F} T$ constă din mulțimile finite din $\mathbb{P} T$



Mulțimi finite

Cardinalitate

• $\mathbb{F} T$ constă din mulțimile finite din $\mathbb{P} T$



Funcții finite

Notații

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

Secvențe

Definiție

- $\bullet \operatorname{seq} T == \{s : \mathbb{Z} + T \mid \operatorname{dom} s = 1..\#s\}$
- secvențele sunt funcții (deci, iterativ, relații, mulțimi)
- lungimea lui s este #s

Notație

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

Exemplu concatenarea secvențelor

•
$$s \cap t == s \cup (\lambda n \in \mathbb{Z} \mid n \in \#s+1..\#s+\#t \bullet n-\#s) \circ t$$



Sisteme clasice: Limbajul de specificare Z

Cuprins:

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



Scheme

Forma generală

Notație liniară

• Name = [SymbolDeclarations | ConstrainingPredicates]



..Scheme

Formă particulară (fară predicate)

__Name_____ SymbolDeclarations

Notația liniară

• Name = [SymbolDeclarations]



..Scheme

Exemplu: Bilete de teatru

[Seat] [Person]

_TicketsForPerformance0_____

 $seating: \mathbb{P}Seat$

 $sold: Seat \rightarrow Person$

 $dom sold \subseteq seating$



Schemele ca tipuri/mulţimi

Schema

Name $x_1:T_1$ \dots $x_n:T_n$ ConstrainingPredicates

poate fi privită ca mulțimea/tipul următoarelor tuple

Name =
$$\{x_1: T_1; \ldots; x_n: T_n \mid ConstrainingPredicates \bullet (x_1, \ldots, x_n)\}$$



Incluziunea schemelor

Incluziune

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

Sens

• se adaugă *declarațiile* și *predicatele de constrângere* la părțile corespunzatoare ale structurii în care se include



..Incluziune schemelor

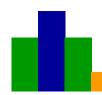
Exemplu

NumberInSet $a: \mathbb{Z}$ $c: \mathbb{P} \mathbb{Z}$ $a \in c$

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

este același lucru cu

 $\{a: \mathbb{Z}, c: \mathbb{P}\mathbb{Z} \mid a \in c \land a = 0 \bullet c\}$ (toate mulțimile de întregi care conțin pe 0)



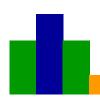
Schemele ca predicate

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 declarație) la parțile corespunzatoare ale structurii în care se include



..Schemele ca predicate

Exemplu

NumberIn01 $a: \mathbb{Z}$ $c: \mathbb{P} \mathbb{Z}$ $a \in c$ $c \subseteq \{0,1\}$

 $\forall a : \mathbb{Z}; c : \mathbb{PZ} \mid NumberInO1 \bullet NumberInSet$

este același lucru cu

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



Scheme generice

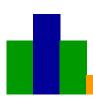
In definiția schemelor se pot folosi variabile pentru tipuri ori mulțimi.

Exemplu

NumberInSetGeneric[X] a: X $c: \mathbb{P}X$ $a \in c$

Atunci

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



Redenumirea variabilelor în scheme

Variabilele din scheme pot fi redenumite

Exemplu

numberInSet[a/q, c/s]

este echivalent cu

NumberInSet_

 $q:\mathbb{Z}$

 $s: \mathbb{P}\mathbb{Z}$

 $q \in s$



Conjuncții de scheme

Schemele pot fi compuse conjunctiv.

Exemplu Dacă sunt date două scheme

atunci următoarele formulări sunt echivalente

$$ConDis1 \land ConDis2$$

$$\begin{array}{|c|c|}
\hline
a:A;b:B;c:C\\
\hline
P\\Q
\end{array}$$



Disjuncții de scheme

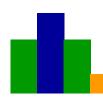
Schemele pot fi compuse disjunctiv.

Exemplu Similar, dacă sunt date două scheme

atunci următoarele formulări sunt echivalente

$$ConDis1 \lor ConDis2$$

$$\begin{array}{|c|c|}
\hline
a:A;b:B;c:C\\
\hline
P\vee Q
\end{array}$$



Exemplu: Bilete la teatru

Specificare informală

Teatru: Biletele pentru premieră se vând doar prietenilor

Specificare în Z

Status ::= standard | firstNight

Friends

 $friends: \mathbb{P}$ Person

status: Status

 $sold: Seat \rightarrow Person$

 $status = firstNight \Rightarrow ransold \subseteq friends$

Slide 2.44



..Exemplu: Bilete la teatru

$TicketsForPerformance1 \cong TicketsForPerformance0 \land Friends$

si

_TicketsForPerformance1 _____ Friends TicketsForPerformance0

sunt echivalente cu formularea

_TicketsForPerformance1 _____

 $friends: \mathbb{P} Person; status: Status$

 $sold: Seat \rightarrow Person; seating: \mathbb{P} Seat$

 $status = firstNight \Rightarrow ransold \subseteq friends$ $dom sold \subseteq seating$



Scheme normalizate

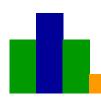
Normalizare

O schemă este normalizată dacă

- variabile sunt tipizate
- ... dar nu restrânse la submulțimi de tipuri

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

 $x:\mathbb{N}$ P



Negația unei scheme

Negația unei scheme se obține negând partea de predicate în schema normalizată.

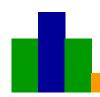
Exemplu O schemă (stânga) și normalizata ei (dreapta)

$$x:\mathbb{N}$$
 P

$$\begin{array}{c}
x : \mathbb{Z} \\
\hline
x \in \mathbb{N} \\
P
\end{array}$$

Ultima negată este

$$\overline{\neg(x \in \mathbb{N} \land P)}$$



Schemele ca operații

Stări

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

Operații

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

Notație

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



..Schemele ca operații

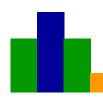
Exemplu

NumberInSet'

este tot una cu

Alte decorații

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



Exemplu: Bilete la teatru

Informal

Teatru: Vânzare de bilete

Purchase0

TicketsForPerformance0

TicketsForPerformance0'

s?: Seat

p?: Person

 $s? \in seating \setminus dom sold$ $sold' = sold \cup \{s? \mapsto p?\}$ seating' = seating

(schema nu are variabile de ieşire)



..Exemplu: Bilete la teatru

 $Response ::= okey \mid sorry$

Success r!: Response r! = okey

In fine,

 $Purchase0 \land Success$

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



..Schemele ca operații

Forma generală

```
StateSpace
x_1:T_1; ...; x_n:T_n
inv(x_1,...,x_n)
```

```
Operation
StateSpace
StateSpace'
i_1?: U_1; ...; i_m?: U_m
o_1!: V_1; ...; o_p!: V_p

pre(i_1?,...,i_m?,x_1,...,x_n)
op(i_1?,...,i_m?,x_1,...,x_n,x_1',...,x_n',o_1!,...,o_p!)
```



Operatorul Δ

Operatorul Δ

ΔSchema abreviază Schema ∧ Schema¹

Forma generală cu Δ



Operatorul **E**

Operatorul **E**

• Ξ *Schema* abreviază Δ *Schema* \wedge $(x_1 = x'_1 \wedge ... \wedge x_n = x'_n)$ unde $x_1, ..., x_n$ sunt variabilele declarate în *Schema*

Forma generală cu **E**

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



Exemple Δ , Ξ

Exemplu

• *ENumberInSet*

este echivalent cu

NumberInSet NumberInSet'

$$a = a'$$

$$c = c'$$



..Exemple Δ , Ξ

Exemplu

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

```
 Durchase1 $$\Delta TicketsForPerformance1 $$s?: Seat $$p?: Person $$ s? ∈ seating \ dom sold $$ status = firstNight ⇒ (p? ∈ friends) $$ sold' = sold <math>\cup \{s? \mapsto p?\} $$ seating' = seating $$ status' = status $$ friends' = friends $$
```



..Exemple Δ , Ξ

```
NotAvailable
\Xi ticketsForPerformance1
s?: Seat
p?: Person
s? \in dom sold \lor (status = firstNight \land \neg p? \in friends)
Failure
r!: Response
```

In fine,

TicketsForPerformance =
 (Purchase1 ∧ Success) ∨
 (NotAvailable ∧ Failure)

r! = sorry

Cuantificarea (ascunderea) variabileleor in scheme

Cuantificarea schemelor

 $\forall x : S \bullet Schema$

 $\exists x : S \bullet Schema$

operatorul de quantificare existentială se mai numește 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 operații

Definiție:

- Schemele de operații se pot compune folosind 3, 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 operații

$$\begin{bmatrix}
 Op1 \\
 x_1 : T_1; \dots; x_p : T_p \\
 z_1 : V_1; \dots; z_n : V_n \\
 z'_1 : V_1; \dots; z'_n : V_n
 \end{bmatrix}$$
 $op1(x_1, \dots, x_p, z_1, \dots, z'_n)$

```
Op2
y_1: U_1; ...; y_q: U_q
z_1: V_1; ...; z_n: V_n
z'_1: V_1; ...; z'_n: V_n
op2(y_1, ..., y_q, z_1, ..., z'_n)
```

```
\begin{array}{c}
Op1_{\S}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\\
\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)
\end{array}
```



Exemplu

Purchase1 § Purchase1 [s?/s2?]

este echivalent cu

```
\Delta TicketsForPerformance1
s?: Seat; s2?: Seat; p?: Person
s? \in seating \setminus domsold
s2? \in seating \setminus 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
```



Sisteme clasice: Limbajul de specificare Z

Cuprins:

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



Concluzii, diverse, etc.

a se insera...