Spécification formelle

LANGAGE Z

École Nationale d'Ingénieurs de Carthage
3ème année Ingénierie des Systèmes Intelligents

E. Menif Abassi

Plan du module

2

Plan du chapitre

3

1. Introduction
2. Abstraction des données
3. Schémas
4. Preuve de propriétés

Plan du chapitre

4

1. Introduction
2. Abstraction des données
3. Schémas
4. Preuve de propriétés

Introduction

Jean-Raymond Abrial

- Créée par Jean-Raymond Abrial
- Années 70
- Z en hommage au mathématicien *Ernest Zermelo* (un mathématicien allemand ayant eu des contributions fondamentales à la théorie des ensembles)
- Université Oxford (UK)
- Standard ISO/IEC 13568:2002

Ernest Zermelo

Introduction

- Bases mathématiques
  - Théorie des ensembles ZF (axiomes de **Zermelo-Fraenkel**)
  - La logique des prédicats
- Concept clé = Schéma
- Deux types d'abstractions
  - Abstraction des données
  - Abstraction des opérations

Plan du chapitre

7

1. Introduction
2. Abstraction des données
   a. Types
   b. Abréviation
   c. Relations et fonctions
   d. Quantificateurs
   e. Types libres
   f. Séquences
   g. Multi-ensembles
3. Schémas
4. Preuve de propriétés

Abstraction des données

8

- Abstraction des données est décrite par:
  - Définition des types
  - Définition des constantes globales
  - Déclaration des états

→ Ensembles + fonctions + relations + séquences

→ Schémas

Plan du chapitre

9

1. Introduction
2. Abstraction des données
   a. Types
   b. Abréviation
   c. Relations et fonctions
   d. Quantificateurs
   e. Types libres
   f. Séquences
   g. Multi-ensembles
3. Schémas
4. Preuve de propriétés

Abstraction des données

10

a. Types

- Z est fortement typé ⇒ chaque variable, constante et expression a un type
  - Détecter les erreurs plus facilement
  - Écrire de meilleures spécifications
  - Permettre une automatisation de la vérification du typage
- Dans Z, un type est un ensemble« maximal ». Un type est un ensemble qui n'est inclus dans aucun autre (sauf lui-même).

Abstraction des données

11

a. Types

- Le type décrit l'ensemble des valeurs qu'un objet de ce type peut avoir ⇒ un objet est un élément d'un ensemble
- Une déclaration d'un objet est l'association d'un ensemble à une variable (objet) telle que la variable peut prendre les valeurs (les termes) de cet ensemble (qui peut être un type)
  - x:E ⇔ $\forall x \bullet x \in E$
  - $x_1, x_2, x_3,\ldots, x_n$: E ⇔ x_1 : E, x_2 : E, x_3 : E,…, x_n : E
  - **Déclaration axiomatique**:

Déclartion
Prédicat

- Deux catégories de types et ensembles: Types simples + Types composés

Abstraction des données

12

a. Types

- Les types simples:
  - Types de base (par convention en majuscule) : sont les ensembles dont les éléments ne peuvent être énumérés. Tous les types de base sont, par définition, de cardinalité infinie. On énumère au début d'une spécification les noms des ensembles que nous souhaitons avoir.
    - **Syntaxe**: **[ident,…,ident]**
    - **Contrainte**: Ne doivent pas avoir été déclaré globale auparavant.
    - **Portée**: De la définition jusqu'à la fin de la spécification
    - **Exemples**: [TITRE], [TITRE, AUTEUR, CODE, ANNEE]

Abstraction des données

13

a. Types

- Les types simples:
  - Types prédéfinis: les ensembles définis par Z
    - $\emptyset$ ou {}: ensemble vide (polymorphe)
    - $\mathbb{Z}$: ensemble des entiers (+, -, *, div, mod, ≤, ≥, <,>)
  - Types définis : correspondent à des définitions d'ensembles
    - Ensembles définis en extension: **{élément$_1$, élément$_2$,…, élément$_n$}**
      - **NOMS ≙ {Yassine, Youssef, Yasmine}**
      - **ENS ≙ {1, 2, 3}**
    - Ensembles définis par intervalle: **n..m**
      - **ENS ≙ 1..3 c'est pareil que d'écrire ENS ≙ {1, 2, 3}**

Abstraction des données

14

a. Types

- Les types simples:
  - Ensembles définis en compréhension: introduisent des types obtenus par spécialisation de types non vides déjà définis
    - **Syntaxe**: $\{x: E \mid p(x) \bullet f(x)\}$ où **E** est un type (ensemble), f est une fonction dont le domaine est **E** et p est un prédicat: L'ensemble de tous les $f(x)$ tels que x**: E** satisfait le prédicat $p(x)$.
    - $\{x: E \bullet f(x)\} \triangleq \{x: E \mid vrai \bullet f(x)\}$
    - $\{x: E \mid p(x)\} \triangleq \{x: E \mid p(x) \bullet x\}$

Abstraction des données

15

a. Types

- Les types simples:
  - Types définis en compréhension: introduisent des types obtenus par spécialisation de types non vides déjà définis
    - **Exemples**:
    - $\{x : \mathbb{Z} \mid 1 \leq x \leq 20 \bullet x * 2\}$ = **{2,4,6,…,40}**
    - $\{x : 2..8 \bullet x * x\}$ = **{4,9,16,25,36,49,64}**
    - $\{x : \mathbb{Z} \mid x \bmod 2 = 0\}$ = **Les entiers pairs**

Abstraction des données

16

a. Types

- Les types composés: construits à l'aide des constructeurs d'ensembles (ensemble des parties, produit cartésien) et des schémas
  - Ensemble des parties:
    - $\mathbb{P}$: sous-ensembles
    - $\mathbb{F}$: sous-ensembles finis
  - Produit cartésien: si x et y sont deux objets de types t et u respectivement alors la paire (x,y) est un objet de type $t \times u$

Abstraction des données

17

a. Types

- Les types composés:
  - Produit cartésien:
    - **Exemple**: **LIVRE == TITRE × AUTEUR × CODE × ANNEE**

livre1, *livre2*: **LIVRE**

livre1 = **(Introduction aux méthodes formelles, Monin, 1111111, 2000)**
livre2 = **(Software engineering, Sommerville, 2222222, 2010)**

Abstraction des données

18

a. Types

- Opérations ensemblistes:

Opérateur	Sens	Définition
$\#S$	Cardinal	$\lvert S \rvert$
$x \notin S$	Non appartenance	$\triangleq \neg(x \in S)$
$S \subseteq T$	Inclusion	$\triangleq (\forall x : S \bullet x \in T)$
$S \subset T$	Inclusion stricte	$\triangleq S \subseteq T \wedge S \neq T$
$\mathbb{P} S$	Sous-ensemble	$T \in \mathbb{P} S \Leftrightarrow (\forall x \bullet x \in T \Rightarrow x \in S)$
$\mathbb{F} S$	Sous-ensembles fini	$\triangleq \{T : \mathbb{P} S \mid fini(T)\}$
$S \times T$	Produit cartésien	$\triangleq \{x \in S, y \in T \bullet (x, y)\}$ $(x,y) \in (S \times T) \Leftrightarrow \exists y, z \bullet (y \in S \wedge z \in T \wedge x = (y,z))$

Abstraction des données

19

a. Types

- Opérations ensemblistes:

Opérateur	Sens	Définition
$S \cap T$	Intersection	$S, T : \mathbb{P} X \quad \triangleq \{x : X \mid x \in S \land x \in T\}$
$S \cup T$	Union	$S, T : \mathbb{P} X \quad \triangleq \{x : X \mid x \in S \lor x \in T\}$
$S \setminus T$	Différence	$S, T : \mathbb{P} X \quad \triangleq \{x : X \mid x \in S \land x \notin T\}$
$\bigcap SS$	Intersection distribuée	$SS : \mathbb{P}(\mathbb{P} X) \quad \triangleq \{x : X \mid \forall S : SS \bullet x \in S\}$
$\bigcup SS$	Union distribuée	$SS : \mathbb{P}(\mathbb{P} X) \quad \triangleq \{x : X \mid \exists S : SS \bullet x \in S\}$
$\min S$	Minimum	$S : \mathbb{F}\mathbb{N} \mid S \neq \emptyset \quad \min S \in S \land (\forall x \in S \bullet x \geq \min S)$
$\max S$	Maximum	$S : \mathbb{F}\mathbb{N} \mid S \neq \emptyset \quad \max S \in S \land (\forall x \in S \bullet x \leq \max S)$

Plan du chapitre

20

Abstraction des données

21

b. Abréviation

- Introduction d'une nouvelle constante globale
- Syntaxe:
  - x == y
  - x≙ y
- Exemples:
  - LIVRE ≙ TITRE × AUTEUR × CODE × ANNEE
  - NATPAIR ≙ $\{x: \mathbb{Z} \mid x \geq 0 \bullet 2*x\}$
  - $\mathbb{N} \triangleq \{x: \mathbb{Z} \mid x \geq 0\}$ et $\mathbb{N}_1 \triangleq \{x: \mathbb{Z} \mid x \geq 1\}$

Abstraction des données

22

Exercices

1. #{Ø,{1},{2},{1,2} ,{1,3} ,{2,3}} = 6
2. #$\{n \in \mathbb{N} \mid n < 10$ et n est pair$\}$ = 5
3. #{0, 2, 4, 6, 8, …} = l'ensemble est infini et Z ne peut prendre sa cardinalité
4. #{chat, chien, oiseau, lion} = 4
5. #Ø = 0
6. $\emptyset \in \mathbb{P}(\mathbb{N})$? oui
7. $\emptyset \subset \mathbb{P}(\mathbb{N})$? oui
8. $\emptyset \in \mathbb{F}(\mathbb{N})$? oui
9. $\emptyset \subset \mathbb{F}(\mathbb{N})$? oui

Abstraction des données

Exercices

10. A-t-on toujours $\mathbb{F}(E) \subseteq \mathbb{P}(E)$ quelque soit E? **oui**
11. A-t-on toujours $\#\mathbb{F}(E) \leq \#\mathbb{P}(E)$ quelque soit E? **oui**
12. Un ensemble E est fini si est seulement si $\mathbb{P}(E)$ est fini? **oui**
13. Écrire {4,9,16,25} en compréhension

$$\{x{:}\mathbb{Z} \mid (x \leq 5 \wedge x \geq 2) \vee (x \leq -2 \wedge x \geq -5) \bullet x*x\}$$

Plan du chapitre

Abstraction des données

25

c. Relations et fonctions

- Relation
  - La relation est un ensemble de liens entre éléments ⇒ un sous-ensemble du produit cartésien (binaire et orienté)
    - Soient X et Y deux ensembles alors l'ensemble des relations entre X et Y est définie par R: $X \leftrightarrow Y$ ($X \leftrightarrow Y \triangleq \mathbb{P}(X \times Y)$)
    - Les éléments de X faisant partie de la relation forment le **domaine** de la relation
    - Les éléments de Y faisant partie de la relation forment le **codomaine** de la relation

Abstraction des données

26

c. Relations et fonctions

- Relation
  - Un élément de la relation est appelé lien: $x \mapsto y$
    - x est l'**antécédent**
    - y est l'**image**
  - **Exemple**: On considère des étudiants et des notes obtenues par les étudiants. Les étudiants sont représentés par un type de base et les notes par un intervalle.

    ***notes*: ETUDIANT ↔ 1..20**

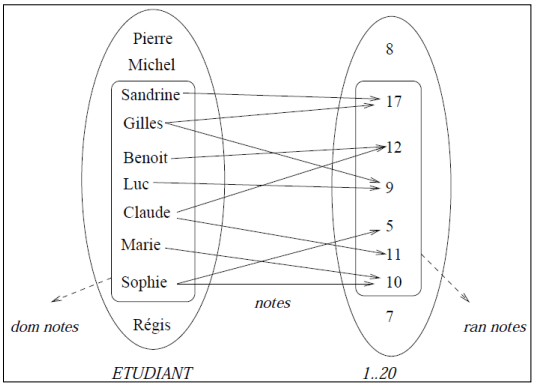
Abstraction des données

27

c. Relations et fonctions

- Relation

(Sandrine,17)∈ *notes*,
(Gilles,17) ∈ *notes*



Représentation graphique de la relation *notes*

Abstraction des données

28

c. Relations et fonctions

- Relation: Opérateurs (R: $\mathbb{P}(X \times Y)$)
  - **dom R**:(**Domaine**) $\{x:X \mid (\exists y:Y \bullet (x,y) \in R)\}$
    - dom *notes* = {Sandrine, Gilles, Benoit, Luc, Claude, Marie, Sophie}
  - **ran R**: (**Codomaine**) $\{y:Y \mid (\exists x:X \bullet (x,y) \in R)\}$
    - ran *notes* = {5, 9, 10, 11, 12, 17}
  - **id R**: (**Identité**) $\{x:X \bullet x \mapsto x\}$
  - **R˜**: (**Inverse**) $\{y:Y, x:X \mid (x,y) \in R\}$
    - *notes* ˜={(17, Sandrine), (17, Gilles) , (12, Benoit), (12, Claude), …}

Abstraction des données

29

c. Relations et fonctions

- Relation: Opérateurs (**R**: $\mathbb{P}(X \times Y)$, **S**: $\mathbb{P}\, X$, **S'**: $\mathbb{P}\, Y$ et **R'**: $\mathbb{P}(Y \times Z)$)
  - **R ⨾ R'**:(**Composition**) $\{x{:}X,\ z{:}Z \mid (\exists y{:}Y \bullet (x,y) \in R \land (y,z) \in R')\}$
  - **R^k**: (**Composition récurrente**) Si **R:** $X \leftrightarrow X$ alors $R^0 \triangleq id\ X$ et $R^{k+1} \triangleq R^k \fatsemi R$
  - **R(S)**: (**Image relationnelle**) $\{y{:}Y \mid (\exists x{:}S \bullet (x,y) \in R)\}$
  - **S ◁ R**: (**Restriction du domaine**) $\{x{:}X, y{:}Y \mid x \in S \land (x,y) \in R\}$
  - **R ▷ S'**: (**Restriction du codomaine**) $\{x{:}X, y{:}Y \mid y \in S' \land (x,y) \in R\}$

Abstraction des données

30

c. Relations et fonctions

- Relation: Opérateurs **R**: $\mathbb{P}(X \times Y)$, **S**: $\mathbb{P}\, X$, **S'**: $\mathbb{P}\, Y$ et **R'**: $\mathbb{P}(Y \times Z)$
  - **S ⩤ R**: (**Soustraction de domaine**) $(X \setminus S) \triangleleft R$
  - **R ⩥ S'**: (**Soustraction de codomaine**) $R \rhd (Y \setminus S')$
  - **R ⊕ R'**: (**Surcharge**) (dom R' ⩤ R) ∪ R'
    = $\{x{:}X, y{:}Y \mid (x,y) \in R' \lor (x \notin \text{dom } R' \land (x,y) \in R)\}$

Abstraction des données

31

c. Relations et fonctions

- Relation: Opérateurs – Exemples

$R : \mathbb{Z} \leftrightarrow \mathbb{Z}$

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

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

$R = \{(1, 2), (1,7), (2,5), (7,-1), (7,-71)\}$

$E = \{1,3,5\}$

$F = \{x : \mathbb{Z} \mid x < 3\}$

$E \lhd R = \{(1, 2), (1,7)\}$

$E \lhd\!\!\!\!- R = \{(2,5), (7,-1), (7,-71)\}$

$R \rhd F = \{(1,2),(7,-1), (7,-71)\}$

$R \rhd\!\!\!\!- F = \{(1,7), (2,5)\}$

Abstraction des données

32

c. Relations et fonctions

- Relation: Opérateurs – Exemples

$R, S : \mathbb{Z} \leftrightarrow \mathbb{Z}$

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

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

$R = \{(1, 2), (1,7), (2,5), (7,-1), (7,-71)\}$

$S = \{x, y : \mathbb{Z} \mid y = 2*x + 1\}$

$E = \{1,3,5\}$

$F = \{x : \mathbb{Z} \mid x < 3\}$

$R \oplus S = S$

$S \oplus R = \{..., (0,1), (1,2), (1,7), (2,5), (3,7), ..., (7,-1), (7,-71), (8,17), ...\}$

$R \fatsemi S = \{(1, 5), (1, 15), (2, 11), (7,-1), (7,-141)\}$

$S \fatsemi R = \{(0,2), (0,7), (3,-1), (3,-71)\}$

$R(\!|E|\!) = \{2,7\}$

Abstraction des données

33

c. Relations et fonctions

- Relation: Définitions axiomatique et générique

$\_ \uparrow \_ : \mathbb{N} \leftrightarrow \mathbb{N}$

Le caractère – désigne l'emplacement des arguments

$\forall x,y: \mathbb{N} \bullet x \uparrow y \Leftrightarrow \exists\, z : \mathbb{N} \bullet x * z = y$

Le paramètre générique est placé entre [] La définition est applicable quelque soit le type de X

$[X]$

$\_ \subseteq \_ : \mathbb{P}X \leftrightarrow \mathbb{P}X$

$\forall S,T: \mathbb{P}X \bullet S \subseteq T \Leftrightarrow \exists\, x \in S \wedge x \in T$

Abstraction des données

34

c. Relations et fonctions

- Fonction
  - La fonction est un cas particulier de relation, dont les antécédents ont au plus une image
    - L'application d'une fonction f à un élément x s'écrit $f\ x$ ou $f(x)$
    - Une fonction est dite **interne** si le type de son domaine et de son codomaine est le même $f: X \rightarrow X$
    - Il existe plusieurs variantes de fonctions selon des contraintes de type "au plus" ou "au moins" s'appliquant aux images ou aux antécédents

Abstraction des données

35

c. Relations et fonctions

- Variantes de fonctions

Notation	Sens	Commentaire, Définition, Propriété
$X \pfun Y$	Fonctions	$\triangleq \{f : X \rel Y \mid (\forall x : dom\ f \bullet (\exists! y : Y \bullet x\ f\ y))\}$
$X \fun Y$	Fonctions totales	$\triangleq \{f : X \pfun Y \mid dom\ f = X\}$
$X \pinj Y$	Injections	$\triangleq \{f : X \pfun Y \mid (\forall x_1, x_2 : dom\ f \bullet f(x_1) = f(x_2) \Rightarrow x_1 = x_2)\}$
$X \inj Y$	Injections totales	$\triangleq \{f : X \pinj Y \mid dom\ f = X\}$ $\triangleq (X \pinj Y) \cap (X \fun Y)$
$X \psur Y$	Surjections	$\triangleq \{f : X \pfun Y \mid ran\ f = Y\}$
$X \sur Y$	Surjections totales	$\triangleq \{f : X \psur Y \mid dom\ f = X\}$ $\triangleq (X \psur Y) \cap (X \fun Y)$

Abstraction des données

36

c. Relations et fonctions

- Variantes de fonctions

Notation	Sens	Commentaire, Définition, Propriété
$X \bij Y$	Bijections	$\triangleq (X \pinj Y) \cap (X \psur Y)$
$X \tbij Y$	Bijections totales	$\triangleq (X \sur Y) \cap (X \inj Y)$
$X \ffun Y$	Fonctions finies	$\triangleq \{f : X \pfun Y \mid dom\ f \in \mathbb{F}\ X\}$
$X \finj Y$	Injections finies	$\triangleq (X \finj Y) \cap (X \pinj Y)$
$(\lambda D \mid P \bullet E)$	λ-expression	Tout objet dans D vérifie P, le résultat est E sinon les objets mêmes si E est omis
$(\mu D \mid P \bullet E)$	μ-expression	L'unique objet dans D qui vérifie P, le résultat est E sinon l'objet même si E est omis
succ: $\mathbb{N} \to \mathbb{N}$	Fonction successeur	$\triangleq \forall n : \mathbb{N} \bullet succ(n) = n+1$

E. Menif Abassi Spécification formelle ENICAR

Abstraction des données

37

c. Relations et fonctions

- Fonctions: Opérateurs – Exemples

$f, g, h, k : \mathbb{Z} \leftrightarrow \mathbb{Z}$

$f = \{(1, 2), (2,3), (3,5)\}$
$\forall x: \mathbb{Z} \bullet g(x) = x*x + 2$
$h = f \fatsemi g$
$\forall x: \mathbb{Z} \bullet k(x) = -x + 1$

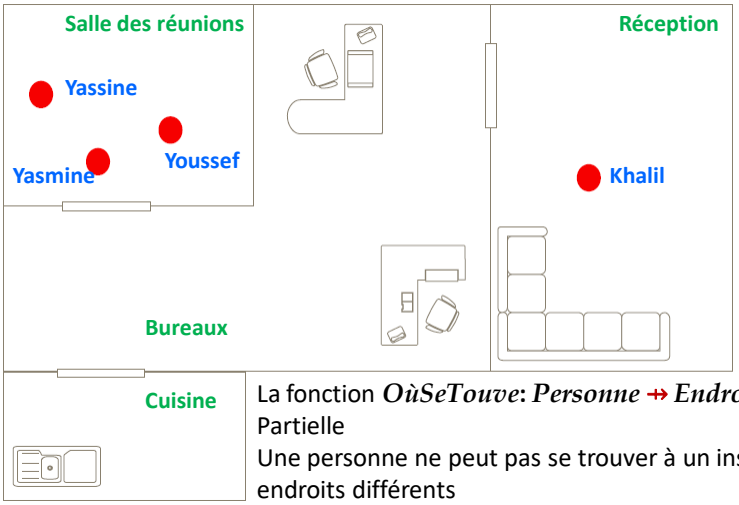
Dites si chacune de ces relations est une fonction, injective, surjective, bijective?

λ et μ expression

$Min: \mathbb{P}\mathbb{N} \pfun \mathbb{N}$

$Min = (\lambda\ s: \mathbb{P}\mathbb{N} \mid s \neq \emptyset \bullet (\mu\ x: s \mid (\forall\ y:s \mid y \neq x \bullet y > x)))$

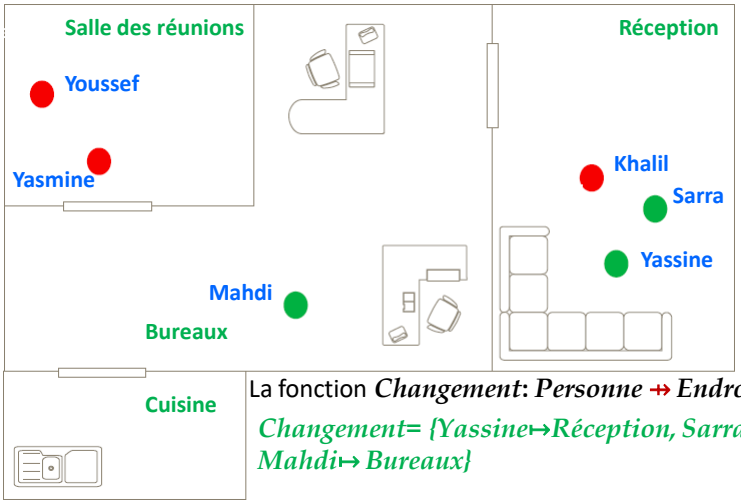
Abstraction des données



La fonction *OùSeTouve*: *Personne* ⇸ *Endroit* est une fonction Partielle
Une personne ne peut pas se trouver à un instant donné à deux endroits différents

OùSeTouve = {Yassine↦Réunion, Youssef ↦ Réunion, Yasmine ↦ Réunion, Khalil ↦Réception}

Abstraction des données



La fonction $Changement: Personne \nrightarrow Endroit$ est partielle

$Changement = \{Yassine \mapsto Réception, Sarra \mapsto Réception, Mahdi \mapsto Bureaux\}$

$OùSeTouveNouveau = OùSeTouve \oplus Changement = \{Yassine \mapsto Réception, Youssef \mapsto Réunion, Yasmine \mapsto Réunion, Khalil \mapsto Réception, Sarra \mapsto Réception, Mahdi \mapsto Bureaux\}$

Abstraction des données

c. Relations et fonctions

- Fonctions: Opérateurs – Exemples

$$
\begin{array}{|l}
som\_ : \mathbb{P}\mathbb{N} \rightarrow \mathbb{N} \\
moy\_ : \mathbb{P}\mathbb{N} \rightarrow \mathbb{N} \\
moyenne\_ : \text{ETUDIANT} \nrightarrow 0..20 \\
\hline
\\
som\ \emptyset = 0 \\
\forall s : \mathbb{P}\mathbb{N} \mid s \neq \emptyset \bullet (\exists\, x : \mathbb{N} \mid x \in s \bullet som\ s = x + som\ (s \setminus \{x\})) \\
moy\ \emptyset = 0 \\
\forall s : \mathbb{P}\mathbb{N} \mid s \neq \emptyset \bullet moy\ s = (som\ s)\ \mathbf{div}\ \#s \\
\mathbf{dom}\ moyenne = \mathbf{dom}\ notes \\
\forall e : \text{ETUDIANT} \mid e \in \mathbf{dom}\ notes \bullet moyenne\ e = moy\ (notes \lhd \{e\} \rhd)
\end{array}
$$

Plan du chapitre

41

1. Introduction
2. Abstraction des données
   a. Types
   b. Abréviation
   c. Relations et fonctions
   d. Quantificateurs
   e. Types libres
   f. Séquences
   g. Multi-ensembles
3. Schémas
4. Preuve de propriétés
5. Étude de cas

Abstraction des données

42

c. Quantificateurs

- L'utilisation des quantificateurs suit la syntaxe suivante:

$$Q\, x : a \mid p \bullet q$$

avec: Q le quantificateur, x la variable liée, a l'ensemble auquel appartient x, p la contrainte que doit vérifier x, q le prédicat

- Équivalence:

$(\exists x : a \mid p \bullet q) \Leftrightarrow (\exists x : a \bullet p \wedge q)$: Il existe un x dans a qui satisfait p tel que q est vrai

$(\forall x : a \mid p \bullet q) \Leftrightarrow (\forall x : a \bullet p \Rightarrow q)$: pour tout x dans a qui satisfait p, q est vrai

Abstraction des données

c. Quantificateurs

- **Exemples**:

$$\mathbb{N} : \mathbb{P}\mathbb{Z}$$
$$\forall z : \mathbb{Z} \bullet z \in \mathbb{N} \Leftrightarrow z \geq 0$$

$$som\_ : \mathbb{P}\mathbb{N} \rightarrow \mathbb{N}$$
$$som\ \emptyset = 0$$
$$\forall s : \mathbb{P}\mathbb{N} \mid s \neq \emptyset \bullet (\exists x : \mathbb{N} \mid x \in s \bullet som\ s = x + som\ (s \setminus \{x\}))$$

$$[X]$$
$$- \subseteq - : \mathbb{P}X \leftrightarrow \mathbb{P}X$$
$$\forall S, T : \mathbb{P}X \bullet S \subseteq T \Leftrightarrow \exists x \in S \wedge x \in T$$

Plan du chapitre

Abstraction des données

45

e. Types libres

- Définition des types par énumération et par récursivité
  - **Syntaxe**:
    - **TI**::= **constante**$_1$|…|**constante**$_n$
    - **TI**::= **constructeur**$_1$[⟪**source**$_1$⟫]|…|**constructeur**$_n$[⟪**source**$_n$⟫]
    - Le terme **constructeur**$_i$⟪**source**$_i$⟫ fait référence à une fonction injective qui accepte un objet de type **source**$_i$ et retourne un objet de type **TI**

Abstraction des données

46

e. Types libres

- Définition des types par énumération et par récursivité
  - **Exemples**:
    - MOIS ::= Janvier|Fevrier|Mars|Avril|…|Decembre
    - NAT::= zero | succ⟪NAT⟫
- Il n'y a pas de type booléen prédéfini
  - BOOLEEN::= vrai | faux

Plan du chapitre

47

1. Introduction
2. Abstraction des données
   a. Types
   b. Abréviation
   c. Relations et fonctions
   d. Quantificateurs
   e. Types libres
   f. Séquences
   g. Multi-ensembles
3. Schémas
4. Preuve de propriétés
5. Étude de cas

Abstraction des données

48

f. Séquences

- Introduit une notion d'ordre des éléments dans un ensemble
- Un ensemble ordonné
  - **Syntaxe**: **seq** X
  - **Définition** : seq $X \triangleq \{f : \mathbb{N}_1 \nrightarrow X \mid \exists n:\mathbb{N}_1 \bullet \mathrm{dom}\, f = 1..n\}$
  - **Exemples**:
    - **COULEUR ::= blanc | noir | jaune |rouge**
    - **(s:seq COULEUR) ≙ {(1,blanc), (2,noir),(3,jaune),(4,rouge)}**

Abstraction des données

49

f. Séquences

Notation	Sens	Commentaire, Définition, Propriété
$\langle x_1, x_2, \ldots, x_n \rangle$ $[x_1, x_2, \ldots, x_n]$	Séquence: Notation alternative	$\{1 \mapsto x_1, 2 \mapsto x_2, \ldots, n \mapsto x_n\}$ ou $\{(1, x_1), (2, x_2), \ldots, (n, x_n)\}$
$\langle\rangle$ ou $[]$	Séquence vide	
$s(i)$	$i^{ème}$ élément si $i \in 1..\#s$	
$\mathbf{seq}_1\ X$	Séquences finies non vides	$\triangleq \{f : \text{seq}\ X \mid \#f > 0\}$
$\mathbf{iseq}\ X$	Séquences injectives	$\triangleq \text{seq}\ X \cap (\mathbb{N} \rightarrowtail X)$
$\#s$	Cardinal	
$s \frown t$	concaténation	$\triangleq s \cup \{n : \text{dom}\ t \bullet n + \#s \mapsto t(n)\}$
$\mathbf{rev}\ s$	Inversion	$(\lambda n : \text{dom}\ s \bullet s(\#s - n + 1))$
$\mathbf{head}\ s$	Premier élément	$\forall s : \text{seq}_1 X \bullet \mathbf{head}\ s = s(1)$
$\mathbf{tail}\ s$	Liste sans le premier élément	$\forall s : \text{seq}_1 X \bullet \mathbf{tail}\ s = (\lambda n : 1..\ \#s\text{-}1 \bullet s(n + 1))$

Abstraction des données

50

f. Séquences: Il existe d'autres opérations sur les séquences

$s, t : \mathbf{seq}\ X$ et $f : \mathbb{N} \twoheadrightarrow X$

Notation	Sens	Commentaire, Définition, Propriété
$\mathbf{last}\ s$	Dernier élément	$\forall s : \text{seq}_1 X \bullet \mathbf{last}\ s = s(\#s)$
$\mathbf{front}\ s$	Liste sans le dernier élément	$\forall s : \text{seq}_1 X \bullet \mathbf{front}\ s = (1..(\#s\text{-}1)) \lhd s$
$\mathbf{squash}\ f$	Construit une séquence à partir d'une fonction	$(\mathbb{N} \twoheadrightarrow X) \rightarrow \text{seq}\ X$
$s \lceil A$	Filtre une séquence en ne considérant que les éléments de A	$\mathbf{squash}\ (s \rhd A)$
$I \rceil s$	Extrait une sous-séquence formée d'éléments avec des indices de I	$\mathbf{squash}\ (I \lhd s)$
$\langle A, B \rangle$ **partition** C	$A \cap B = \emptyset \wedge A \cup B = C$	_ partition _ : $(\mathbb{N} \rightarrow \mathbb{P}X) \leftrightarrow \mathbb{P}X$

Abstraction des données

51

f. Séquences

$s : \text{seq } \mathbb{N}$

$s = \langle 2, 4, 6, 8, 10, 12 \rangle$

head $s = 2$

last $s = 12$

front $s = \langle 2, 4, 6, 8, 10 \rangle$
$= \{(1, 2), (2, 4), (3, 6), (4, 8), (5, 10)\}$

tail $s = \langle 4, 6, 8, 10, 12 \rangle$
$= \{(1, 4), (2, 6), (3, 8), (4, 10), (5, 12)\}$

$s \frown \langle 1, 3, 5, 7, 9, 11 \rangle = \langle 2, 4, 6, 8, 10, 12, 1, 3, 5, 7, 9, 11 \rangle$

Plan du chapitre

52

Abstraction des données

53

g. Multi-ensembles

- *Bags*
- Un ensemble dans lequel un élément a plusieurs occurrences dans l'ensemble
  - **Syntaxe**: **bag** X
  - **Définition formelle**: **bag** $X \triangleq X \nrightarrow \mathbb{N}_1$
  - **Exemples**:
    - USCoins ≙ {penny, nickel, dime,quarter}
    - (b:bag USCoins) ≙ {penny↦2, nickel ↦3, quarter ↦4}

Abstraction des données

54

g. Multi-ensembles

Notation	Sens	Commentaire, Définition, Propriété
$[\![]\!]$	Multi-ensemble vide	
$[\![x_1,x_2, \dots , x_n]\!]$	Multi-ensemble en extension	$x_1,x_2, \dots , x_n:X$
$\mathbf{count}\ b$	Nombre de chaque élément x dans b	$\forall\, b:\mathbf{bag}\ X \bullet \mathbf{count} b = (\lambda x : X \bullet 0) \oplus b$
$b \# x$	Nombre d'occurrences de x dans b	$\forall x:X,\ b:\mathbf{bag}\ X \bullet b \# x = \mathbf{count} b\ x$
$n \otimes b$	Mise à l'échelle	$\forall n :\mathbb{N},\ x : X,\ b : \mathrm{bag}\ X$ $(n \otimes b) \# x = n * (b \# x)$

Abstraction des données

55

g. Multi-ensembles

Notation	Sens	Commentaire, Définition, Propriété
$x \sqsubseteq b$	Appartenance	$\forall x{:}X,\ b{:}\mathbf{bag}\ X \bullet (x \sqsubseteq b \Leftrightarrow x \in \mathbf{dom}\ b)$
$b \sqsubseteq c$	Inclusion	$\forall b, c{:}\mathbf{bag}\ X \bullet b \sqsubseteq c \Leftrightarrow (\forall x{:}X \bullet b\ \#\ x \leq c\ \#\ x)$
$b \uplus c$	Union	$\forall b, c{:}\mathbf{bag}\ X,\ x{:}X \bullet (b \uplus c)\ \#x = b\ \#\ x + c\ \#\ x$
$b \uplus c$	Différence	$\forall b, c{:}\mathbf{bag}\ X,\ x{:}X \bullet (b \uplus c)\ \#x = \max\{b\ \#\ x - c\ \#\ x, 0\}$

Abstraction des données

56

g. Multi-ensembles

b_1, b_2 : bag $\mathbb{N}$

$b_1 = [\![1, 1, 1, 2, 2, 4, 8, 8]\!]$
$b_2 = [\![3, 3, 5, 1, 4, 4]\!]$

$b_1\ \#8 = 2$

$b_1\ \#5 = 0$

count $b_1 = (\lambda x : \mathbb{N} \bullet 0) \oplus \{(1, 3), (2, 2), (4, 1), (8, 2)\}$

$2 \sqsubseteq b_1$
$[\![1, 2, 2]\!] \sqsubseteq b_1$

$b_1 \uplus b_2 = [\![1, 1, 1, 1, 2, 2, 3, 3, 4, 4, 4, 5, 8, 8]\!]$

Plan du chapitre

57

1. Introduction
2. Abstraction des données
3. Schémas
   a. Définition
   b. Schémas et types
   c. Schémas et déclarations
   d. Schémas et prédicats
   e. Schémas d'états
   f. Schémas d'opérations
   g. Calcul des schémas
4. Preuve de propriétés
5. Étude de cas

Plan du chapitre

58

1. Introduction
2. Abstraction des données
3. Schémas
   a. Définition
   b. Schémas et types
   c. Schémas et déclarations
   d. Schémas et prédicats
   e. Schémas d'états
   f. Schémas d'opérations
   g. Calcul des schémas
4. Preuve de propriétés
5. Étude de cas

Schémas

59

a. Définition

- Z comprend deux langages: langage mathématique + langage des schémas
- Schéma = Ensembles de déclarations de variables locales et de prédicats portant sur ces variables
  - **On encapsule des informations**
  - **On nomme les schémas pour réutilisation**
- Un schéma peut représenter:
  - Un état du système ⇒Partie statique du système
  - Une opération ⇒ Partie dynamique du système

Schémas

60

a. Définition

- Un schéma est défini par:
  - Un **nom** unique qui déclare une variable dont la portée est globale
  - Des **types** comme paramètres comme dans les déclarations génériques
  - Des déclarations de **variables locales** au schéma
  - Des **prédicats** pour exprimer les **invariants**

Schémas

61

a. Définition

- **Syntaxe**:
  - **Format vertical**

Nom_Schéma [types paramètres]
Déclarations
Prédicat

- **Exemples**

PremierSchéma
$a : \mathbb{Z}$ $b : \mathbb{P}\,\mathbb{Z}$

DeuxièmeSchéma
$a : \mathbb{Z}$ $c : \mathbb{P}\,\mathbb{Z}$
$a \in c \wedge c \neq \emptyset$

  - **Format horizontal**

Nom_Schéma[types paramètres]≙ [Déclarations | Prédicat]

Schémas

62

a. Définition

- **Exemple**: Modélisation d'une classe d'étudiants. On assume le type de base [ETUDIANT]
  - **Format vertical**

Classe
$effectif\_max : \mathbb{N}_1$ $élèves : \mathbb{P}$ ETUDIANT
$\#élèves \leq effectif\_max$

  - **Format horizontal**

$Classe \triangleq [effectif\_max : \mathbb{N}_1 ; élèves : \mathbb{P}\ \text{ETUDIANT} \mid \#élèves \leq effectif\_max]$

Schémas

63

a. Définition

- **Exemple**: Schéma générique
  - **Format vertical**

DeuxièmeSchémaGénérique[*X*]

$a : X$
$c : \mathbb{P}\, X$

$a \in c \land c \neq \emptyset$

Schémas

64

a. Définition

- **Exemple**: Une salle de de spectacle utilise un système pour gérer les réservations pour les différentes représentations. Dans la salle se trouve un nombre de sièges, dont certains ou tous peuvent êtres disponibles aux spectateurs. Le guichet doit enregistrer quel siège est vendu à quel spectateur
- On commence par les types de base [*SIEGE*, *SPECTATEUR*]
- Fonction $Vendus \in SIEGE \pfun SPECTATEUR$
- Afin de permettre l'ajout ou la suppression de sièges de la salle, on introduit un ensemble *Places* qui représente les sièges alloués à une représentation
- Il ne serait pas possible de réserver un siège qui ne fait pas partie des sièges alloués à la représentation, d'où la condition $dom\ Vendus \subseteq Places$ doit toujours être vraie

Schémas

65

a. Définition

- **Exemple**: On peut encapsuler les déclarations et les propriétés. On modélise un schémas ***Guichet***

Guichet

$$Places : \mathbb{P}\ SIEGE$$
$$Vendus : SIEGE \pfun SPECTATEUR$$
$$dom\ Vendus \subseteq Places$$

Schémas

66

a. Définition

- Utilisation des schémas:
  - les types (comme une structure de données)
  - les déclarations pour la définition d'autres types
  - les prédicats

Plan du chapitre

67

1. Introduction
2. Abstraction des données
3. Schémas
   a. Définition
   b. Schémas et types
   c. Schémas et déclarations
   d. Schémas et prédicats
   e. Schémas d'états
   f. Schémas d'opérations
   g. Calcul des schémas
4. Preuve de propriétés
5. Étude de cas

Schémas

68

b. Schémas et types

- le schéma s'ajoute aux autres formes de déclarations de types (Types des base, types libres, produit cartésien, ensemble puissance)
- Un schéma diffère d'un produit cartésien: un élément du type est référencé par son nom et non sa position
- **Exemple**:

cl: *Classe*

g: *Guichet*

⇒ pour accéder à une variable locale d'un schéma

cl.élèves ou *élèves(cl)*

g.Place ou *Place(g)*

Cette opération s'appelle **projection**

Schémas

69

b. Schémas et types

- Deux schémas sont **équivalents** s'ils ont les mêmes variables et mêmes contraintes sur ces variables:

Classe

$effectif\_max : \mathbb{N}_1$
$élèves : \mathbb{P}$ ETUDIANT

$\#élèves \leq effectif\_max$

Guichet

$Places : \mathbb{P}\ SIEGE$
$Vendu : SIEGE \pfun SPECTATEUR$

$dom\ Vendus \subseteq Places$

$effectif\_max : \mathbb{Z}$
$élèves : \mathbb{P}$ ETUDIANT

$effectif\_max \geq 1$
$\#élèves \leq effectif\_max$

$Places : \mathbb{P}\ SIEGE$
$Vendu : SIEGE \rel SPECTATEUR$

$dom\ Vendus \subseteq Places$
$Vendus \in SIEGE \pfun SPECTATEUR$

Schémas

70

b. Schémas et types

- Les instances d'un type schéma sont appelées **liens** ou **liaisons** (*bindings*)
- Affectation de valeurs aux variables du schéma qui vérifie la partie prédicative

  $\Rightarrow$ Définition d'un objet en extension
- On note une affection : $\fatsemi$ ou $\mapsto$
- **Exemple**:

PremierSchema

$a : \mathbb{Z}$
$b : \mathbb{P}\ \mathbb{Z}$

Une liaison possible: $\langle a \fatsemi 2 , b \fatsemi \{1,2,3\} \rangle$ ou : $\langle a \mapsto 2 , b \mapsto \{1,2,3\} \rangle$

Schémas

71

b. Schémas et types

- On note l'ensemble des liaisons possibles d'un schéma S: $\ominus S$
- Un schéma S peut être vue comme l'ensemble de ses liaisons $\{S: \ominus S\}$

$$\begin{array}{|l} effectif: Classe \to \mathbb{Z} \\ \hline \forall Classe \bullet effectif(\ominus Classe) = \#élèves \end{array}$$

est un raccourci de la déclaration:

$$\begin{array}{|l} effectif: Classe \to \mathbb{Z} \\ \hline \forall c: Classe \bullet effectif(c) = \#c.élèves \end{array}$$

Plan du chapitre

72

Schémas

c. Schémas et déclarations

- Un schéma peut être utilisé là où une déclaration est attendue
- **Exemple**:

$NAT\_PAIR \triangleq \{Positif \bullet 2 * x\}$

$NAT\_PAIR \triangleq \{x : \mathbb{Z} \mid x \geq 0 \bullet 2 * x\}$

$NAT\_PAIR \triangleq \{n : Positif \bullet 2 * n.x\}$

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

Schémas

c. Schémas et déclarations

- Un schéma peut être inclus dans un autre schéma

Schéma1
Déclarations1
Prédicat1

Schéma2
Schéma1 Déclarations2
Prédicat2

=

Schéma2
Déclaration1 Déclarations2
Prédicat1 ∧ Prédicat2

Fusion des déclarations + Conjonction des prédicats →

Schémas

75

c. Schémas et déclarations

- Exemple d'inclusion

Etudiant

nom : seq CHAR
num: $\mathbb{N}_1$

$\# nom \leq 15$

EtudiantAdmin

Etudiant
$adresse$: seq CHAR
$âge$: $\mathbb{N}_1$

$âge \geq 18 \wedge âge \leq 85$

EtudiantAdmin

nom : seq CHAR
num : $\mathbb{N}_1$
$adresse$: seq CHAR
$âge$: $\mathbb{N}_1$

$\#nom \leq 15 \wedge âge \geq 18 \wedge âge \leq 85$

Schémas

76

c. Schémas et déclarations

- On peut utiliser les schémas dans la partie déclarative d'une expression quantifiée
- **Exemple**:

DeuxièmeSchéma

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

$a \in c \wedge c \neq \emptyset$

$\exists$ *DeuxièmeSchéma* • $a = 0$

est équivalent à

$\exists$ s: *DeuxièmeSchéma* • $s.a = 0$

est équivalent à

$\exists\, a : \mathbb{Z} ; c : \mathbb{P}\,\mathbb{Z} \mid a \in c \wedge c \neq \emptyset \bullet a = 0$

Plan du chapitre

77

1. Introduction
2. Abstraction des données
3. Schémas
   a. Définition
   b. Schémas et types
   c. Schémas et déclarations
   d. Schémas et prédicats
   e. Schémas d'états
   f. Schémas d'opérations
   g. Calcul des schémas
4. Preuve de propriétés
5. Étude de cas

Schémas

78

d. Schémas et prédicats

- Un schéma peut servir de prédicat, on ne garde que la partie prédicative du schéma prédicat
- **Exemple**:

DeuxièmeSchéma

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

$a \in c \land c \neq \emptyset$

TroisièmeSchéma

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

$a \in c \land c \neq \emptyset$
$c \subseteq \{0,1\}$

$$\forall a : \mathbb{Z}; c : \mathbb{P}\,\mathbb{Z} \mid \textit{TroisièmeSchéma} \bullet \textit{DeuxièmeSchéma}$$

est équivalent à

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

Schémas

d. Schémas et prédicats

- **Remarque**: La partie déclarative d'un schéma peut inclure des contraintes
- **Exemple**: Ces deux schémas paraissent équivalents

DeuxièmeSchéma

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

$a \in c \land c \neq \emptyset$

QuatrièmeSchéma

$a : \mathbb{N}$
$c : \mathbb{P}\,\mathbb{N}$

$a \in c \land c \neq \emptyset$

Il faudra **normaliser** le schéma *QuatrièmeSchéma.*
Normalisation: réduction de la partie déclarative à sa forme unique et canonique

QuatrièmeSchémaNormalisé

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

$a \in \mathbb{N}$
$c \in \mathbb{P}\,\mathbb{N}$
$a \in c \land c \neq \emptyset$

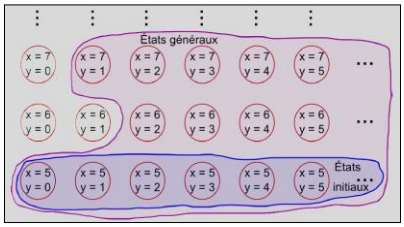
Plan du chapitre

Schémas

81

e. Schémas d'états

- Les schémas d'états sont de deux types:
  - Schémas représentant l'état général du système: décrivent les états possibles du système grossièrement
  - Schémas représentant l'état initial du système: décrivent les états initiaux possibles du système



Vue statique du système

Schémas

82

e. Schémas d'états

- L'état général du guichet est ce qui caractérise le guichet à tout moment :

Guichet

$Places : \mathbb{P}\ SIEGE$
$Vendus : SIEGE \nrightarrow SPECTATEUR$

$dom\ Vendus \subseteq Places$

- On doit indiquer tous les invariants possibles (prédicats qui doivent être vrais à tout moment de l'exécution).
- On définit un état initial du guichet comme un schéma qui initialise les différentes variables:

GuichetInit

$Guichet$

$Places = Allocation\_Initiale$
$Vendus = \emptyset$

On suppose que pour un spectacle donné, les sièges alloués initialement sont déclarés par la variable globale:
$Allocation\_Initiale : \mathbb{P}\ Siège$

Schémas

e. Schémas d'états

1. Est-ce que l'état initial est tel que $Vendus$ satisfait obligatoirement la propriété :
$$dom\ Vendus \subseteq Places$$
Oui, car cette propriété est satisfaite pour tous les états possibles du système. Elle l'est donc en particulier pour l'état initial.

2. Est-ce que $Vendus$ satisfait obligatoirement la propriété: $dom\ Vendus \subseteq Places$
quel que soit l'état du système ? **Oui, car cette propriété est décrite dans un schéma de l'état général du système.**

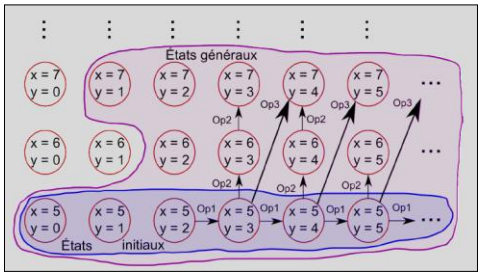
3. Est-ce que $Vendus$ satisfait obligatoirement la propriété: $Vendus = \emptyset$ quel que soit l'état du système ? **Non, car cette propriété n'est pas décrite dans un schéma de l'état général du système. Cependant, pour l'état initial, la propriété est satisfaite.**

Plan du chapitre

Schémas

f. Schémas d'opérations

- Ils permettent de modifier l'état du système (des variables définies dans les schémas d'états).
- En partant des états initiaux possibles, les schémas d'opérations définissent l'ensemble des états accessibles par le système (toujours limités par les invariants).



Schémas

f. Schémas d'opérations: Décoration de variables

- Les opérations dans le langage Z expriment les relations entre l'état du système avant l'opération et l'état après l'opération.
- Le langage Z identifie tout paramètre d'une opération comme une entrée ou comme une sortie.
- Pour exprimer aisément ces relations, Z utilise les **décorations** qui sont des marques de ponctuation (**'**, **?**, **!**) sur les variables dans les schémas d'opérations
- Le sens de la décoration :
  - x : la variable dans son état avant l'opération ;
  - x**'** : la variable dans son état après l'opération ;
  - x**?** : une variable d'entrée du schéma ;
  - x**!** : une variable de sortie du schéma.

Schémas

87

f. Schémas d'opérations: Notation Δ

- Le schéma **S'** est le même schéma que **S** où toutes les variables déclarées sont décorées par **' (primées)** dans tout le schéma.

Guichet'

$Places' : \mathbb{P}\ SIEGE$
$Vendus' : SIEGE \twoheadrightarrow SPECTATEUR$

$dom\ Vendus' \subseteq Places'$

- La notation **ΔS** est un raccourci d'écriture pour définir l'inclusion des schémas **S** et **S'**

ΔS

S
S'

Schémas

88

f. Schémas d'opérations: Notation Δ

Pour acheter une place de spectacle, nous avons besoin du nom du spectateur et le siège qui lui est réservé. La notation **Δ**S indique que le système peut être modifié par l'opération

AchatPlace

$\Delta Guichet$
$s? : SIEGE$
$c? : SPECTATEUR$

$s? \in Places \setminus dom\ Vendus$
$Vendus' = Vendus \cup \{s? \mapsto c?\}$
$Places' = Places$

Il faut indiquer la nouvelle valeur de TOUTES les variables du système, même de celles qui ne changent pas. Ne pas indiquer de nouvelle valeur correspond à indiquer que la valeur n'est pas importante, ce qui est rarement le cas quand on définit une opération.

Schémas

f. Schémas d'opérations: Notation Ξ

- La notation ΞS indique que l'opération ne doit pas changer l'état (par exemple si le but est de lire une partie de cet état)

ΞS

ΔS

ΘS = ΘS'

Disponibilité

Ξ *Guichet*
Disponible!: ℕ

Disponible! = *libre* Θ *Guichet*

- *libre* est une fonction définie comme suit:

$$libre \triangleq (\lambda\, Guichet \bullet \#(Places \setminus \mathrm{dom}\, Vendus))$$

Plan du chapitre

Schémas

91

g. Calcul des schémas

- **Calcul des schémas** : une algèbre des schémas avec des opérateurs

- Le langage Z propose des opérateurs spécifiques pour manipuler les schémas
  - Opérateurs logiques: ¬ ; ∧ ; ∨ ; ⇒ ; ⇔
  - Opérateurs de masquage: ∀ ; ∃ ; ⌉ ; \
  - Calcul de pré-condition: *pre*
  - Opérateurs de composition: ⨾ ; ≫

- **Problème de cohérence**: **lorsque deux schémas comportent des déclarations de variables communes, ces variables doivent avoir le même type pour qu'une fusion soit possible**

Schémas

92

g. Calcul des schémas

- Soient S et T deux schémas quelconques, P un prédicat et D des déclarations

Notation	Sens	Commentaire, Définition, Propriété
tuple S	Déclaration	La partie déclarative de S
pred S	Prédicat	La partie prédicative de S
$S \mid P$	Ajout de prédicat	[tuple S | **pred $S \wedge P$**]
$S; D$	Déclarations jointes	[**tuple $S; D$** | pred S]
$S[new_1/old_1, ..., new_n/old_n]$	Le nouveau schémas est S tel que les variables $old_1, ..., old_n$ sont renommées resp. en $new_1,..., new_n$	
$\neg S$	Négation	[tuple S | **¬pred S**]
$S \wedge T$	Produit	[**tuple S; tuple T** | **pred $S \wedge$ pred T**]
$S \vee T$	Union	[**tuple S; tuple T** | **pred $S \vee$ pred T**]

Schémas

93

g. Calcul des schémas

- Soient S et T deux schémas quelconques

Notation	Sens	Commentaire, Définition, Propriété
$S \Rightarrow T$	Implication	[tuple S; tuple T | pred $S \Rightarrow$ pred T]
$S \Leftrightarrow T$	Équivalence	[tuple S; tuple T | pred $S \Leftrightarrow$ pred T]
$S \setminus (v_1, v_2, \ldots, v_n)$	Masquage	Déclarations de S sans celles des variables concernées qui doivent être liées par ∃ dans $pred\ S$
$S \rceil (v_1, v_2, \ldots, v_n)$	Projection	Inverse du masquage (uniquement les variables nommées)
$pre\ S$	Pré-condition	Retourne la précondition d'un schéma opération
$S \fatsemi T$	Composition	Composition séquentielle de schémas
$S \gg T$	Tubage	Les sorties de S deviennent les entrées de T

Schémas

94

g. Calcul des schémas (Exemples)

- **Ajout de prédicat**: $S \mid P$

EtudiantAdmin

nom : seq CHAR
num : $\mathbb{N}_1$
$adresse$: seq CHAR
$âge$: $\mathbb{N}_1$

$\#nom \leq 15 \wedge âge \geq 10 \wedge âge \leq 85$

$EtudiantAdmin \mid âge \geq 80$ ⇒

EtudiantAdminA

nom : seq CHAR
num : $\mathbb{N}_1$
$adresse$: seq CHAR
$âge$: $\mathbb{N}_1$

$\#nom \leq 15 \wedge âge \geq 80 \wedge âge \leq 85$

Schémas

g. Calcul des schémas (Exemples)

- **Déclaration jointes**: S; D

EtudiantAdmin

nom : seq CHAR
num : $\mathbb{N}_1$
$adresse$: seq CHAR
$âge$: $\mathbb{N}_1$

$\#nom \leq 15 \land âge \geq 10 \land âge \leq 85$

EtudiantAdmin ; *tel* : seq $\mathbb{N}$ ⇒

EtudiantAdminTel

nom : seq CHAR
num : $\mathbb{N}_1$
$adresse$: seq CHAR
$âge$: $\mathbb{N}_1$
tel : seq $\mathbb{N}$

$\#nom \leq 15 \land âge \geq 10 \land âge \leq 85$

Schémas

g. Calcul des schémas (Exemples)

- **Negation**: $\neg S$

Gimel

$y: \mathbb{Z}$
$z: 1..10$

$y = z * z$

Avant d'appliquer la négation, il faut d'abord normaliser le schéma (expliciter la contribution de la déclaration sur la partie prédicative)

Gimel

$y, z: \mathbb{Z}$

$1 \leq z \leq 10 \land$
$y = z * z$

Négation ¬ *Gimel* ⇒

$y, z: \mathbb{Z}$

$z < 1 \lor z > 10 \lor$
$y \neq z * z$

Schémas

97

g. Calcul des schémas (Exemples)

- **Produit**:

Faire très attention aux déclarations partagées sinon ça mène à des incohérences

Etudiant

nom : seq CHAR
num: $\mathbb{N}_1$

$\# nom \leq 15$

Adresse

num_rue : $\mathbb{N}_1$
rue : seq CHAR
$code\_postal$: seq $\mathbb{N}_1$
$ville$: seq CHAR
$pays$: seq CHAR

$\# code\_postal = 5 \wedge \#rue \leq 30 \wedge \#ville \leq 30 \wedge \#pays \leq 30$

Etudiant ∧ *Adresse*

nom : seq CHAR
num : $\mathbb{N}_1$
num_rue : $\mathbb{N}_1$
rue : seq CHAR
$code\_postal$: seq $\mathbb{N}_1$
$ville$: seq CHAR
$pays$: seq CHAR

$\#nom \leq 15$
$\# code\_postal = 5 \wedge \#rue \leq 30 \wedge \#ville \leq 30 \wedge \#pays \leq 30$

Schémas

98

g. Calcul des schémas (Exemples)

- **Produit**: On peut ajouter une réponse au client pour l'opération d'achat. La réponse est déclarée comme type libre: $Réponse ::= positive \mid négative$. Un achat avec succès serait représenté par un schéma produit $AchatPlace \wedge Succès$

AchatPlace

$\Delta Guichet$
$s?$: $SIEGE$
$c?$: $SPECTATEUR$

$s? \in Places \setminus dom\ Vendus$
$Vendus' = Vendus \cup \{s? \mapsto c?\}$
$Places' = Places$

Succès

$r!$: $Réponse$

$r! = positive$

AchatPlace ∧ *Succès*

$\Delta Guichet$
$s?$: $SIEGE$
$c?$: $SPECTATEUR$
$r!$: $Réponse$

$s? \in Places \setminus dom\ Vendus$
$Vendus' = Vendus \cup \{s? \mapsto c?\}$
$Places' = Places$
$r! = positive$

Schémas

99

g. Calcul des schémas (Exemples)

- **Union $S \vee T$**: Si un spectateur demande une place non disponible, le schéma *AchatPlace* ne spécifie pas ce qui se passe dans ce cas

NonDisponible

$\Xi Guichet$
$s?: SIEGE$

$s? \notin Places \setminus dom\ Vendus$

Échec

$r! : Réponse$

$r! = négative$

- On peut alors définir l'opération achat d'une manière totale:

$$Achat \triangleq (AchatPlace \wedge Succès) \vee (NonDisponible \wedge Échec)$$

Schémas

100

g. Calcul des schémas (Exemples)

- **Masquage**: $S \setminus (v_1, v_2, \ldots, v_n)$

Etudiant

nom : seq CHAR
num: $\mathbb{N}_1$

$\# nom \leq 15$

$S_0 = Etudiant \setminus (nom)$

S_0

num: $\mathbb{N}_1$

$(\exists nom$: seq CHAR $\bullet \# nom \leq 15)$

Schémas

101

g. Calcul des schémas (Exemples)

- **Projection**: $S \rceil (v_1, v_2, \ldots, v_n)$

Etudiant
nom : seq CHAR num: $\mathbb{N}_1$
$\# nom \leq 15$

$S_1 = Etudiant \rceil (nom)$

S_1
nom : seq CHAR
$(\exists num: \mathbb{N}_1 \bullet \# nom \leq 15)$

Simplification

S_1
nom : seq CHAR
$\# nom \leq 15$

Schémas

102

g. Calcul des schémas (Exemples)

- **Quantificateurs comme opérateurs de masquage**: Soit D une déclaration, P un prédicat et S un schéma

- $\forall D \mid P \bullet S$: s'étend en un nouveau schéma
- $\exists D \mid P \bullet S$: s'étend en un nouveau schéma

Expansion du quantificateur dans le schéma

- Toutes les variables introduites dans D doivent être déclarées dans S et avoir le même type
- Le schéma résultant contient les mêmes variables que S sauf ceux introduits par D
- Ces variables masquées sont quantifiées dans le prédicat de S

S
a: A b: B
P

$\forall b: B \bullet S$ **s'étend** en le schéma

a: A
$\forall b$: B $\bullet P$

$\exists b: B \bullet S$ **s'étend** en le schéma

a: A
$\exists b$: B $\bullet P$

Schémas

103

g. Calcul des schémas (Exemples)

- **Quantificateurs comme opérateurs de masquage**: Exemple

Gimel

$y : \mathbb{Z}$
$z : 1..10$

$y = z * z$

Avant d'appliquer le quantificateur, il faut d'abord normaliser

$\forall z : \mathbb{Z} \mid z > 5 \bullet$ *Gimel*

$y : \mathbb{Z}$

$\forall z : \mathbb{Z} \mid z > 5 \bullet z \in 1..10 \land y = z * z$

Schémas

104

g. Calcul des schémas (Exemples)

- **Calcul de pré-condition**: c'est un schéma qui contient uniquement l'état avant l'opération et les paramètres d'entrée ⇒ masquage de l'état après l'opération et des paramètres de sortie $pre\ Op \triangleq (\exists Etat' ; y! : Y \bullet Op)$

Op

Etat (l'état avant l'opération)
Etat' (l'état après l'opération)
y! : Y (paramètre de sortie)
...

...

- L'expression retourne les conditions sur l'état de départ pour que l'état final existe. S'il n'y a pas de pré-condition, l'expression sera évaluée à vrai (true).

Schémas

g. Calcul des schémas (Exemples)

- **Calcul de pré-condition**: Exemple

Compteur

$valeur, limite : \mathbb{N}$

$valeur \leq limite$

Add

$Compteur$
$Compteur'$
$saut ? : \mathbb{N}$
$nouvelle\_valeur ! : \mathbb{N}$

$valeur' = valeur + saut ?$
$limite' = limite$
$nouvelle\_valeur ! = valeur'$

pre Add

$Compteur$
$saut ? : \mathbb{N}$

$\exists Compteur'; nouvelle\_valeur ! : \mathbb{N} \bullet$
$valeur' = valeur + saut ?$
$limite' = limite$
$nouvelle\_valeur ! = valeur'$

Schémas

g. Calcul des schémas (Exemples)

- **Composition séquentielle**: La composition séquentielle de deux opérations $S \fatsemi T$ est définie comme une opération dont l'état initial est celui de S et l'état final est celui de $T \Rightarrow$ L'état final de S est l'état initial de T
- L'idée est de faire abstraction de l'état intermédiaire (on le cache)
  - Chaque variable primée de S doit lui correspondre une variable non primée de T du même nom et du même type et sera masquée
  - Toute autre variable commune doit avoir le même type dans les deux schémas
  - Le schémas $S \fatsemi T$ contient la fusion des variables des deux schémas S et T
  - Le schémas $S \fatsemi T$ contient la conjonction des prédicats des deux schémas S et T
- Formellement: $Etat''$ est l'état intermédiaire à cacher

$$\exists Etat'' \bullet (\exists Etat' \bullet [S ; Etat'' \mid \theta Etat' = \theta Etat'']) \land (\exists Etat \bullet [T ; Etat \mid \theta Etat = \theta Etat''])$$

Schémas

107

g. Calcul des schémas (Exemples)

- **Composition séquentielle**: Exemple

Compteur

$valeur, limite : \mathbb{N}$

$valeur \leq limite$

Inc

$Compteur$
$Compteur'$

$valeur' = valeur + 1$
$limite' = limite$

Add

$Compteur$
$Compteur'$
$saut ? : \mathbb{N}$
$nouvelle\_valeur ! : \mathbb{N}$

$valeur' = valeur + saut ?$
$limite' = limite$
$nouvelle\_valeur ! = valeur'$

Inc ⨾ Add

$Compteur$
$Compteur'$
$saut ? : \mathbb{N}$
$nouvelle\_valeur ! : \mathbb{N}$

$valeur' = valeur + 1 + saut ?$
$limite' = limite$
$nouvelle\_valeur ! = valeur'$

Schémas

108

g. Calcul des schémas (Exemples)

- **Composition séquentielle**: Exemple

Compteur

$valeur, limite : \mathbb{N}$

$valeur \leq limite$

Inc

$Compteur$
$Compteur'$

$valeur' = valeur + 1$
$limite' = limite$

Add

$Compteur$
$Compteur'$
$saut ? : \mathbb{N}$
$nouvelle\_valeur ! : \mathbb{N}$

$valeur' = valeur + saut ?$
$limite' = limite$
$nouvelle\_valeur ! = valeur'$

Add ⨾ Inc

$Compteur$
$Compteur'$
$saut ? : \mathbb{N}$
$nouvelle\_valeur ! : \mathbb{N}$

$valeur' = valeur + saut ? + 1$
$limite' = limite$
$nouvelle\_valeur ! = valeur + saut?$ (avant Inc)

Schémas

109

g. Calcul des schémas (Exemples)

- **Tubage ou Canalisation (Piping)**: Le tubage de deux opérations $S \gg T$ est similaire à la composition séquentielle mais porte sur les entrées et les sorties
  - Aucune correspondance n'est exigée sur les variables d'états (faire attention aux incohérences)
  - Il n'est pas nécessaire que toutes les variables de sortie du premier schéma soient mises en correspondance avec toutes les entrées du second. Toute correspondance (variable tubée) sera masquée
  - Le schémas $S \gg T$ contient la fusion des variables des deux schémas S et T après masquage
  - Le schémas $S \gg T$ contient la conjonction des prédicats des deux schémas S et T
- Formellement: *Pipe* contient uniquement les variables tubées

$$\exists Pipe!? \bullet (\exists Pipe! \bullet [S\ ;\ Pipe!?\ \mid\ \theta Pipe! = \theta\ Pipe!?]) \wedge$$
$$(\exists Pipe? \bullet [T\ ;\ Pipe?\ \mid\ \theta Pipe? = \theta\ Pipe!?])$$

Schémas

110

g. Calcul des schémas (Exemples)

- **Tubage ou Canalisation (Piping)**: Exemple

$AuCarréPuisAjouter \triangleq AuCarré \gg Add\ [y! / saut?]$

Pas de variables d'état dans cette opération

AuCarré

$x?, y!: \mathbb{N}$

$y! = x? * x?$

Add

$\Delta compteur$
$saut? : \mathbb{N}$
$nouvelle\_valeur! : \mathbb{N}$

$valeur' = valeur + saut?$
$limite' = limite$
$nouvelle\_valeur! = valeur'$

Schéma de composition

$\Delta compteur$
$x? : \mathbb{N}$
$nouvelle\_valeur! : \mathbb{N}$

$valeur' = valeur + x? * x?$
$limite' = limite$
$nouvelle\_valeur! = valeur'$

Le renommage sert à faire correspondre la variable de sortie de *Aucarré* à la variable d'entrée de *Add*

Plan du chapitre

111

1. Introduction
2. Abstraction des données
3. Schémas
4. Preuve de propriétés
   a. Démarche de spécification
      i. Organisation de la spécification
      ii. Document de spécification
   b. Validation
   c. Raffinage
5. Étude de cas

Preuve de propriétés

112

a. Démarche de spécification

- Z n'est pas une méthode
  - Z ne définit pas de démarche propre
  - Z ne possède pas d'environnement de développement
  - Utiliser une démarche générale induite par les spécifications formelles: cycle linéaire ou contractuel
  - Z convient aux systèmes séquentiels (Il y a d'autres versions de Z)

Preuve de propriétés

113

a. Démarche de spécification

- Théorie du contrat (Programmation par contrat)
  - Il s'agit d'un paradigme de programmation visant principalement à réduire le nombre de bugs dans un programme.
  - Développée dans le but d'aider les informaticiens à développer des systèmes fiables ⇒ capacité à faire le travail pour lequel il a été conçu (un programme conforme à sa spécification) ainsi que sa capacité à gérer correctement les situations exceptionnelles.
  - les composantes sont vues comme des entités communiquant entre elles. Chacune s'attend à ce que l'on communique avec elle selon des règles bien précises (Obligations). En retour, elle s'engage à exécuter certaine(s) tâche(s) sans mettre le système dans un état indésirable (Bénéfices).

  ⇒**Un contrat**

Preuve de propriétés

114

a. Démarche de spécification

- De manière générale, un système de programmation par contrat gère différents types de conditions :
  - **Pré-conditions**: conditions relatives à une méthode et qui doivent être vérifiés avant l'exécution de cette méthode.
  - **Post-conditions**: conditions devant être vérifiées après l'exécution de la méthode. Ils peuvent porter sur la valeur de retour de la méthode ou sur la valeur qu'avaient certaines variables avant l'exécution de la méthode, pour comparer l'état d'une variable de classe avant l'appel à la méthode avec son état après l'exécution de cette méthode.
  - **Invariants**: permettent de vérifier la cohérence des valeurs des variables. Ces contrats doivent être vérifiés avant et après chaque appel à une méthode, mais pas forcément au cours de l'exécution de ces méthodes

Preuve de propriétés

a. Démarche de spécification

- Théorie du contrat et Z
  - **Invariants**: propriétés qui doivent être satisfaites tout au long de la vie du (ou d'une partie du) système, de sa création à sa destruction ⇒ **Prédicats dans les schémas d'état**
  - **Pré-conditions (Obligations du contrat)**: conditions à satisfaire pour appeler une opération et s'assurer qu'elle s'exécute correctement ⇒ **Prédicats formés des variables non primées dans les schémas d'opération**
  - **Post-conditions (Bénéfices du contrat)**: propriétés à satisfaire par la sortie que retourne l'opération ⇒ **Prédicats formés des variables primées dans les schémas d'opération**

Plan du chapitre

Preuve de propriétés

117

i. Organisation de la spécification

- Une spécification Z est structurée comme suit: Récap

1. **Déclarations globales:** types de base, les types libres, constantes, variables et fonctions globales
2. **Types structurés :** définis par l'utilisateur
   - Schéma d'état regroupant des variables et un prédicat sur ces variables
   - Schéma initial est donné pour chaque schéma d'état
   - Schémas d'opérations décrivent chaque opération du type structuré, se décomposent en:
     - Un schéma du nom de l'opération
     - Un schéma de pré-condition est calculé ensuite

Preuve de propriétés

118

i. Organisation de la spécification

- Une spécification Z est structurée comme suit: Récap

3. **État du système :** défini par
   - Schéma d'état donnant la structure du système
   - Schéma initial décrivant l'état initial (ou ensemble des états initiaux)du système
4. **Opérations du système :** Schémas d'opérations décrivant chaque opération du système et se décomposent en:
   - Un schéma du nom de l'opération
   - Un schéma de pré-condition est calculé ensuite
5. Des théorèmes et des preuves enrichissent la spécification

Plan du chapitre

119

1. Introduction
2. Abstraction des données
3. Schémas
4. Preuve de propriétés
   a. Démarche de spécification
      i. Organisation de la spécification
      ii. Document de spécification
   b. Validation
   c. Raffinage
5. Étude de cas

Preuve de propriétés

120

ii. Document de spécification

- Une spécification Z peut être difficile à décoder même pour les connaisseurs ⇒ une spécification formelle ne vient jamais seule mais accompagnée de commentaires qui expliquent les grandes lignes

Une fois la spécification formelle terminée, quelle forme devrait prendre le document qui sera remis aux ingénieurs logiciels qui vont suivre dans le processus de développement du logiciel ?

pas de réponse absolue à cette question. Plusieurs façons de faire sont possibles, plusieurs d'entre elles s'équivalent. Les entreprises ont souvent des normes maison

Preuve de propriétés

ii. Document de spécification

Approche basée sur des idées développées dans les laboratoires du *IBM Hursley* et du *Oxford University Programming Research Group*

1. **Introduction**: On présente le contexte et les grandes lignes du système spécifié
2. **Description des ensembles (types) et des différentes constantes**: Dans cette section du rapport, on fournit d'abord la spécification formelle des différents types et constantes. On décrit aussi de façon textuelle chacun de ces types et constantes lorsque nécéssaire
3. **Présentation de toute théorie utile**: Par exemple, pour la spécification formelle d'une centrale nucléaire, il peut être utile de fournir quelques explications pour comprendre les détails de la spécification. Entre autres, quelques rappels de lois de physique nucléaire pourraient s'avérer utiles

Preuve de propriétés

ii. Document de spécification

4. **Description des états standards du système et des propriétés qu'ils doivent satisfaire**: On fournit ici la spécification formelle d'un état standard valide pour chacune des composantes du système avec des explications supplémentaires lorsque c'est nécessaire pour enlever toute ambiguïté.
5. **Description d'un état initial**: Dans cette partie du rapport, on fournit la spécification formelle d'un état initial valide du système. Il peut y en avoir plus d'un si l'on veut laisser une certaine liberté au programmeur. Si des indications doivent être données aux ingénieurs logiciels qui vont suivre dans la suite du processus de développement quant à l'état initial, c'est ici qu'il faut le faire
6. **Description des différentes fonctions**: On présente ici la spécification formelle des différentes fonctions du système(conditions normales d'exécution, aucune mention des cas d'erreurs ou des cas d'exceptions). Le tout doit toujours être accompagné de toute explication supplémentaire nécessaire

Preuve de propriétés

ii. Document de spécification

7. **Préconditions, postconditions et invariants de chacune des fonctions**: Pour chacune des fonctions partielles (la version qui correspond aux conditions normales d'exécution) , on présente le tableau suivant:

Fonction	*Nom de la fonction*
Entrées/Sorties	*Variables d'entrée et de sortie (nom décoration: type)*
Pré-conditions	*Prédicats*
Post-conditions	*Prédicats*
Invariants	*Prédicats*

Preuve de propriétés

ii. Document de spécification

8. **Description des fonctions totales**: On présente ici la spécification formelle de la version totale des différentes fonctions. On doit donc présenter ici comment les fonctions réagissent dans les différents cas d'erreurs ou d'exceptions. Si, pour une raison ou pour une autre, on décide de ne pas rendre totale une certaine fonction, il faut expliquer les raisons sous-jacentes.

9. **Résumé et index**: La spécification formelle d'un système peut rapidement devenir un très gros document. Il est alors utile d'avoir une liste des variables et des différents noms de schémas utilisés ainsi que le numéro des pages où ils apparaissent

Plan du chapitre

125

1. Introduction
2. Abstraction des données
3. Schémas
4. Preuve de propriétés
   a. Démarche de spécification
   b. Validation
      i. Théorème de l'initialisation
      ii. Théorème des pré-conditions
   c. Raffinage
5. Étude de cas

Preuve de propriétés

126

a. Validation

- **Rappel**:
  - Le but de la spécification formelle est de minimiser les erreurs commises dans la conception d'un logiciel le plus tôt possible dans le processus de développement
  - Les méthodes formelles proposent de travailler avec un prototype du logiciel décrit à l'aide d'outils mathématiques
  - Il faut que cette description soit très précise quant aux besoins du client et qu'elle permette d'éviter toute anomalie
  - Il faut démontrer des théorèmes qui aideront à s'assurer que le logiciel qui sera programmé va satisfaire les besoins du client et qu'il n'y aura aucune mauvaise surprise.

Preuve de propriétés

127

a. Validation

- Quels théorèmes doit on démontrer?
- On ne peut pas tout démonter, alors sur quelles propriétés va-t-on se concentrer et pourquoi?
- N'est-il pas naturel de croire que plus on démontre de propriétés, plus on minimise le nombre d'erreurs possibles lors des étapes subséquentes du développement du logiciel en question ?

Preuve de propriétés

128

a. Validation

- La validation comprend trois parties distinctes:
  - Preuve de propriétés générales: donnent un certains degré de confiance en la spécification. Deux propriétés sont importantes dans les systèmes séquentiels: cohérence et complétude
  - Preuve de propriétés attendues du système: ces propriétés sont indiquées par les analystes dans la spécification informelle. Elles sont à la base du test de spécification et de réalisation
  - Preuve de propriétés liées à la pratique de Z (Obligation de preuves), de plusieurs ordres:
    - **Théorème de l'initialisation**: Existence d'un état initial
    - **Théorème des pré-conditions**: Préservation des invariants dans les opérations
    - **Preuve de raffinage** de données et d'opérations

Plan du chapitre

129

1. Introduction
2. Abstraction des données
3. Schémas
4. Preuve de propriétés
   a. Démarche de spécification
   b. Validation
      i. Théorème de l'initialisation
      ii. Théorème des pré-conditions
   c. Raffinage
5. Étude de cas

Preuve de propriétés

130

i. Théorème de l'initialisation

- Il faut prouver que l'état initial respecte l'état et sa contrainte (prédicat), c-à-d. que l'état initial est un état valide du système
- Ce théorème s'écrit:

$$\exists\, Etat \bullet\ EtatInit$$

Il existe un état valide qui satisfait les conditions de l'état initial

- Rappelons que les schémas peuvent être utilisés dans les déclarations, on dit qu'ils sont utilisés comme texte: $Declaration \mid Predicat$
- Nous allons écrire quelquefois $[Declaration \mid Predicat]$ au lieu de $Declaration \mid Predicat$ pour mettre en évidence le fait que l'expression provient d'un schéma

Preuve de propriétés

131

i. Théorème de l'initialisation (Exemple)

- Rappelons l'exemple du guichet:

[*SIEGE, SPECTATEUR*]

| *Allocation_Initiale* : ℙ *Siège*

Guichet

Places: ℙ *SIEGE*
Vendus: *SIEGE* ⇸ *SPECTATEUR*

dom Vendus ⊆ *Places*

GuichetInit

Guichet

Places = *Allocation_Initiale*
Vendus = Ø

Preuve de propriétés

132

i. Théorème de l'initialisation (Exemple)

- Le théorème d'initialisation est: ∃ *Guichet* • *GuichetInit*
- Remplaçons le schéma *GuichetInit* par sa définition, nous aurons:

∃ *Guichet* • [*Guichet* | *Places* = *Allocation_Initiale* ∧ *Vendus* = ∅]

⇔ ⟨**Expansion de ∃**⟩

∃ *Guichet* • *Places* = *Allocation_Initiale* ∧ *Vendus* = ∅

⇔ ⟨**Définition de** *Guichet* + **Logique: Axiome de transfert**⟩

∃ *Places*: ℙ *SIEGE* •
∃ *Vendus*: *SIEGE* ⇸ *SPECTATEUR* •
dom Vendus ⊆ *Places* ∧
Places = *Allocation_Initiale* ∧
Vendus = ∅ *Nous aurons besoin à ce niveau de l'axiome du point*

Preuve de propriétés

133

i. Théorème de l'initialisation (Exemple)

- **Définition(Axiome du point)**: Il s'agit d'un axiome très utile en logique pour se débarrasser du quantificateur **∃** lorsque la valeur est suggérée par le prédicat

$$\exists x : E \bullet (p \wedge x = t) \Leftrightarrow t \in E \wedge p[t/x]$$

pourvu que x ne soit pas libre dans t

De plus, il faut s'assurer d'une application correcte de la loi de substitution dans p[t/x] : aucune variable libre de t ne peut devenir liée dans p[t/x] (dans ce cas-ci, le renommage permet d'utiliser l'axiome du point).

- **Interprétation** : si la variable liée par ∃ doit être égale à *t*, alors pour que l'expression à gauche du signe d'équivalence soit vraie, il faut et il suffit que *t* soit élément de **E** et que **p** soit vraie

Preuve de propriétés

134

i. Théorème de l'initialisation (Exemple)

- **Axiome du point : Exemples**

$$\exists x : E \bullet (p \wedge x = t) \Leftrightarrow t \in E \wedge p[t/x]$$

pourvu que x ne soit pas libre dans t

- $\exists x : \mathbb{N} \bullet x = x + 1 \wedge x > 3$ ⇨ Rien à faire
- $\exists x : \mathbb{N} \bullet x = y + 1 \wedge impair(x)$ ⇨ $y + 1 \in \mathbb{N} \wedge impair(y + 1)$
- $\exists x : \mathbb{N} \bullet x = y + 1 \wedge (\exists y : \mathbb{N} \bullet x = y)$ ⇨ Renommage de y par z dans $(\exists y:\mathbb{N} \bullet x = y)$ on aura alors $y+1 \in \mathbb{N} \wedge (\exists z:\mathbb{N} \bullet y+1 = z)$
- $\exists x : \mathbb{N} \bullet x \bmod 2 = 0 \wedge x = 3$ ⇨ 3 mod 2 =0 qui est faux

Preuve de propriétés

i. Théorème de l'initialisation (Exemple)

- Théorème d'initialisation (suite) : $\exists\ Guichet \bullet GuichetInit$

$\exists\ Places: \mathbb{P}\ SIEGE \bullet$

$\exists\ Vendus: SIEGE \twoheadrightarrow SPECTATEUR \bullet$

$dom\ Vendus \subseteq Places \land$

$Places = Allocation\_Initiale \land$

$Vendus = \emptyset$

$\Leftrightarrow$ ⟨Axiome du point (deux fois)⟩

$Allocation\_Initiale \in \mathbb{P}\ SIEGE \land \emptyset \in SIEGE \twoheadrightarrow SPECTATEUR \land$

$dom\ \emptyset \subseteq Allocation\_Initiale$

$\Leftrightarrow$ ⟨Théorie des ensembles⟩

$true$

Plan du chapitre

Preuve de propriétés

137

ii. Théorème des pré-conditions

- Il faut vérifier si les fonctions spécifiées n'envoient pas le système dans des états indésirables ⇒ vérifier qu'à partir d'un état valide, si les pré-conditions d'une fonction sont satisfaites, alors cette fonction envoie toujours le système dans un état valide

$$\forall Etat; Entrees? \mid Proprietes \bullet \textbf{pre } Operation$$

- Le théorème des pré-conditions exprime le fait que toutes les pré-conditions sont énoncées. Si on peut montrer que ce théorème est vrai, alors pour tout état de départ, pour toutes les entrées qui satisfont les pré-conditions, un état final existe

Il faut faire très attention entre le calcul des pré-conditions et le théorème des pré-conditions

Preuve de propriétés

138

ii. Théorème des pré-conditions

- Reconsidérons l'exemple du *Guichet* avec l'opération *AchatPlace*. Le théorème des pré-conditions s'énonce comme suit:

AchatPlace

$\Delta Guichet$
$s?: SIEGE$
$c?: SPECTATEUR$

$s? \in Places \setminus dom\ Vendus$
$Vendus' = Vendus \cup \{s? \mapsto c?\}$
$Places' = Places$

$$\forall Guichet; s?: SIEGE; c?: SPECTATEUR \mid s? \in Places \setminus dom\ Vendus \bullet \textbf{pre } AchatPlace$$

Preuve de propriétés

139

ii. Théorème des pré-conditions

- Après le calcul de **pre** *AchatPlace*, qui donne le schéma qu'on nomme *PreAchatPlace*, le théorème devient donc:

$\forall Guichet; s? : SIEGE ; c? : SPECTATEUR \mid s? \in Places \setminus dom\ Vendus \bullet$

PreAchatPlace

$Guichet$
$s? : SIEGE$
$c? : SPECTATEUR$

$\exists Guichet' \bullet s? \in Places \setminus dom\ Vendus$
$Vendus' = Vendus \cup \{s? \mapsto c?\}$
$Places' = Places$

Preuve de propriétés

140

ii. Théorème des pré-conditions

- le théorème devient donc:

⇔ **simplification de la pré-condition ⟨Définition de *Guichet'* ,Axiome du point, Théorie des ensemble⟩**

$\forall Guichet; s? : SIEGE ; c? : SPECTATEUR \mid s? \in Places \setminus dom\ Vendus \bullet$
$[Guichet; s? : SIEGE ; c? : SPECTATEUR \bullet s? \in Places \setminus dom\ Vendus]$

⇔ **⟨Expansion de ∀et logique⟩**

true

Preuve de propriétés

- Il existe d'autres théorèmes que nous pouvons démontrer tels que la vérification des invariants et la vérification du domaine (s'assurer que toutes les fonctions sont utilisées correctement. Par exemple, pour l'opérateur #, il faut qu'il soit appliqué sur un ensemble fini.)
- Si nous démontrons le théorème de l'initialisation et le théorème des pré-conditions pour chaque opération de la spécification (entouré par la vérification de domaine), alors nous obtenons automatiquement que :

Le système ne se retrouvera jamais dans un état indésirable (non contrôlé).

Preuve de propriétés

- Démontrons que Le système ne se retrouvera jamais dans un état indésirable (non contrôlé).

Démonstration (par induction mathématique) :

- Cas de base : À l'état initial, le système est dans un état valide puisque le théorème de l'initialisation a été démontré.
- Cas d'induction : Supposons que le système soit dans un état valide. Il faut montrer que peu importe l'état dans lequel il se retrouvera ensuite, celui-ci doit être aussi valide. (Rappelons que le système ne peut changer d'état qu'à travers ses opérations.) . Puisque:
  1. le théorème des pré-conditions pour toutes les opérations du système a été démontré et
  2. le système est dans un état valide par hypothèse,

  ⇒Ainsi l'état après une opération est obligatoirement valide.

Plan du chapitre

1. Introduction
2. Abstraction des données
3. Schémas
4. Preuve de propriétés
   a. Démarche de spécification
   b. Validation
   c. Raffinage
      i. Raffinage des opérations
      ii. Raffinage des données
5. Étude de cas

Preuve de propriétés

a. Raffinage

- Z permet de spécifier à des niveaux plus ou moins proches des langages de programmation
- Raffinage = Passage d'un niveau d'abstraction à un niveau plus concret
- On distingue deux types de raffinage en Z:
  - Raffinage des opérations
  - Raffinage des données

Preuve de propriétés

a. Raffinage

- Avantages des méthodes formelles:
  - Raffinage formalisée par une relation qui doit être prouvée
  - Obtenir un programme qui respecte la spécification
  - Réduire le non-déterminisme (Raffiner un type de base par un type schéma, réduire le domaine de valeur

Plan du chapitre

Preuve de propriétés

147

i. Raffinage des opérations

- Les structures de contrôle du langage de programmation cible sont introduites progressivement (séquences, conditionnelles,, itératives)
- Règle de raffinage: Si une opération concrète Cop est un raffinage d'une opération abstraite Aop ayant le même espace d'état et comme entrée x?: X et comme sortie y!:Y, alors les deux opérations peuvent différer sur deux cas:
  1. La pré-condition de Cop est plus libérale que Aop: Cop est assurée de terminer dans plus d'états que Aop
  2. Cop est plus déterministe que Aop: pour certains états avant l'opération, le nombre d'états après est plus petit

Plan du chapitre

148

Preuve de propriétés

ii. Raffinage des données

- Les structures de données du langage de programmation cible son introduites progressivement (structures, tableaux, pointeurs)