# Développement de logiciel avec Z

## Survol et exemples

Pascal ANDRE

MIAGE
Université de Nantes

Master Miage M1

# Plan

# Contexte de l'intervention

Le développement du logiciel

1. Avec rigueur
   - Modèles, produits formels
   - Processus dérivatif
   - Validation, vérification
2. Qualité induite pour la sûreté de fonctionnement

## Focus

- approche formelle (voir la présentation `introMF`)
- notation Z et outils de vérification (Z-EVES, ...)

  Vers un usage rationnel
  - bonnes pratiques de spécification (précision)
  - bonnes pratiques de conception (assertions, preuves)
  - bonnes pratiques de programmation (tests)
  - bonnes pratiques du processus (rigueur)

# La notation Z

- Modélisation mathématique

# La notation Z

- Modélisation mathématique
- Abrial (Grenoble, Oxford)

# La notation Z

- Modélisation mathématique
- Abrial (Grenoble, Oxford)
- Années 70

# La notation Z

- Modélisation mathématique
- Abrial (Grenoble, Oxford)
- Années 70
- Principes

# La notation Z

- Modélisation mathématique
- Abrial (Grenoble, Oxford)
- Années 70
- Principes
  - spécification mathématique, axiomatique de Hoare

# La notation Z

- Modélisation mathématique
- Abrial (Grenoble, Oxford)
- Années 70
- Principes
  - spécification mathématique, axiomatique de Hoare
  - abstraction

# La notation Z

- Modélisation mathématique
- Abrial (Grenoble, Oxford)
- Années 70
- Principes
  - spécification mathématique, axiomatique de Hoare
  - abstraction
  - schéma

# La notation Z

- Modélisation mathématique
- Abrial (Grenoble, Oxford)
- Années 70
- Principes
  - spécification mathématique, axiomatique de Hoare
  - abstraction
  - schéma
  - preuve de propriétés

# La notation Z

- Modélisation mathématique
- Abrial (Grenoble, Oxford)
- Années 70
- Principes
  - spécification mathématique, axiomatique de Hoare
  - abstraction
  - schéma
  - preuve de propriétés
  - raffinement (raffinage, réification)

# La notation Z

- Modélisation mathématique
- Abrial (Grenoble, Oxford)
- Années 70
- Principes
  - spécification mathématique, axiomatique de Hoare
  - abstraction
  - schéma
  - preuve de propriétés
  - raffinement (raffinage, réification)
- VDM, B

# Quelques références autour de Z

Quelques références autour de Z

- Notation Z [Spi94, WL88]
- Méthodes formelles et Z [AV01, AV04]
- Développer avec Z [PST96, Bow96, Jac97, Wor92]


Pascal André and Alain Vailly.
Spécification des logiciels ; Deux exemples de pratiques récentes : Z et UML, volume 2 of Collection Technosup.
Editions Ellipses, 2001.
ISBN 2-7298-0774-8.


Pascal André and Alain Vailly.
Exercices corrigés en langage Z ; Les spécifications formelles par la pratique, volume 4 of Collection Technosup.
Editions Ellipses, 2004.
ISBN 2-7298-1942-8.


Jonathan Bowen.
Formal Specification & Documentation using Z: A Case Study Approach.
International Thomson Publishing, 1996.
ISBN 1-85032-230-9.


Jonathan Jacky.
The Way of Z: Practical Programming with Formal Methods.
Cambridge University Press, 1997.
ISBN 0-521-55041-6.


Ben Potter, Jane Sinclair, and David Till.
Introduction to Formal Specification and Z.

International Series in Computer Science. Prentice-Hall, 2 edition, 1996.
ISBN 0-13-242207-7.

Mike Spivey.
La notation Z.
Collection Méthodologies du logiciel. Editions Masson, 1994.
Traduit de l'anglais par Michel Lemoine, ISBN 2-225-84367-8.

J. Woodcock and M. Loomes.
Software Engineering Mathematics.
Addison Wesley, 1988.

John B. Wordsworth.
Software development with Z.
Addison Wesley, 1992.
ISBN 0-201-62757-4.

# Plan

# La logique

- Logique des propositions

# La logique

- Logique des propositions

*"Loin des yeux, loin du cœur.*
*Mieux vaut tard que jamais.*
*Le monde appartient à celui qui se lève tôt.*
*La raison du plus fort est toujours la meilleure.*
*Gérard est le plus fort.*
*Gérard se lève tard et vit la nuit.*
*Après la pluie, le beau temps."*

# La logique

- Logique des propositions

  *"Loin des yeux, loin du cœur.*
  *Mieux vaut tard que jamais.*
  *Le monde appartient à celui qui se lève tôt.*
  *La raison du plus fort est toujours la meilleure.*
  *Gérard est le plus fort.*
  *Gérard se lève tard et vit la nuit.*
  *Après la pluie, le beau temps."*

  - exclusion : vrai ou faux
  - contradiction : non (vrai et faux)
  - combinaison : et, ou, non, ou exclusif, implique, etc.

  cohérence ? *Gérard*

# La logique

- Logique des propositions
- Logique des prédicats

# La logique

- Logique des propositions
- Logique des prédicats
  - variables : paramétrer les propositions
  - quantificateur : décrire le domaine de la variable (le domaine sera le type en Z)
  - portée : espace d'existence de la variable.
  - combinaison : opérateurs des propositions.

# La logique

- Logique des propositions
- Logique des prédicats

| | |
|---|---|
| $impératif(x)$ | la variable $x$ est libre |
| $impératif(Pascal)$ | est une proposition |
| $fonctionnel(caml)$ | est une proposition |
| $objet(C++)$ | est une proposition |
| $\forall x \bullet impératif(x) \Rightarrow langage(x)$ | $x$ est liée |
| $impératif(C) \Rightarrow impératif(C++)$ | $C$ est une constante |
| $\forall x \bullet objet(x) \Rightarrow$ | |
| $fonctionnel(x) \vee impératif(x)$ | $x$ est liée |

# La logique des propositions

- Constantes : *true* et *false*

# La logique des propositions

- Constantes : *true* et *false*
- Opérateurs :

| **Opérateur** | **Sens** | **Notations alternatives** |
|---|---|---|
| $\neg P$ | négation | *not* $P$, $\sim P$, $\overline{P}$ |
| $P \wedge Q$ | conjonction | $P$ *and* $Q$, $P.Q$, $P\&Q$ |
| $P \vee Q$ | disjonction | $P$ *or* $Q$, $P + Q$ |
| $P \Rightarrow Q$ | négation | $P$ *implies* $Q$, *if* $P$ *then* $Q$ |
| $P \Leftrightarrow Q$ | équivalence | équivalence logique |

A ces opérateurs peuvent être ajoutés de nouveaux opérateurs tels que le *ou exclusif* ($P$ *xor* $Q$).

# La logique des propositions

Propriétés :

- Axiomes

# La logique des propositions

Propriétés :

- Axiomes
- Théorèmes

| Nom | Propriété | Proposition |
|---|---|---|
| $P_1$ | Commutativité | $P \wedge Q \Leftrightarrow Q \wedge P$ |
| $P_2$ | | $P \vee Q \Leftrightarrow Q \vee P$ |
| $P_7$ | Lois DE MORGAN (formes normales) | $\neg (P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$ |
| $P_8$ | | $\neg (P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$ |
| $P_9$ | Forme normale | $P \Rightarrow Q \Leftrightarrow \neg P \vee Q$ |

# La logique des propositions

Propriétés :

- Axiomes
- Théorèmes
- Preuve $\Rightarrow$ raisonnement par

# La logique des propositions

Propriétés :

- Axiomes
- Théorèmes
- Preuve $\Rightarrow$ raisonnement par
  - réduction

# La logique des propositions

Propriétés :

- Axiomes
- Théorèmes
- Preuve $\Rightarrow$ raisonnement par
  - réduction
  - déduction

# La logique des propositions

Propriétés :

- Axiomes
- Théorèmes
- Preuve $\Rightarrow$ raisonnement par
  - réduction
  - déduction
  - l'absurde

# La logique des propositions

Propriétés :

- Axiomes
- Théorèmes
- Preuve $\Rightarrow$ raisonnement par
  - réduction
  - déduction
  - l'absurde
  - cas

# La logique des propositions

Propriétés :

- Axiomes
- Théorèmes
- Preuve $\Rightarrow$ raisonnement par
  - réduction
  - déduction
  - l'absurde
  - cas
  - hypothèse

# La logique des propositions

Propriétés :

- Axiomes
- Théorèmes
- Preuve $\Rightarrow$ raisonnement par
  - réduction
  - déduction
  - l'absurde
  - cas
  - hypothèse
  - récurrence

# La logique des propositions

Propriétés :

- Axiomes
- Théorèmes
- Preuve $\Rightarrow$ raisonnement par
  - réduction
  - déduction
  - l'absurde
  - cas
  - hypothèse
  - récurrence
  - autres (tables de vérité, Karnaugh, résolution)

# La logique des propositions

Propriétés :

- Axiomes
- Théorèmes
- Preuve $\Rightarrow$ raisonnement par
  - réduction
  - déduction
  - l'absurde
  - cas
  - hypothèse
  - récurrence
  - autres (tables de vérité, Karnaugh, résolution)

++ Détails : [AV01], chapitre 2, section 2

# La logique des prédicats

- Constantes : *true* et *false*

# La logique des prédicats

- Constantes : *true* et *false*
- Quantificateurs :

variable libre

| | |
|---|---|
| $P(x)$ | prédicat paramétré par la variable $x$ |

variable liée

| | |
|---|---|
| $\forall x \bullet P(x)$ | P est vrai pour toute valeur de $x$ |
| $\exists x \bullet P(x)$ | il y a au moins une valeur de $x$ telle que P soit vrai |
| $\exists ! x \bullet P(x)$<br>$\exists_1 x \bullet P(x)$ | il y a une seule valeur de $x$ telle que P soit vrai |

# La logique des prédicats

Interprétation :

- Domaine : défini ou pas (cf types)

# La logique des prédicats

Interprétation :

- Domaine : défini ou pas (cf types)
- Opérateurs de base : propositions

# La logique des prédicats

Interprétation :

- Domaine : défini ou pas (cf types)
- Opérateurs de base : propositions
- Expressions : déduction naturelle

| **Nom** | **Règle d'inférence** |
|---|---|
| $\forall$-introduction | $\dfrac{P(a)}{\forall x \bullet P(x)}$ où $a$ est un terme arbitraire |
| $\forall$-élimination | $\dfrac{\forall x \bullet P(x)}{P(a)}$ où $a$ est un terme arbitraire |

Règles de déduction du calcul des prédicats

# La logique des prédicats

Propriétés :

- Axiomes

# La logique des prédicats

Propriétés :

- Axiomes
- Théorèmes : exemple de la distribution

| Nom | Propriété | Formule |
|---|---|---|
| $P'_5$ | $\forall$ sur $\wedge$ | $(\forall x \bullet P(x) \wedge Q(x)) \Leftrightarrow (\forall x \bullet P(x)) \wedge (\forall x \bullet Q(x))$ |
| $P'_6$ | $\forall$ sur $\vee$ | $(\forall x \bullet P(x) \vee Q(x)) \Leftarrow (\forall x \bullet P(x)) \vee (\forall x \bullet Q(x))$ |
| $P'_7$ | $\exists$ sur $\wedge$ | $(\exists x \bullet P(x) \wedge Q(x)) \Rightarrow (\exists x \bullet P(x)) \wedge (\exists x \bullet Q(x))$ |
| $P'_8$ | $\exists$ sur $\vee$ | $(\exists x \bullet P(x) \vee Q(x)) \Leftrightarrow (\exists x \bullet P(x)) \vee (\exists x \bullet Q(x))$ |

# La logique des prédicats

Propriétés :

- Axiomes
- Théorèmes : exemple de la distribution
- Preuve $\Rightarrow$ propositions

# La logique des prédicats

Propriétés :

- Axiomes
- Théorèmes : exemple de la distribution
- Preuve $\Rightarrow$ propositions

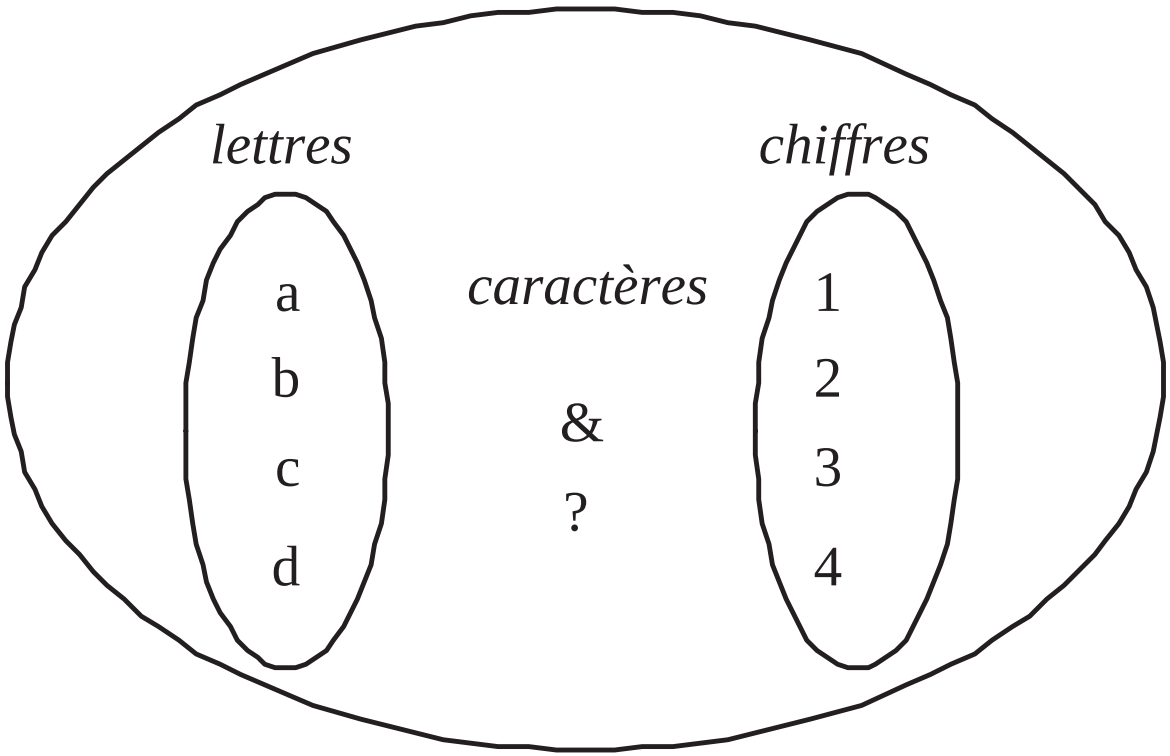
++ Détails : [AV01], chapitre 2, section 4

# Les types et les ensembles

- Ensemble : collection d'éléments

# Les types et les ensembles

- Ensemble : collection d'éléments
- Types : ensembles disjoints d'éléments



# Les types et les ensembles

- Ensemble : collection d'éléments
- Types : ensembles disjoints d'éléments
- Différents types

# Les types et les ensembles

- Ensemble : collection d'éléments
- Types : ensembles disjoints d'éléments
- Différents types
  - vide (polymorphe)

# Les types et les ensembles

- Ensemble : collection d'éléments
- Types : ensembles disjoints d'éléments
- Différents types
  - vide (polymorphe)
  - de base (indéfinis)

# Les types et les ensembles

- Ensemble : collection d'éléments
- Types : ensembles disjoints d'éléments
- Différents types
  - vide (polymorphe)
  - de base (indéfinis)
  - de prédéfinis ($\mathbb{Z}$)

# Les types et les ensembles

- Ensemble : collection d'éléments
- Types : ensembles disjoints d'éléments
- Différents types
  - vide (polymorphe)
  - de base (indéfinis)
  - de prédéfinis ($\mathbb{Z}$)
  - libres (énumérations, intervalles) $bool ::= vrai \mid faux$

# Les types et les ensembles

- Ensemble : collection d'éléments
- Types : ensembles disjoints d'éléments
- Différents types
  - vide (polymorphe)
  - de base (indéfinis)
  - de prédéfinis ($\mathbb{Z}$)
  - libres (énumérations, intervalles) $bool ::= vrai \mid faux$
  - déclarés

# Les types et les ensembles

- Ensemble : collection d'éléments
- Types : ensembles disjoints d'éléments
- Différents types
  - vide (polymorphe)
  - de base (indéfinis)
  - de prédéfinis ($\mathbb{Z}$)
  - libres (énumérations, intervalles) $bool ::= vrai \mid faux$
  - déclarés
    - extension $nat \mathrel{\widehat=} \{0, 1, 2, 3, 4, 5\}$

# Les types et les ensembles

- Ensemble : collection d'éléments
- Types : ensembles disjoints d'éléments
- Différents types
  - vide (polymorphe)
  - de base (indéfinis)
  - de prédéfinis ($\mathbb{Z}$)
  - libres (énumérations, intervalles) $bool ::= vrai \mid faux$
  - déclarés
    - extension $nat \mathrel{\widehat=} \{0, 1, 2, 3, 4, 5\}$
    - compréhension $nat\_pair \mathrel{\widehat=} \{x : \mathbb{Z} \mid x \geq 0 \bullet 2 * x\}$

# Les types et les ensembles

- Ensemble : collection d'éléments
- Types : ensembles disjoints d'éléments
- Différents types
  - vide (polymorphe)
  - de base (indéfinis)
  - de prédéfinis ($\mathbb{Z}$)
  - libres (énumérations, intervalles) $bool ::= vrai \mid faux$
  - déclarés
    - extension $nat \mathrel{\widehat=} \{0, 1, 2, 3, 4, 5\}$
    - compréhension $nat\_pair \mathrel{\widehat=} \{x : \mathbb{Z} \mid x \geq 0 \bullet 2 * x\}$
    - récursif $nat ::= zero \mid succ\langle\langle nat \rangle\rangle$

# Les types et les ensembles

- Opérateurs

# Les types et les ensembles

- Opérateurs
  - éléments : égalité, différence

# Les types et les ensembles

- Opérateurs
  - éléments : égalité, différence
  - éléments/ensemble : déclaration de type, appartenance

# Les types et les ensembles

- Opérateurs
  - éléments : égalité, différence
  - éléments/ensemble : déclaration de type, appartenance
  - ensembles : intersection, union, produit cartésien, différence, cardinal, puissance...

# Les types et les ensembles

- Opérateurs
  - éléments : égalité, différence
  - éléments/ensemble : déclaration de type, appartenance
  - ensembles : intersection, union, produit cartésien, différence, cardinal, puissance...
- exemples

$$ens \mathrel{\widehat=} \{1, 2, 3\} \qquad alpha ::= a \mid b$$
$$1 \in ens \qquad \#ens = 3$$
$$ens \subseteq \mathbb{N}$$
$$\mathbb{P}\, ens = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{1,3\}, \{1,2,3\}\}$$
$$ens \times alpha = \{(1,a), (1,b), (2,a), (2,b), (3,a), (3,b)\}$$
$$ens \cup \mathbb{N} = \mathbb{N} \qquad \wedge \qquad ens \cap \mathbb{N} = ens$$

# Les types et les ensembles

- Toutes les variables sont typées en Z.
  - déclaration libre $x_{1a}, x_{1b} : T_1;\ x_2 : T_2;\ \ldots\ x_n : T_n$
  - déclaration contrainte

$$\begin{array}{|l} x : \mathbb{Z} \\ \hline 0 \leq x \end{array}$$

  - portée de la déclaration (global, schéma, liaison)

# Les types et les ensembles

- Toutes les variables sont typées en Z.
- Différents types
  - vide $\emptyset$, $\{\}$
  - de base $[PERSONNE]$ (global)
  - de prédéfinis ($\mathbb{Z}$)
  - libres $Etat ::= libre \mid occupé$
  - déclarés
    - extension $nat \mathrel{\widehat=} \{0, 1, 2, 3, 4, 5\}$
    - compréhension $\boldsymbol{N} \mathrel{\widehat=} \{x : \mathbb{Z} \mid x \geq 0\}$
    - récursif

$$arbreBinaire ::= Feuille\langle\langle E \rangle\rangle \mid Noeud \langle\langle arbreBinaire \times arbreBinaire \rangle\rangle$$

# Les types et les ensembles

- Opérateurs

# Les types et les ensembles

- Opérateurs
  - éléments : $x \notin S \Leftrightarrow \neg (x \in S)$

# Les types et les ensembles

- Opérateurs
  - éléments : $x \notin S \Leftrightarrow \neg (x \in S)$
  - éléments/ensemble : $\in$

# Les types et les ensembles

- Opérateurs
  - éléments : $x \notin S \Leftrightarrow \neg (x \in S)$
  - éléments/ensemble : $\in$
  - ensembles :

| | |
|---|---|
| $S \subseteq T$ | $(\forall x : S \bullet x \in T)$ |
| $\boldsymbol{P}\, S$ | $S \in \boldsymbol{P}\, T \Leftrightarrow (\forall x \bullet x \in S \Rightarrow x \in T)$ |
| $S \times T$ | $\{x : S; y : T \bullet (x, y)\}$ |
| $S \cap T$ | $\{x : X \mid x \in S \wedge x \in T\}$ avec $S, T : \boldsymbol{P}\, X$ |

# Les types et les ensembles

- Propriétés

| | |
|---|---|
| appartient | $\forall x \bullet x \in \{x\}$ |
| vide | $\neg\ \exists x \bullet x \in \{\}$ |
| extension | $(S = T) \Leftrightarrow (\forall x \bullet x \in S \Leftrightarrow x \in T)$ |
| prédicat | $x \in \{y : S \mid P(y)\} \Leftrightarrow (x \in S \wedge P(x))$ |
| motif | $x \in \{y : S \bullet t(y)\} \Leftrightarrow (\exists y \bullet y \in S \wedge x = t(y))$ |

# Les types et les ensembles

- Propriétés

| | |
|---|---|
| appartient | $\forall x \bullet x \in \{x\}$ |
| vide | $\neg\ \exists x \bullet x \in \{\}$ |
| extension | $(S = T) \Leftrightarrow (\forall x \bullet x \in S \Leftrightarrow x \in T)$ |
| prédicat | $x \in \{y : S \mid P(y)\} \Leftrightarrow (x \in S \land P(x))$ |
| motif | $x \in \{y : S \bullet t(y)\} \Leftrightarrow (\exists y \bullet y \in S \land x = t(y))$ |

- Preuve $\Rightarrow$ logique

# Les types et les ensembles

- Propriétés

| | |
|---|---|
| appartient | $\forall x \bullet x \in \{x\}$ |
| vide | $\neg\ \exists x \bullet x \in \{\}$ |
| extension | $(S = T) \Leftrightarrow (\forall x \bullet x \in S \Leftrightarrow x \in T)$ |
| prédicat | $x \in \{y : S \mid P(y)\} \Leftrightarrow (x \in S \land P(x))$ |
| motif | $x \in \{y : S \bullet t(y)\} \Leftrightarrow (\exists y \bullet y \in S \land x = t(y))$ |

- Preuve $\Rightarrow$ logique

++ Détails : [AV01], chapitre 2, section 5

# Cas des booléens

- Pas de type prédéfini, aucun lien avec les constantes de la logique !

# Cas des booléens

- Pas de type prédéfini, aucun lien avec les constantes de la logique !
- 2 solutions :

# Cas des booléens

- Pas de type prédéfini, aucun lien avec les constantes de la logique !
- 2 solutions :
  - Type libre $bool ::= vrai \mid faux$

# Cas des booléens

- Pas de type prédéfini, aucun lien avec les constantes de la logique !
- 2 solutions :
  - Type libre $bool ::= vrai \mid faux$
    - \+ permet de rendre un booléen en résultat

# Cas des booléens

- Pas de type prédéfini, aucun lien avec les constantes de la logique !
- 2 solutions :
  - Type libre $bool ::= vrai \mid faux$
    - \+ permet de rendre un booléen en résultat
    - \- implique un test permanent de la valeur

# Cas des booléens

- Pas de type prédéfini, aucun lien avec les constantes de la logique !
- 2 solutions :
  - Type libre $bool ::= vrai \mid faux$
    - \+ permet de rendre un booléen en résultat
    - \- implique un test permanent de la valeur
  - Sous-ensemble

$$[ETUDIANT]$$
$$inscrits, admis : \mathbb{P}\ ETUDIANT$$
$$admis \subseteq inscrits$$

# Cas des booléens

- Pas de type prédéfini, aucun lien avec les constantes de la logique !
- 2 solutions :
  - Type libre $bool ::= vrai \mid faux$
    - \+ permet de rendre un booléen en résultat
    - \- implique un test permanent de la valeur
  - Sous-ensemble

    $[ETUDIANT]$
    $inscrits, admis : \mathbb{P}\ ETUDIANT$
    $admis \subseteq inscrits$

    \+ intégration directe avec la logique de Z

# Cas des booléens

- Pas de type prédéfini, aucun lien avec les constantes de la logique !
- 2 solutions :
  - Type libre $bool ::= vrai \mid faux$
    - \+ permet de rendre un booléen en résultat
    - \- implique un test permanent de la valeur
  - Sous-ensemble

    $[ETUDIANT]$
    $inscrits, admis : \mathbb{P}\ ETUDIANT$
    $admis \subseteq inscrits$

    - \+ intégration directe avec la logique de Z
    - \- pas de booléen en résultat

# Cas des booléens

- Pas de type prédéfini, aucun lien avec les constantes de la logique !
- 2 solutions :
  - Type libre $bool ::= vrai \mid faux$
    - \+ permet de rendre un booléen en résultat
    - \- implique un test permanent de la valeur
  - Sous-ensemble

    $[ETUDIANT]$
    $inscrits, admis : \mathbb{P}\ ETUDIANT$
    $admis \subseteq inscrits$

    - \+ intégration directe avec la logique de Z
    - \- pas de booléen en résultat
- Preuve $\Rightarrow$ logique

# Cas des booléens

- Pas de type prédéfini, aucun lien avec les constantes de la logique !
- 2 solutions :
  - Type libre $bool ::= vrai \mid faux$
    - \+ permet de rendre un booléen en résultat
    - \- implique un test permanent de la valeur
  - Sous-ensemble

    $[ETUDIANT]$
    $inscrits, admis : \mathbb{P}\ ETUDIANT$
    $admis \subseteq inscrits$

    - \+ intégration directe avec la logique de Z
    - \- pas de booléen en résultat
- Preuve $\Rightarrow$ logique

++ Détails : [AV01], chapitre 2, section 5

# Plan

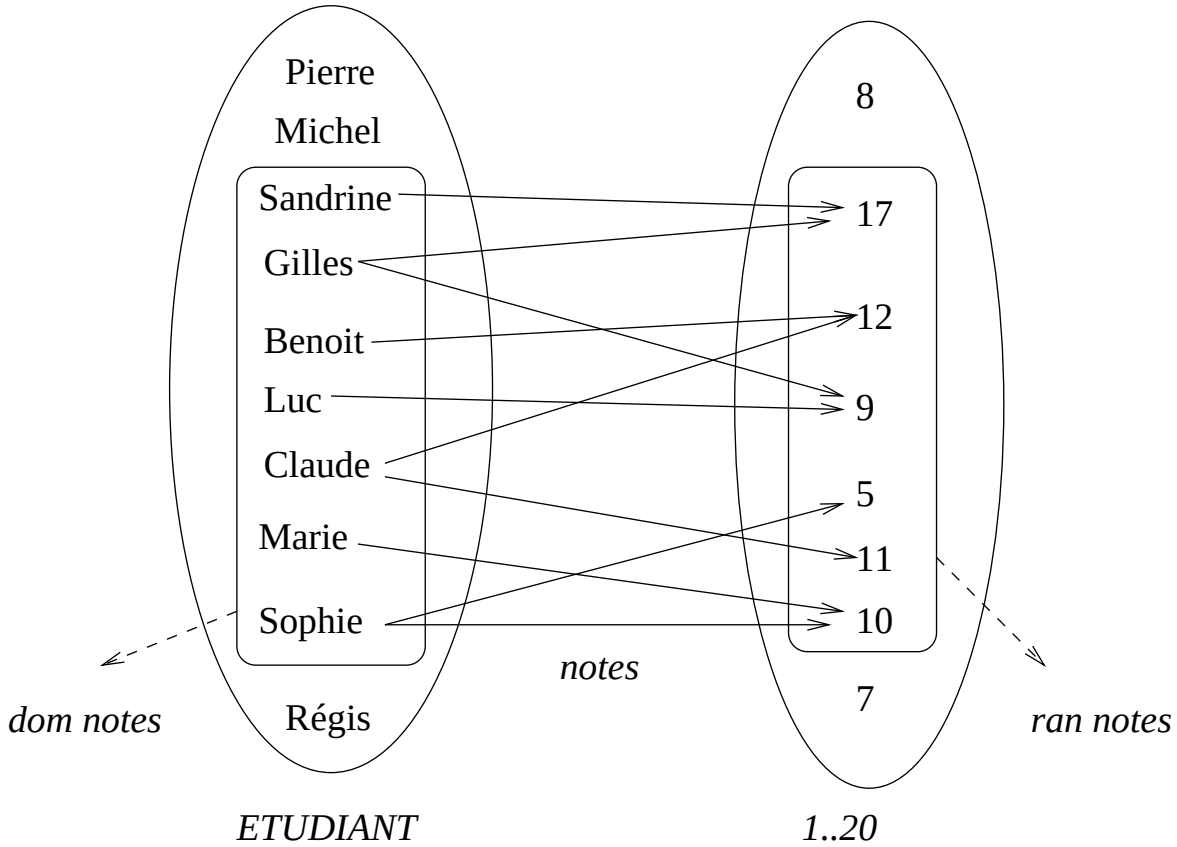
# Les relations

- Relation : sous-ensemble d'un produit cartésien

# Les relations

- Relation : sous-ensemble d'un produit cartésien

$\Rightarrow$ binaire, orientée



# Les relations

- Relation : sous-ensemble d'un produit cartésien

$\Rightarrow$ binaire, orientée

$\Rightarrow$ couple (antécédent, image)

# Les relations

- Relation : sous-ensemble d'un produit cartésien

$\Rightarrow$ binaire, orientée

$\Rightarrow$ couple (antécédent, image)

- Domaine et co-domaine

# Les relations

- Relation : sous-ensemble d'un produit cartésien

$\Rightarrow$ binaire, orientée

$\Rightarrow$ couple (antécédent, image)

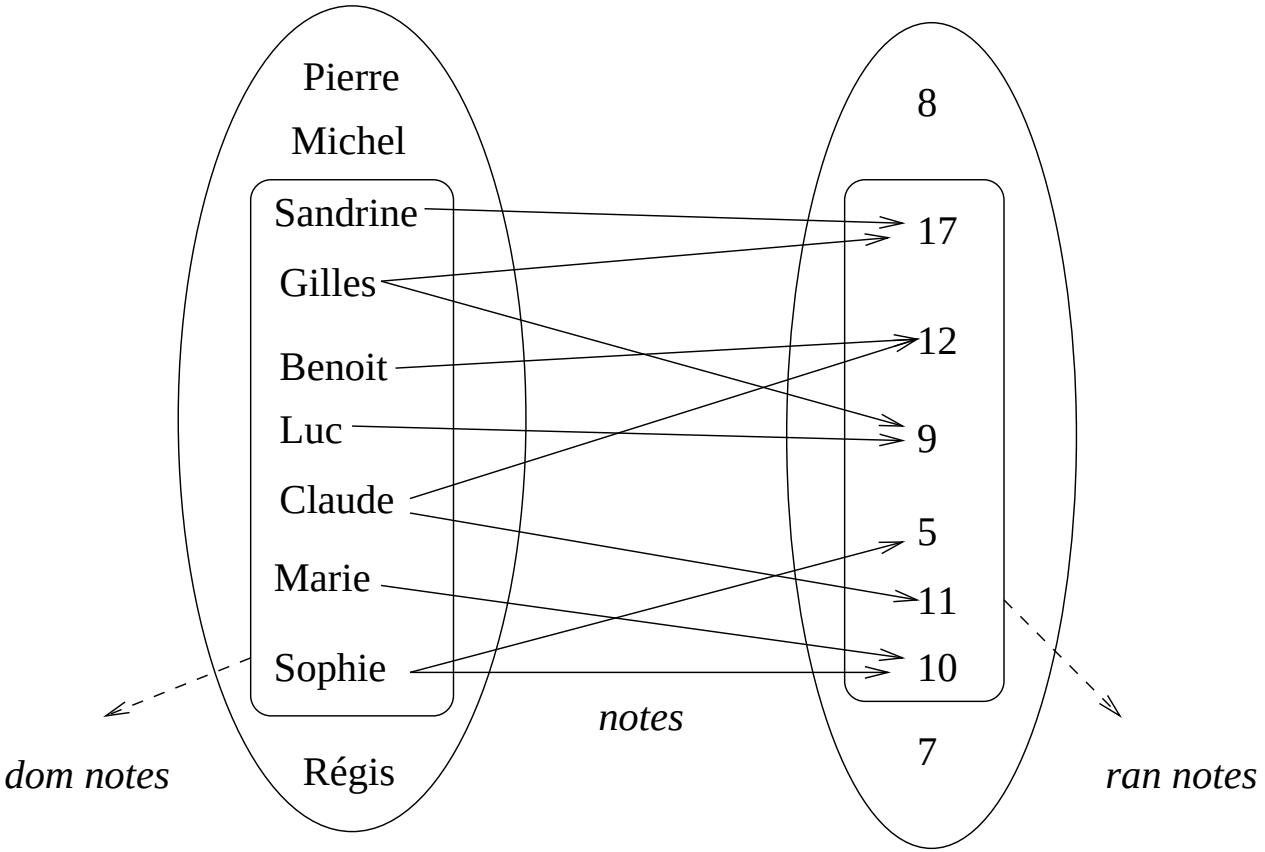
- Domaine et co-domaine
- Opérateurs : inversions, restrictions, compositions, fermeture...

# Les relations

- Relation : $X \leftrightarrow Y \mathrel{\widehat=} \boldsymbol{P}\,(X \times Y)$.
- Soient $X$, $Y$ et $Z$ des ensembles ;
  $x : X$ ; $y : Y$ et $R : X \leftrightarrow Y$.

| | |
|---|---|
| $x\ R\ y$ | $\mathrel{\widehat=} (x, y) \in R$ ; ou $x\ \underline{R}\ y$ ou $x \mapsto y \in R$ |
| $\mathrm{dom}\, R$ [ran] | $\mathrel{\widehat=} \{x : X \mid (\exists y : Y \bullet x\ R\ y)\}$ |
| $\mathrm{id}\, X$ | $\mathrel{\widehat=} \{x : X \bullet x \mapsto x\}$ |
| $R^{-1}$ ou $R^{\sim}$ | $\mathrel{\widehat=} \{y : Y;\ x : X \mid x\ R\ y\}$ |
| $R \circ R'$ | $\mathrel{\widehat=} R' \fatsemi R$ |
| $R^*$ | $\mathrel{\widehat=} \bigcup \{n : \boldsymbol{N} \bullet R^n\}$ |
| $R(\!\mid S \mid\!)$ | $\mathrel{\widehat=} \{y : Y \mid (\exists x : S \bullet x\ R\ y)\}$ |
| $S \lhd R$ | $\mathrel{\widehat=} \{x : X;\ y : Y \mid x \in S \wedge x\ R\ y\}$ |
| $R \oplus R'$ | $\mathrel{\widehat=} (\mathrm{dom}\, R' \lhd R) \cup R'$ |

# Les relations




Exemple

*ETUDIANT*

*1..20*

# Les relations

Définitions axiomatiques, génériques

$$\begin{array}{|l} \_ \uparrow \_ : \mathbb{N} \leftrightarrow \mathbb{N} \\ \hline \forall x, y : \mathbb{N} \bullet x \uparrow y \Leftrightarrow \exists z : \mathbb{N} \bullet x \times z = y \end{array}$$

$$\begin{array}{|l} [X] \\ \hline \_ \subseteq \_ : \mathbb{P}\, X \leftrightarrow \mathbb{P}\, X \\ \hline \forall S, T : \mathbb{P}\, X \bullet S \subseteq T \Leftrightarrow \forall x : X \bullet x \in S \Rightarrow x \in T \end{array}$$

# Les relations

Propriétés

| |
|---|
| $reflexive(R) \Leftrightarrow \forall x : X \bullet x\ R\ x$ |
| $symmetric(R) \Leftrightarrow \forall x, y : X \bullet x\ R\ y \Rightarrow y\ R\ x$ |
| $antisymmetric(R) \Leftrightarrow \forall x, y : X \bullet x\ R\ y \land y\ R\ x \Rightarrow x = y$ |
| $asymmetric(R) \Leftrightarrow \forall x, y : X \bullet x\ R\ y \Rightarrow \neg\ (y\ R\ x)$<br>ou $asymmetric(R) \Leftrightarrow \forall x, y : X \bullet \neg\ (x\ R\ y \land y\ R\ x)$ |
| $transitive(R) \Leftrightarrow \forall x, y, z : X \bullet x\ R\ y \land y\ R\ z \Rightarrow x\ R\ z$ |
| $equivalence(R) \Leftrightarrow$<br>$reflexive(R) \land symmetric(R) \land transitive(R)$ |

# Les fonctions

- Fonction (partielle) : relation dont les antécédents ont au plus une image

# Les fonctions

- Fonction (partielle) : relation dont les antécédents ont au plus une image
- Variantes :

# Les fonctions

- Fonction (partielle) : relation dont les antécédents ont au plus une image
- Variantes :
  - Fonction totale : fonction dont les antécédents ont tous une image

# Les fonctions

- Fonction (partielle) : relation dont les antécédents ont au plus une image
- Variantes :
  - Fonction totale : fonction dont les antécédents ont tous une image
  - Injection : fonction dont les images ont au plus un antécédent

# Les fonctions

- Fonction (partielle) : relation dont les antécédents ont au plus une image
- Variantes :
  - Fonction totale : fonction dont les antécédents ont tous une image
  - Injection : fonction dont les images ont au plus un antécédent
  - Surjection : fonction dont les images ont au moins un antécédent

# Les fonctions

- Fonction (partielle) : relation dont les antécédents ont au plus une image
- Variantes :
  - Fonction totale : fonction dont les antécédents ont tous une image
  - Injection : fonction dont les images ont au plus un antécédent
  - Surjection : fonction dont les images ont au moins un antécédent
  - Bijection : fonction dont les images ont exactement un antécédent

# Les fonctions

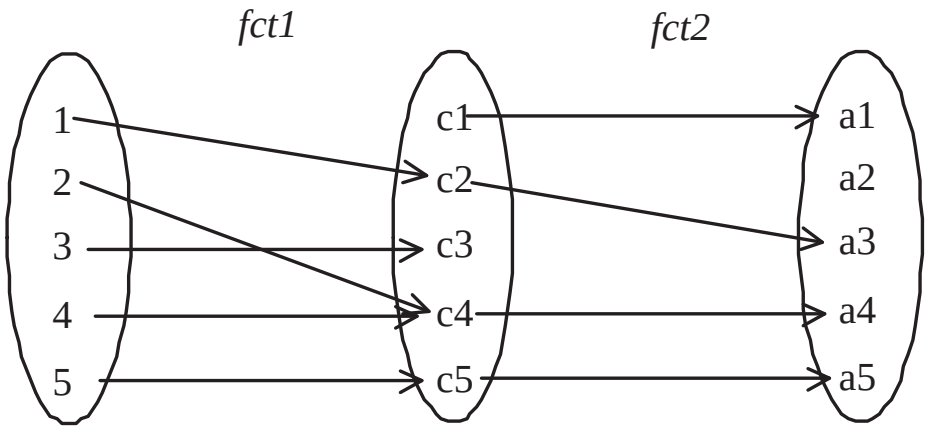
- Fonction (partielle) : relation dont les antécédents ont au plus une image
- Variantes :
  - Fonction totale : fonction dont les antécédents ont tous une image
  - Injection : fonction dont les images ont au plus un antécédent
  - Surjection : fonction dont les images ont au moins un antécédent
  - Bijection : fonction dont les images ont exactement un antécédent
  - $\lambda$-abstraction : fonction anonyme

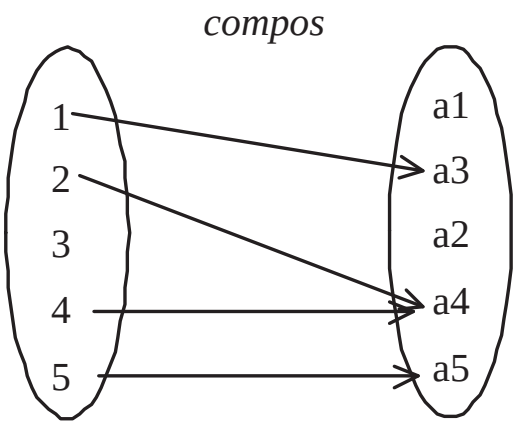
# Les fonctions

- Fonction (partielle) : relation dont les antécédents ont au plus une image
- Variantes :
  - Fonction totale : fonction dont les antécédents ont tous une image
  - Injection : fonction dont les images ont au plus un antécédent
  - Surjection : fonction dont les images ont au moins un antécédent
  - Bijection : fonction dont les images ont exactement un antécédent
  - $\lambda$-abstraction : fonction anonyme
- Opérateurs : idem relations

# Les fonctions

## Exemple : composition



# Les fonctions

- Fonction (partielle) :
  $X \pfun Y \mathrel{\widehat=} \{f : X \rel Y \mid (\forall x : domf \bullet (\exists! y : Y \bullet x\ f\ y))\}$

# Les fonctions

- Fonction (partielle) :
  $X \pfun Y \mathrel{\widehat=} \{f : X \rel Y \mid (\forall x : dom f \bullet (\exists! y : Y \bullet x\ f\ y))\}$
- Fonction totale : $X \fun Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{dom}\, f = X\}$

# Les fonctions

- Fonction (partielle) :
  $X \pfun Y \mathrel{\widehat=} \{f : X \rel Y \mid (\forall x : domf \bullet (\exists! y : Y \bullet x\ f\ y))\}$
- Fonction totale : $X \fun Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{dom}\, f = X\}$
- Injection : $X \pinj Y$
  $\mathrel{\widehat=} \{f : X \pfun Y \mid (\forall x_1, x_2 : \mathrm{dom}\, f \bullet f(x_1) = f(x_2) \Rightarrow x_1 = x_2)\}$

# Les fonctions

- Fonction (partielle) :
  $X \pfun Y \mathrel{\widehat=} \{f : X \rel Y \mid (\forall x : \mathit{dom} f \bullet (\exists! y : Y \bullet x\ f\ y))\}$
- Fonction totale : $X \fun Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{dom}\, f = X\}$
- Injection : $X \pinj Y$
  $\mathrel{\widehat=} \{f : X \pfun Y \mid (\forall x_1, x_2 : \mathrm{dom}\, f \bullet f(x_1) = f(x_2) \Rightarrow x_1 = x_2)\}$
- Surjection : $X \psur Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{ran}\, f = Y\}$

# Les fonctions

- Fonction (partielle) :
  $X \pfun Y \mathrel{\widehat=} \{f : X \rel Y \mid (\forall x : domf \bullet (\exists! y : Y \bullet x\ f\ y))\}$
- Fonction totale : $X \fun Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{dom}\, f = X\}$
- Injection : $X \pinj Y$
  $\mathrel{\widehat=} \{f : X \pfun Y \mid (\forall x_1, x_2 : \mathrm{dom}\, f \bullet f(x_1) = f(x_2) \Rightarrow x_1 = x_2)\}$
- Surjection : $X \psur Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{ran}\, f = Y\}$
- Bijection : $X \bij Y \mathrel{\widehat=} (X \psur Y) \cap (X \pinj Y)$

# Les fonctions

- Fonction (partielle) :
  $X \pfun Y \mathrel{\widehat=} \{f : X \rel Y \mid (\forall x : domf \bullet (\exists! y : Y \bullet x\ f\ y))\}$
- Fonction totale : $X \fun Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{dom}\, f = X\}$
- Injection : $X \pinj Y$
  $\mathrel{\widehat=} \{f : X \pfun Y \mid (\forall x_1, x_2 : \mathrm{dom}\, f \bullet f(x_1) = f(x_2) \Rightarrow x_1 = x_2)\}$
- Surjection : $X \psur Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{ran}\, f = Y\}$
- Bijection : $X \bij Y \mathrel{\widehat=} (X \psur Y) \cap (X \pinj Y)$
- Fonction finie : $X \ffun Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{dom}\, f \in \boldsymbol{F}\, X\}$

# Les fonctions

- Fonction (partielle) :
  $X \pfun Y \mathrel{\widehat=} \{f : X \rel Y \mid (\forall x : domf \bullet (\exists! y : Y \bullet x\ f\ y))\}$
- Fonction totale : $X \fun Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{dom}\, f = X\}$
- Injection : $X \pinj Y$
  $\mathrel{\widehat=} \{f : X \pfun Y \mid (\forall x_1, x_2 : \mathrm{dom}\, f \bullet f(x_1) = f(x_2) \Rightarrow x_1 = x_2)\}$
- Surjection : $X \psur Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{ran}\, f = Y\}$
- Bijection : $X \bij Y \mathrel{\widehat=} (X \psur Y) \cap (X \pinj Y)$
- Fonction finie : $X \ffun Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{dom}\, f \in \boldsymbol{F}\, X\}$
- $\lambda$-abstraction : $(\lambda x : X \mid P \bullet t) \mathrel{\widehat=} \{x : X \mid P \bullet x \mapsto t\}$

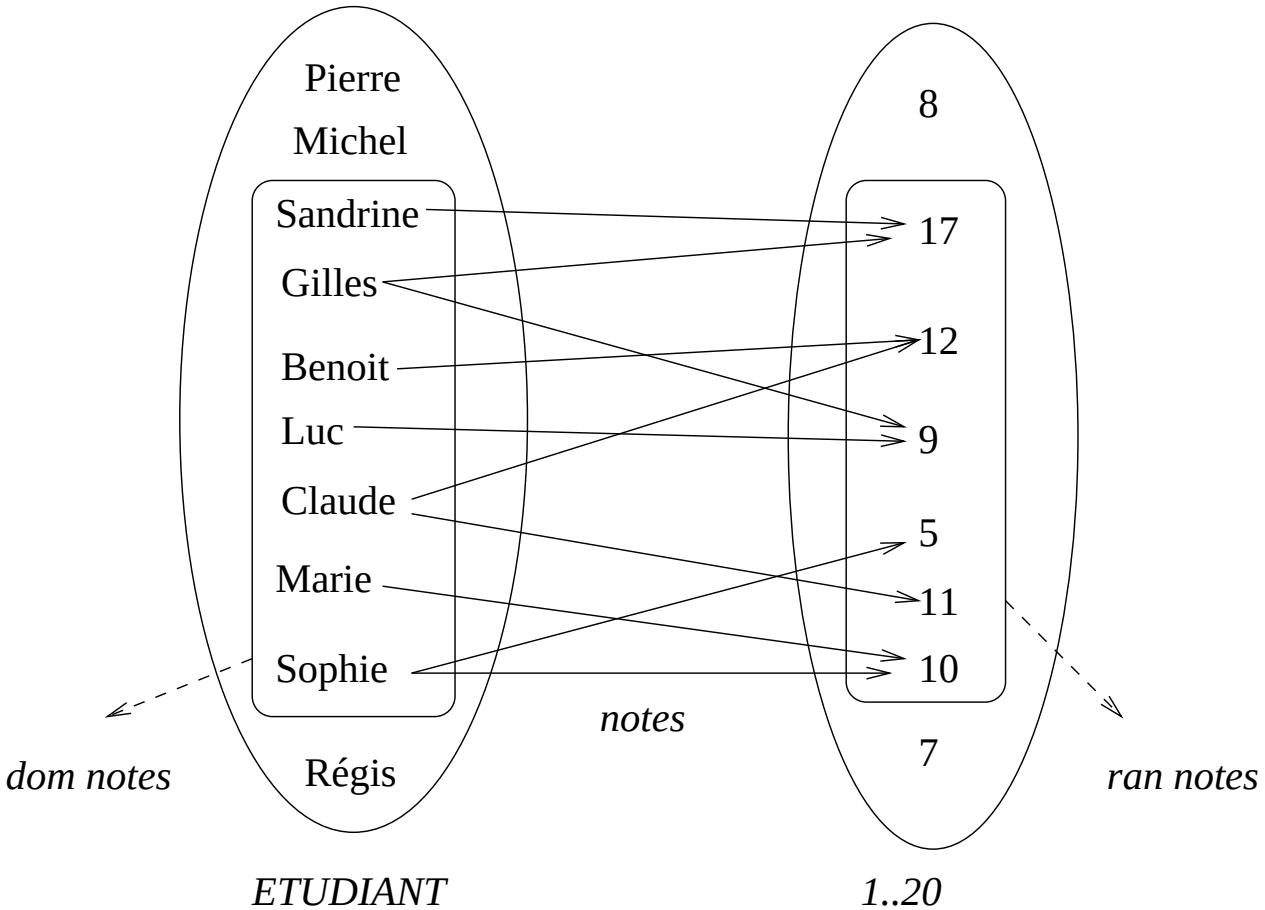
# Les fonctions

- Fonction (partielle) :
  $X \pfun Y \mathrel{\widehat=} \{f : X \rel Y \mid (\forall x : domf \bullet (\exists! y : Y \bullet x\ f\ y))\}$
- Fonction totale : $X \fun Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{dom}\, f = X\}$
- Injection : $X \pinj Y$
  $\mathrel{\widehat=} \{f : X \pfun Y \mid (\forall x_1, x_2 : \mathrm{dom}\, f \bullet f(x_1) = f(x_2) \Rightarrow x_1 = x_2)\}$
- Surjection : $X \psur Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{ran}\, f = Y\}$
- Bijection : $X \bij Y \mathrel{\widehat=} (X \psur Y) \cap (X \pinj Y)$
- Fonction finie : $X \ffun Y \mathrel{\widehat=} \{f : X \pfun Y \mid \mathrm{dom}\, f \in \boldsymbol{F}\, X\}$
- $\lambda$-abstraction : $(\lambda x : X \mid P \bullet t) \mathrel{\widehat=} \{x : X \mid P \bullet x \mapsto t\}$

  Exemple fonction successeur :
  $\mathrm{succ} : \boldsymbol{N} \fun \boldsymbol{N} \mathrel{\widehat=} \forall n : \boldsymbol{N} \bullet \mathrm{succ}\,(n) = n + 1$

# Les fonctions

Exemple : Calculer la moyenne des étudiants.



# Les fonctions

Pas d'itérateur en Z $\Rightarrow$ fonction annexe récursive

$$
\begin{array}{|l}
som\_ : \mathbb{P}\mathbb{N} \rightarrow \mathbb{N} \\
moy\_ : \mathbb{P}\mathbb{N} \rightarrow \mathbb{N} \\
\hline
som\ \emptyset = 0 \land \\
\forall s : \mathbb{P}\mathbb{N} \mid s \neq \emptyset \bullet (\exists x : \mathbb{N} \mid x \in s \bullet som\ s = x + som\ (s \setminus \{x\})) \\
moy\ \emptyset = 0 \land \\
\forall s : \mathbb{P}\mathbb{N} \mid s \neq \emptyset \bullet moy\ s = (som\ s)\ \mathrm{div}\ \#s
\end{array}
$$

$$
\begin{array}{l}
moyenne : ETUDIANT \pfun 0 \upto 20 \\
\mathrm{dom}\ moyenne = \mathrm{dom}\ notes \\
\forall e : ETUDIANT \mid e \in \mathrm{dom}\ notes \bullet moyenne\ e = moy(notes(\!|e|\!))
\end{array}
$$

# Les multi-ensembles et les séquences

- Multi-ensemble : ensemble dont les éléments peuvent avoir plusieurs occurrences

# Les multi-ensembles et les séquences

- Multi-ensemble : ensemble dont les éléments peuvent avoir plusieurs occurrences
  - Opérateurs : comptage, appartenance, union, conversions...

# Les multi-ensembles et les séquences

- Multi-ensemble : ensemble dont les éléments peuvent avoir plusieurs occurrences
  - Opérateurs : comptage, appartenance, union, conversions...
- Séquence : fonction dont les antécédents forment un intervalle $\Rightarrow$ ordre (liste)

# Les multi-ensembles et les séquences

- Multi-ensemble : ensemble dont les éléments peuvent avoir plusieurs occurrences
  - Opérateurs : comptage, appartenance, union, conversions...
- Séquence : fonction dont les antécédents forment un intervalle $\Rightarrow$ ordre (liste)
  - les éléments peuvent avoir plusieurs occurrences

# Les multi-ensembles et les séquences

- Multi-ensemble : ensemble dont les éléments peuvent avoir plusieurs occurrences
  - Opérateurs : comptage, appartenance, union, conversions...
- Séquence : fonction dont les antécédents forment un intervalle $\Rightarrow$ ordre (liste)
  - les éléments peuvent avoir plusieurs occurrences
  - Opérateurs : concaténation, premier, suite, inversion, partition...

# Les multi-ensembles et les séquences

- Multi-ensemble : ensemble dont les éléments peuvent avoir plusieurs occurrences
  - Opérateurs : comptage, appartenance, union, conversions...
- Séquence : fonction dont les antécédents forment un intervalle $\Rightarrow$ ordre (liste)
  - les éléments peuvent avoir plusieurs occurrences
  - Opérateurs : concaténation, premier, suite, inversion, partition...
- Séquence injective : les éléments ont une seule occurrence

# Les multi-ensembles et les séquences

- Multi-ensemble : ensemble dont les éléments peuvent avoir plusieurs occurrences
  - Opérateurs : comptage, appartenance, union, conversions...
- Séquence : fonction dont les antécédents forment un intervalle $\Rightarrow$ ordre (liste)
  - les éléments peuvent avoir plusieurs occurrences
  - Opérateurs : concaténation, premier, suite, inversion, partition...
- Séquence injective : les éléments ont une seule occurrence
- Matrice : séquence de séquences (pas de notation spécifique)

# Les multi-ensembles

- Définition : $\mathrm{bag}\, X \mathrel{\widehat{=}} X \pfun \mathbb{N}^+$
- Opérateurs : comptage, appartenance, union, conversions...

$$
\begin{array}{|l}
[X] \\
\hline
\mathrm{count} : \mathrm{bag}\, X \rightarrowtail\!\!\!\!\!\rightarrow (X \to \mathbb{N}) \\
\_ \sharp \_ : \mathrm{bag}\, X \times X \to \mathbb{N} \\
\_ \otimes \_ : \mathbb{N} \times \mathrm{bag}\, X \to \mathrm{bag}\, X \\
\hline
\forall B : \mathrm{bag}\, X \bullet \mathrm{count}\ B = (\lambda x : X \bullet 0) \oplus B \\
\forall x : X;\ B : \mathrm{bag}\, X \bullet B \sharp x = \mathrm{count}\ B\ x \\
\forall n : \mathbb{N};\ x : X;\ B : \mathrm{bag}\, X \bullet (n \otimes B) \sharp x = n * (B \sharp x) \\
\hline
\end{array}
$$

# Les séquences

- Définition

# Les séquences

- Définition
  - formelle : $\mathrm{seq}\, X \mathrel{\widehat=} \{f : \mathbb{N}^+ \pfun X \mid \exists n : \mathbb{N} \bullet \mathrm{dom}\, f = 1 \upto n\}$

# Les séquences

- Définition
  - formelle : $\mathrm{seq}\, X \mathrel{\widehat{=}} \{f : \mathbb{N}^+ \pfun X \mid \exists n : \mathbb{N} \bullet \mathrm{dom}\, f = 1 \upto n\}$
  - en extension : $\langle x_1, x_2, \ldots, x_n \rangle$ soit $\{1 \mapsto x_1, 2 \mapsto x_2, \ldots, n \mapsto x_n\}$

# Les séquences

- Définition
  - formelle : $\mathrm{seq}\, X \mathrel{\widehat{=}} \{f : \mathbb{N}^+ \pfun X \mid \exists n : \mathbb{N} \bullet \mathrm{dom}\, f = 1 \upto n\}$
  - en extension : $\langle x_1, x_2, \ldots, x_n \rangle$ soit $\{1 \mapsto x_1, 2 \mapsto x_2, \ldots, n \mapsto x_n\}$
- Opérateurs :

# Les séquences

- Définition
  - formelle : $\mathrm{seq}\, X \mathrel{\widehat{=}} \{f : \mathbb{N}^+ \pfun X \mid \exists n : \mathbb{N} \bullet \mathrm{dom}\, f = 1 \upto n\}$
  - en extension : $\langle x_1, x_2, \ldots, x_n \rangle$ soit $\{1 \mapsto x_1, 2 \mapsto x_2, \ldots, n \mapsto x_n\}$
- Opérateurs :
  - accès au i$^{ème}$ élément : $s\ i$ (si $i \in 1 \upto \#s$)

# Les séquences

- Définition
  - formelle : $\mathrm{seq}\, X \mathrel{\widehat{=}} \{f : \mathbb{N}^+ \pfun X \mid \exists n : \mathbb{N} \bullet \mathrm{dom}\, f = 1 \upto n\}$
  - en extension : $\langle x_1,\ x_2,\ \ldots,\ x_n \rangle$ soit $\{1 \mapsto x_1,\ 2 \mapsto x_2,\ \ldots,\ n \mapsto x_n\}$
- Opérateurs :
  - accès au i$^{ème}$ élément : $s\ i$ (si $i \in 1 \upto \#s$)
  - concaténation : $s \frown t \mathrel{\widehat{=}} s \cup \{n : \mathrm{dom}\, t \bullet n + \#s \mapsto t(n)\}$

# Les séquences

- Définition
  - formelle : $\mathrm{seq}\, X \mathrel{\widehat=} \{f : \mathbb{N}^+ \pfun X \mid \exists n : \mathbb{N} \bullet \mathrm{dom}\, f = 1 \upto n\}$
  - en extension : $\langle x_1,\ x_2,\ \ldots,\ x_n \rangle$ soit $\{1 \mapsto x_1,\ 2 \mapsto x_2,\ \ldots,\ n \mapsto x_n\}$
- Opérateurs :
  - accès au i$^{ème}$ élément : $s\ i$ (si $i \in 1 \upto \#s$)
  - concaténation : $s \frown t \mathrel{\widehat=} s \cup \{n : \mathrm{dom}\, t \bullet n + \#s \mapsto t(n)\}$
  - inversion : $\mathrm{rev}\ s \mathrel{\widehat=} (\lambda n : \mathrm{dom}\, s \bullet s(\#s - n + 1))$

# Les séquences

- Définition
  - formelle : $\mathrm{seq}\, X \mathrel{\widehat=} \{f : \mathbb{N}^+ \pfun X \mid \exists n : \mathbb{N} \bullet \mathrm{dom}\, f = 1 \upto n\}$
  - en extension : $\langle x_1,\ x_2,\ \ldots,\ x_n \rangle$ soit $\{1 \mapsto x_1,\ 2 \mapsto x_2,\ \ldots,\ n \mapsto x_n\}$
- Opérateurs :
  - accès au i$^{ème}$ élément : $s\ i$ (si $i \in 1 \upto \#s$)
  - concaténation : $s \frown t \mathrel{\widehat=} s \cup \{n : \mathrm{dom}\, t \bullet n + \#s \mapsto t(n)\}$
  - inversion : $\mathrm{rev}\ s \mathrel{\widehat=} (\lambda n : \mathrm{dom}\, s \bullet s(\#s - n + 1))$
  - premier : $\mathrm{head}\ s \mathrel{\widehat=} \forall s : \mathrm{seq}_1 X \bullet \mathrm{head}\ s = s(1)$

# Les séquences

- Définition
  - formelle : $\mathrm{seq}\, X \mathrel{\widehat=} \{f : \mathbb{N}^+ \pfun X \mid \exists n : \mathbb{N} \bullet \mathrm{dom}\, f = 1 \upto n\}$
  - en extension : $\langle x_1,\ x_2,\ \ldots,\ x_n \rangle$ soit $\{1 \mapsto x_1,\ 2 \mapsto x_2,\ \ldots,\ n \mapsto x_n\}$
- Opérateurs :
  - accès au i$^{ème}$ élément : $s\ i$ (si $i \in 1 \upto \#s$)
  - concaténation : $s \frown t \mathrel{\widehat=} s \cup \{n : \mathrm{dom}\, t \bullet n + \#s \mapsto t(n)\}$
  - inversion : $\mathrm{rev}\ s \mathrel{\widehat=} (\lambda n : \mathrm{dom}\, s \bullet s(\#s - n + 1))$
  - premier : $\mathrm{head}\ s \mathrel{\widehat=} \forall s : \mathrm{seq}_1 X \bullet \mathrm{head}\ s = s(1)$
  - suite : $\mathrm{tail}\ s \mathrel{\widehat=} \forall s : \mathrm{seq}_1 X \bullet \mathrm{tail}\ s = (\lambda n : 1 \upto \#s - 1 \bullet s(n+1))$

# Les séquences

Exemples.

$$s = \langle a, b, c \rangle,\ t = \langle d, e \rangle$$
$$s = \{(1 \mapsto a), (2 \mapsto b), (3 \mapsto c)\},\ t = \{(1 \mapsto d), (2 \mapsto e)\}$$
$$\#s = 3, \#t = 2$$
$$s \frown t = \langle a, b, c, d, e \rangle$$
$$\mathrm{rev}\ t = \langle e, d \rangle$$
$$\mathrm{last}\ s = c, \mathrm{tail}\ s = \langle b, c \rangle, \mathrm{head}\ s = a$$

# Les séquences

## Opérateurs (suite)

- concaténation distribuée $\frown / ss$ :

$$
\begin{array}{|l}
\hline
[X] \\
\frown / : \operatorname{seq}(\operatorname{seq} X) \to \operatorname{seq} X \\
\hline
\frown /\langle\rangle = \langle\rangle \\
\forall s : \operatorname{seq} X \bullet \frown /\langle s \rangle = s \\
\forall q, r : \operatorname{seq}(\operatorname{seq} X) \bullet \frown /(q \frown r) = (\frown / q) \frown (\frown / r) \\
\hline
\end{array}
$$

# Les séquences

## Opérateurs (suite)

- concaténation distribuée $\frown / ss$ :
- compactage $\mathrm{squash}$, filtrage $\lceil$, extraction $\rceil$ :

$$
\begin{array}{|l}
[X] \\
\hline
\_ \rceil \_ : \boldsymbol{P}\boldsymbol{N}_1 \times \mathrm{seq}\, X \to \mathrm{seq}\, X \\
\_ \lceil \_ : \mathrm{seq}\, X \times \boldsymbol{P}\, X \to \mathrm{seq}\, X \\
\mathrm{squash} : (\boldsymbol{N}_1 \pfun X) \to \mathrm{seq}\, X \\
\hline
\forall U : \boldsymbol{P}\boldsymbol{N}_1;\ s : \mathrm{seq}\, X \bullet U \rceil s = \mathrm{squash}(U \lhd s) \\
\forall s\, \mathrm{seq}\, X;\ V : \boldsymbol{P}\, X \bullet s \lceil V = \mathrm{squash}(s \rhd V) \\
\forall f : \boldsymbol{N}_1 \pfun X \bullet \mathrm{squash}\, f = f \circ (\mu p : 1 \upto \#f \rightarrowtail\!\!\!\!\!\twoheadrightarrow \mathrm{dom}\, f \mid \\
\qquad p \circ \mathrm{succ} \circ p^{\sim} \subseteq (\_ < \_)) \\
\hline
\end{array}
$$

# Les séquences

## Opérateurs (suite)

- concaténation distribuée $\frown / ss$ :
- compactage $\mathrm{squash}$, filtrage $\lceil$, extraction $\rceil$ :
- disjonction $\mathrm{disjoint}$ , partition $\mathrm{partition}$ :

Soit $I$ un ensemble d'indices (sous-ensemble d'entiers).

$$
\begin{array}{|l}
\hline
[I, X] \\
\mathrm{disjoint}\ \_ : \boldsymbol{P}\,(I \pfun \boldsymbol{P}\,X) \\
\_\ \mathrm{partition}\ \_ : (I \pfun \boldsymbol{P}\,X) \rel \boldsymbol{P}\,X \\
\hline
\forall S : I \pfun \boldsymbol{P}\,X;\ T : \boldsymbol{P}\,X \bullet \\
\quad (\mathrm{disjoint}\ S \Leftrightarrow (\forall i, j : \mathrm{dom}\,S \mid i \neq j \bullet S(i) \cap S(j) = \emptyset)) \land \\
\quad (S\ \mathrm{partition}\ T \Leftrightarrow \mathrm{disjoint}\ S \land \bigcup \{i : \mathrm{dom}\,S \bullet S(i)\} = T)) \\
\hline
\end{array}
$$

# Les séquences

Exemples.

$$\frown / \langle s, t \rangle = \langle a, b, c, d, e \rangle$$

$$\text{squash } \{2 \mapsto a,\ 12 \mapsto c,\ 5 \mapsto b\} = \{1 \mapsto a,\ 3 \mapsto c,\ 2 \mapsto b\} = \langle a, b, c \rangle$$

$$\text{disjoint } \langle s, t \rangle = \mathit{true}$$

$$\langle s, t \rangle \text{ partition } \{a, b, c, d, e\} = \mathit{true}$$

# Plan

# Les schémas

- Unité de structuration en Z

# Les schémas

- Unité de structuration en Z
  - déclarations

# Les schémas

- Unité de structuration en Z
  - déclarations
  - prédicat

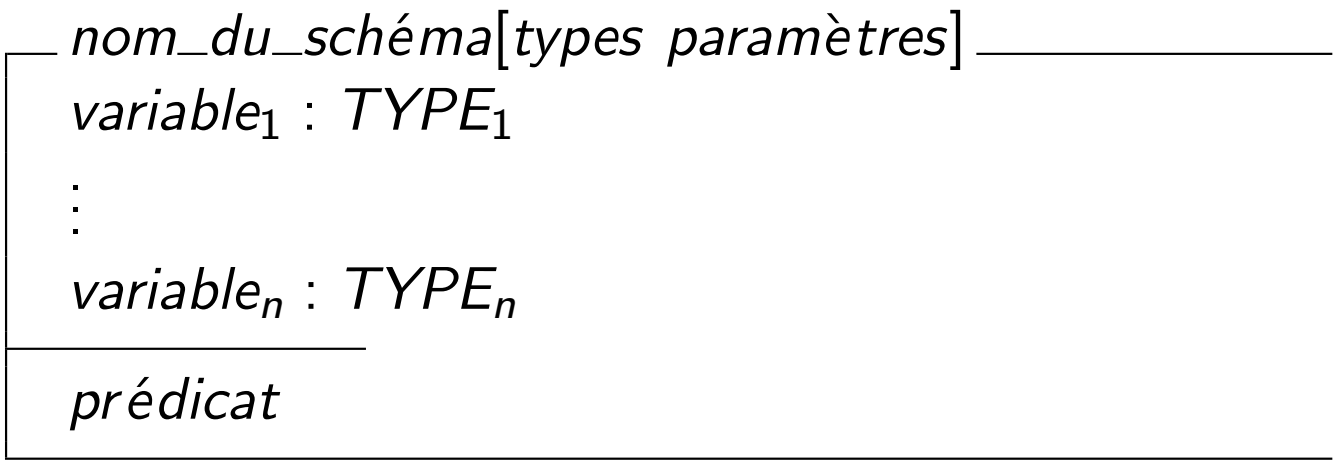
# Les schémas

- Unité de structuration en Z
  - déclarations
  - prédicat
- graphique

$$\begin{array}{|l} \textit{nom\_du\_schéma} \\ \hline \textit{variable}_1 : \textit{TYPE}_1 \\ \vdots \\ \textit{variable}_n : \textit{TYPE}_n \\ \hline \textit{prédicat} \\ \hline \end{array}$$

# Les schémas

- Unité de structuration en Z
  - déclarations
  - prédicat
- graphique
- générique



# Les schémas

- Unité de structuration en Z
  - déclarations
  - prédicat
- graphique
- générique
- type de données

# Les schémas

- Unité de structuration en Z
  - déclarations
  - prédicat
- graphique
- générique
- type de données
- Calcul : inclusion, renommage, opérations logiques, projection, tubage...

# Les schémas

Déclarations contraintes par un prédicat

$$\begin{array}{|l} \textit{Classe} \\ \hline \textit{effectif\_max} : \mathbb{N}_1 \\ \textit{élèves} : \mathbb{P}\ \textit{ETUDIANT} \\ \hline \#\textit{élèves} \leq \textit{effectif\_max} \\ \hline \end{array}$$

ou écrit autrement dans le format horizontal par

$$\textit{Classe} \mathrel{\widehat=} [\textit{effectif\_max} : \mathbb{N}_1; \textit{élèves} : \mathbb{P}\ \textit{ETUDIANT} \mid \#\textit{élèves} \leq \textit{effectif\_max}]$$

# Les schémas

Pas de déclarations circulaires

*ClasseNum*

$$\begin{array}{l} \mathit{numéros} : \mathbb{P}\,\mathbb{N}_1 \\ \mathit{élèves} : \mathit{numéros} \rightarrow \mathit{ETUDIANT} \end{array}$$

les types sont définis

*ClasseNum*

$$\begin{array}{l} \mathit{numéros} : \mathbb{P}\,\mathbb{N}_1 \\ \mathit{élèves} : \mathbb{N}_1 \pfun \mathit{ETUDIANT} \\ \hline \mathrm{dom}\ \mathit{élèves} = \mathit{numéros} \end{array}$$

# Les schémas

Un schéma sert de déclaration (*Multiple*) ou de prédicat (*Pair*)

$$\forall Multiple \bullet y \bmod 2 = 0 \Rightarrow Pair$$

*Multiple*

$$x, y : \mathbb{Z}$$
$$x \geq 0 \land y \geq 0 \land x \bmod y = 0$$

*Pair*

$$x : \mathbb{Z}$$
$$x \geq 0 \land x \bmod 2 = 0$$

est équivalent à

$$\forall x, y : \mathbb{Z} \mid x \geq 0 \land y \geq 0 \bullet (x \bmod y = 0 \land y \bmod 2 = 0) \Rightarrow x \bmod 2 = 0$$

# Les schémas

Un schéma décrit un type de données

$cl : Classe$

$cl.effectif\_max(projection)$

dont les instances sont des liaisons $\theta Classe$ d'un type schéma

$\langle\!| effectif\_max : \mathbb{Z}, élèves : \mathbb{P}\ ETUDIANT |\!\rangle$

# Calcul de schémas

Calcul de schémas

Soit

$$\begin{array}{|l} \textit{Etudiant} \\ \hline \textit{nom} : \mathrm{seq}\ \textit{CHAR} \\ \textit{num} : \mathbb{N}_1 \\ \hline \#\textit{nom} \leq 15 \\ \hline \end{array}$$

# Calcul de schémas

*EtudiantAdmin*

$$\begin{array}{|l} \textit{Etudiant} \\ \textit{adresse} : \operatorname{seq} \textit{CHAR} \\ \hat{a}\textit{ge} : \mathbb{N}_1 \\ \hline \hat{a}\textit{ge} \geq 10 \land \hat{a}\textit{ge} \leq 85 \end{array}$$

*EtudiantAdmin*

$$\begin{array}{|l} \textit{nom} : \operatorname{seq} \textit{CHAR} \\ \textit{num} : \mathbb{N}_1 \\ \textit{adresse} : \operatorname{seq} \textit{CHAR} \\ \hat{a}\textit{ge} : \mathbb{N}_1 \\ \hline \#\textit{nom} \leq 15 \land \\ \hat{a}\textit{ge} \geq 10 \land \hat{a}\textit{ge} \leq 85 \end{array}$$

# Calcul de schémas

Une **décoration** est un suffixage de variables par une marque de ponctuation (', ", ?, !).

Par exemple, le schéma $Etudiant'$ est équivalent à :

$$
\begin{array}{|l}
\hline
Etudiant' \\
\hline
nom' : \mathrm{seq}\ CHAR \\
num' : \mathbb{N}_1 \\
\hline
\#nom' \leq 15 \\
\hline
\end{array}
$$

# Calcul de schémas

$$\Delta Etudiant$$
$$Etudiant$$
$$Etudiant'$$

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

# Calcul de schémas

$$\Xi Etudiant$$
$$\Delta Etudiant$$
---
$$\theta Etudiant = \theta Etudiant'$$

$$\Xi Etudiant$$
$$nom' : \operatorname{seq} CHAR$$
$$num' : \mathbb{N}_1$$
$$nom : \operatorname{seq} CHAR$$
$$num : \mathbb{N}_1$$
---
$$\#nom \leq 15 \land \#nom' \leq 15$$
$$nom = nom' \land num = num'$$

# Calcul de schémas

soient S et T des schémas.

- Opérateurs sur les déclarations
  - Projection déclarations : $\mathrm{tuple}\ S$
  - Ajout déclarations $D$ : $S\ ;D$
  - Masquage : $S \setminus (v_1,\ v_2, ..., v_n)$
  - Projection : $S \rceil (v_1,\ v_2, ..., v_n)$
- Opérateurs sur les prédicats
  - Projection prédicat : $\mathrm{pred}\ S$
  - Ajout prédicat $P$ : $S \mid P$

# Calcul de schémas

- Renommage $P$ : $S[new_1/old_1,\ ...,\ new_n/old_n]$

# Calcul de schémas

- Renommage $P$ : $S[new_1/old_1,\ ...,\ new_n/old_n]$
- Opérateurs logiques ($\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) :

# Calcul de schémas

- Renommage $P$ : $S[new_1/old_1,\ ...,\ new_n/old_n]$
- Opérateurs logiques ($\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) :
  - fusion des déclarations

# Calcul de schémas

- Renommage $P$ : $S[new_1/old_1,\ ...,\ new_n/old_n]$
- Opérateurs logiques ($\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) :
  - fusion des déclarations
  - opération logique des prédicats

# Calcul de schémas

- Renommage $P$ : $S[new_1/old_1,\ ...,\ new_n/old_n]$
- Opérateurs logiques ($\land,\ \lor,\ \neg,\ \Rightarrow,\ \Leftrightarrow$) :
  - fusion des déclarations
  - opération logique des prédicats
- Précondition : $\mathrm{pre}\, S \mathrel{\widehat=} (\exists State'; y! : Y \bullet S)$

# Calcul de schémas

- Renommage $P$ : $S[new_1/old_1, \ ..., \ new_n/old_n]$
- Opérateurs logiques ($\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) :
  - fusion des déclarations
  - opération logique des prédicats
- Précondition : $\mathrm{pre}\, S \mathrel{\widehat=} (\exists State'; y! : Y \bullet S)$
- Surcharge : $S \oplus T \mathrel{\widehat=} (S \wedge \neg \mathrm{pre}\, T) \vee T$

# Calcul de schémas

- Renommage $P$ : $S[new_1/old_1, \ ..., \ new_n/old_n]$
- Opérateurs logiques ($\land$, $\lor$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) :
  - fusion des déclarations
  - opération logique des prédicats
- Précondition : $\mathrm{pre}\, S \mathrel{\widehat=} (\exists State'; y! : Y \bullet S)$
- Surcharge : $S \oplus T \mathrel{\widehat=} (S \land \neg \mathrm{pre}\, T) \lor T$
- Composition : $S \fatsemi T$ (application séquentielle)

# Calcul de schémas

- Renommage $P$ : $S[new_1/old_1, \ ..., \ new_n/old_n]$
- Opérateurs logiques ($\land$, $\lor$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) :
  - fusion des déclarations
  - opération logique des prédicats
- Précondition : $\mathrm{pre}\, S \mathrel{\widehat=} (\exists State'; y! : Y \bullet S)$
- Surcharge : $S \oplus T \mathrel{\widehat=} (S \land \neg \mathrm{pre}\, T) \lor T$
- Composition : $S \fatsemi T$ (application séquentielle)
- Tubage : $S \gg T$ (les sorties de l'un entrent dans l'autre)

# Calcul de schémas

- Renommage $P$ : $S[new_1/old_1, \ldots, new_n/old_n]$
- Opérateurs logiques ($\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) :
  - fusion des déclarations
  - opération logique des prédicats
- Précondition : $\mathrm{pre}\, S \mathrel{\widehat=} (\exists State'; y! : Y \bullet S)$
- Surcharge : $S \oplus T \mathrel{\widehat=} (S \wedge \neg \mathrm{pre}\, T) \vee T$
- Composition : $S \fatsemi T$ (application séquentielle)
- Tubage : $S \gg T$ (les sorties de l'un entrent dans l'autre)

++ Détails et exemples : [AV01], chapitre 3, section 2

# En résumé

Z =

Logique des prédicats

+

Théorie des ensembles

# Plan

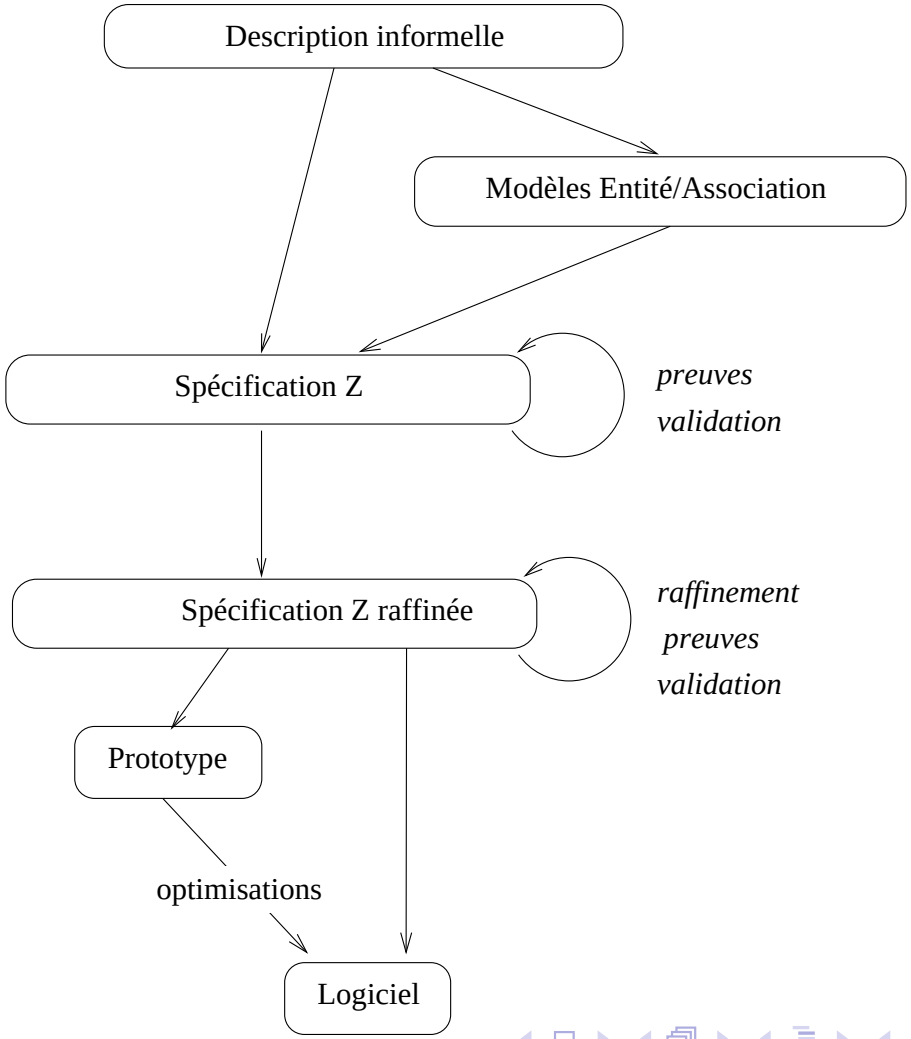
# Démarche de spécification en Z

**Analyse préliminaire**

**Spécification formelle**

**Conception abstraite**

**Réalisation**



# Démarche de spécification en Z

Programmation structurée

=

structure de données

+

procédures

Spécification séquentielle en Z

=

état

+

opérations

# Schémas utilisés

## Schéma d'état

*Etudiant*

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

## Schéma d'opération (décorations)

*ChangeNom*

$$\Delta Etudiant$$
$$nom? : \mathrm{seq}\, CHAR$$
$$\#nom? \leq 15 \land nom' = nom? \land num = num'$$

# Schémas utilisés

Variantes pour l'état initial et les préconditions.

*InitEtudiant*

$$Etudiant'$$

---

$$nom' = \langle\rangle$$

pre *ChangeNom*

$$Etudiant$$
$$nom? : \operatorname{seq} CHAR$$

---

$$\#nom? \leq 15$$

# Les éléments d'une spécification Z (1/2)

- Définitions de types
  - de base
  - libres (énuméré ou union de types)
  - construits par les opérateurs de types
- Déclarations globales
  - Constantes
  - Variables
    - variables globales typées
    - contraintes (définition axiomatique)
  - Opérateurs ou relations par définition axiomatique et/ou génériques
    - relations
    - fonctions
    - opérations

# Les éléments d'une spécification Z (2/2)

- Déclarations de schémas.
  - Etat ou partie de l'état du système.
  - Schéma de l'état initial du système.
  - Opération ou partie d'opérations accédant ou modifiant l'état du système.
  - Schémas des pré-conditions des opérations.
  - Schémas de raffinement.
- Preuves
  - Obligations de preuve (état initial, conservation de l'invariant, pré-conditions).
  - Démontrations de propriétés (théorèmes).
  - Démontrations de validité du raffinement.

# Organisation d'une spécification Z

La spécification Z est organisée de la façon suivante :

1. Les déclarations globales
   - de types de base et de types libres,
   - constantes, variables et fonctions.
2. Les types utilisateur : construits, schémas
3. Un schéma d'état.
4. Un schéma initial.
5. Des schémas d'opérations
6. Des schémas de précondition
7. Des théorèmes et des preuves.
8. Le raffinement.

# Des exemples d'illustration

- Hôpital
  - Spécification de l'état
  - Spécification des opérations
  - Obligations de preuve (état initial, préconditions)
  - Raffinage

++ voir [AV01], chapitre 3

- Facturation
  - Spécification de l'état
  - Spécification des opérations
  - Obligations de preuve (état initial, préconditions)

++ voir [AV01], chapitre 4

# Cas Hôpital - Etat

$[PATIENT]$

$MAX \mathrel{\widehat=} 20$

*SalleAttenteHôpital*

$$patients : \operatorname{seq} PATIENT$$

---

$$\#patients \leq MAX$$

$$\forall i, j : 1 \mathinner{..} \#patients \mid i \neq j \bullet patients(i) \neq patients(j)$$

# Cas Hôpital - Etat

Etat initial

$$\begin{array}{|l} \textit{InitSalleAttenteHôpital} \\ \textit{SalleAttenteHôpital} \\ \hline patients = \langle\rangle \end{array}$$

Preuve de l'existence d'un état initial.

$$\exists \textit{SalleAttenteHôpital} \bullet \textit{InitSalleAttenteHôpital}$$

# Cas Hôpital - Etat

## Preuve état initial

$$
\begin{array}{lll}
0 & \exists patients' : \mathrm{seq}\, PATIENT \bullet \#patients' \leq MAX & \\
& \wedge\, (\forall i, j : 1 \,..\, \#patients' \mid i \neq j \bullet & \\
& \qquad patients'(i) \neq patients'(j)) \wedge patients' = \langle\rangle & \\
1 & \#\langle\rangle \leq MAX \wedge (\forall i, j : 1 \,..\, \#\langle\rangle \mid i \neq j \bullet \langle\rangle(i) \neq \langle\rangle(j)) & \\
2 & 0 \leq MAX \wedge (\forall i, j : 1 \,..\, 0 \mid i \neq j \bullet \langle\rangle(i) \neq \langle\rangle(j)) & \\
3 & 0 \leq MAX & [\wedge\text{-élimination}] \\
3.1 & \quad 0 \leq 20 & [\text{par substitution de } MAX] \\
3.2 & \quad true & [\text{se déduit par } \leq] \\
4 & \forall i, j : 1 \,..\, 0 \mid i \neq j \bullet \langle\rangle(i) \neq \langle\rangle(j) & [\wedge\text{-élimination}] \\
4.1 & \quad true & [1 \,..\, 0 \text{ est l'ensemble vide, tautologie}] \\
5 & true & [\wedge\text{-introduction}]
\end{array}
$$

# Cas Hôpital - Opérations en mise-à-jour

*Arrivée*

$\Delta SalleAttenteHôpital$
$p? : PATIENT$

---

$patients' = patients \frown \langle p? \rangle$
$\#patients < MAX \land \forall i : 1 \,..\, \#patients \bullet patients(i) \neq p?$

*Sortie*

$\Delta SalleAttenteHôpital$
$p! : PATIENT$

---

$patients = \langle p! \rangle \frown patients'$
$patients \neq \langle \rangle$

# Cas Hôpital - Opérations en consultation

*EnAttente*

$\Xi SalleAttenteHôpital$
$n! : \mathbb{N}$

---

$n! = \#patients$

*Sorties*

$\Delta SalleAttenteHôpital$
$p! : PATIENT$
$n? : \mathbb{N}_1$

---

$n? \leq \#patients$

$\exists sa_1 : \operatorname{seq} PATIENT \mid \#sa_1 = n? - 1 \bullet$
$patients = sa_1 \frown \langle p! \rangle \frown patients'$

# Cas Hôpital - précondition

Préservation de l'invariant dans les opérations.

$$\exists SalleAttenteHôpital'; sorties! \bullet SchémaOp.$$

Application à l'opération *Arrivée*

$$\mathrm{pre}\, Arrivée \mathrel{\widehat=} \exists SalleAttenteHôpital' \bullet Arrivée$$

$\mathrm{pre}\, Arrivée$

$$SalleAttenteHôpital$$
$$p? : PATIENT$$

---

$$\exists SalleAttenteHôpital' \bullet (patients' = patients \frown \langle p? \rangle \land$$
$$\#patients < MAX \land (\forall i : 1 \mathinner{..} \#patients \bullet patients(i) \neq p?))$$

# Cas Hôpital - preuve de la précondition

La démonstration se fait en remplaçant les schémas par leurs déclarations et leurs prédicats.

$$\begin{array}{|l}
\hline
\mathrm{pre}\ \mathit{Arrivée} \\
\mathit{SalleAttenteHôpital} \\
p? : \mathit{PATIENT} \\
\hline
\exists \mathit{patients}' : \mathrm{seq}\ \mathit{PATIENT} \bullet (\#\mathit{patients}' \leq \mathit{MAX} \land \\
(\forall i, j : 1 \mathinner{..} \#\mathit{patients}' \mid i \neq j \bullet \mathit{patients}'(i) \neq \mathit{patients}'(j))) \land \\
(\mathit{patients}' = \mathit{patients} \frown \langle p? \rangle \land \\
\#\mathit{patients} < \mathit{MAX} \land (\forall i : 1 \mathinner{..} \#\mathit{patients} \bullet \mathit{patients}(i) \neq p?)) \\
\hline
\end{array}$$

# Cas Hôpital - preuve de la précondition

Puis en substituant progressivement les variables décorées par ’ par des valeurs non décorées par ’ et les sorties par leur valeur.

pre *Arrivée*

$$
\begin{array}{l}
SalleAttenteHôpital \\
p? : PATIENT \\
\hline
(\#(patients \frown \langle p? \rangle) \leq MAX \land \\
(\forall i, j : 1 \,..\, \#(patients \frown \langle p? \rangle) \mid i \neq j \bullet \\
\quad (patients \frown \langle p? \rangle)(i) \neq (patients \frown \langle p? \rangle)(j))) \land \\
\#patients < MAX \land (\forall i : 1 \,..\, \#patients \bullet patients(i) \neq p?)
\end{array}
$$

# Cas Hôpital - preuve de la précondition

Par distributivité de $\#$ sur $\frown$ le schéma se simplifie en

$\mathrm{pre}\ Arrivée$

$SalleAttenteHôpital$

$p? : PATIENT$

---

$$((\#patients + \#\langle p? \rangle) \leq MAX \land$$
$$(\forall i, j : 1 \,..\, (\#patients + \#\langle p? \rangle) \mid i \neq j \bullet$$
$$(patients \frown \langle p? \rangle)(i) \neq (patients \frown \langle p? \rangle)(j))) \land$$
$$\#patients < MAX \land (\forall i : 1 \,..\, \#patients \bullet patients(i) \neq p?)$$

# Cas Hôpital - preuve de la précondition

Par définition de #, nous avons

$$
\begin{array}{|l}
\hline
\mathrm{pre}\ \mathit{Arrivée} \\
\mathit{SalleAttenteHôpital} \\
p? : \mathit{PATIENT} \\
\hline
((\#\mathit{patients} + 1) \leq \mathit{MAX} \land \\
(\forall i, j : 1 \,..\, (\#\mathit{patients} + 1) \mid i \neq j \bullet \\
\quad (\mathit{patients} \frown \langle p? \rangle)(i) \neq (\mathit{patients} \frown \langle p? \rangle)(j))) \land \\
\#\mathit{patients} < \mathit{MAX} \land (\forall i : 1 \,..\, \#\mathit{patients} \bullet \mathit{patients}(i) \neq p?) \\
\hline
\end{array}
$$

# Cas Hôpital - preuve de la précondition

Par application de $-1$ sur chaque membre de l'égalité, et par commutativité de $\wedge$, nous obtenons

$$
\begin{array}{|l}
\hline
\mathrm{pre}\ \mathit{Arrivée} \\
\mathit{SalleAttenteHôpital} \\
p? : \mathit{PATIENT} \\
\hline
\#\mathit{patients} \leq \mathit{MAX} - 1 \wedge \#\mathit{patients} < \mathit{MAX} \wedge \\
(\forall i, j : 1 \,..\, (\#\mathit{patients} + 1) \mid i \neq j \bullet \\
\qquad (\mathit{patients} \frown \langle p? \rangle)(i) \neq (\mathit{patients} \frown \langle p? \rangle)(j)) \wedge \\
(\forall i : 1 \,..\, \#\mathit{patients} \bullet \mathit{patients}(i) \neq p?) \\
\hline
\end{array}
$$

# Cas Hôpital - preuve de la précondition

Par définition de $\leq$ et application de $P_{27}$, nous simplifions le schéma en

$\mathrm{pre}\ Arrivée$

$$SalleAttenteHôpital$$
$$p? : PATIENT$$

---

$$\#patients < MAX \land$$
$$(\forall i, j : 1 \,..\, (\#patients + 1) \mid i \neq j \bullet$$
$$\quad (patients \frown \langle p? \rangle)(i) \neq (patients \frown \langle p? \rangle)(j)) \land$$
$$(\forall i : 1 \,..\, \#patients \bullet patients(i) \neq p?)$$

# Cas Hôpital - preuve de la précondition

Nous nous intéressons maintenant à la propriété d'unicité du patient dans la salle. Pour cela nous avons besoin de deux lemmes.

`lemme 1` (ordre)
*Les premiers termes d'une concaténation de séquence sont ceux de la première séquence :*
$\forall s, t : \mathrm{seq}\, X \bullet \forall i : 1 \mathinner{..} \# s \bullet (s \frown t) i = s\ i$

`lemme 2`
*Un prédicat lié à un intervalle peut être découpé en une conjonction du prédicat sur des sous-intervalles :*
$\forall x : 1 \mathinner{..} n+1 \bullet P(x) \Leftrightarrow ((\forall x : 1 \mathinner{..} n \bullet P(x)) \land P(n+1))$

# Cas Hôpital - preuve de la précondition

Ce dernier point se simplifie dans le schéma de précondition.

$\mathrm{pre}\ \mathit{Arrivée}$

$$\mathit{SalleAttenteHôpital}$$
$$p? : \mathit{PATIENT}$$

---

$$\#\mathit{patients} < \mathit{MAX} \land (\forall i : 1 \,..\, \#\mathit{patients} \bullet \mathit{patients}(i) \neq p?)$$

# Cas Hôpital - preuve de la précondition

Ce dernier point se simplifie dans le schéma de précondition.

$\mathrm{pre}\ \mathit{Arrivée}$

$\mathit{SalleAttenteHôpital}$

$p? : \mathit{PATIENT}$

---

$\#\mathit{patients} < \mathit{MAX} \land (\forall i : 1 \,..\, \#\mathit{patients} \bullet \mathit{patients}(i) \neq p?)$

$\Rightarrow$ utilité des outils !

# Cas Hôpital - Robustesse

La précondition d'une opération robuste est `true`.

$\Rightarrow$ prévoir les cas d'erreur

Une manière de procéder consiste à étudier chaque cas normal et chaque cas d'erreur.

$$MESSAGE ::= Autorisé \mid Refusé$$

$$MessagePossible \mathrel{\widehat=} [msg! : MESSAGE \mid msg! = Autorisé]$$

$$ArrivéeRobuste \mathrel{\widehat=} (Arrivée \land MessagePossible) \lor ArrivéeImpossible$$

# Cas Hôpital - Robustesse

*ArrivéeImpossible*

$\Xi SalleAttenteHôpital$

$p? : PATIENT$

$msg! : MESSAGE$

---

$msg! = Refusé$

$\#patients = MAX \lor \exists i : 1 \,..\, \#patients \bullet patients(i) = p?$

Exercice : vérifier que la précondition est vraie.

# Cas Hôpital - Raffinage

Deux types de raffinage sont distingués en Z :

- Le raffinage des opérations. Les structures de contrôle du langage de programmation cible sont introduites progressivement, *e.g.*séquences, conditionnelles, itérations pour la programmation impérative.
- Le raffinage des données. Les structures de données du langage de programmation cible sont introduites progressivement, *e.g.*structures, tableaux, pointeurs pour la programmation impérative.

# Cas Hôpital - Raffinage de données

La file est "implantée" par un tableau avec des pointeurs.

En Z, un tableau se représente par une séquence (!!).

**Déclaration** $file : Array[1 \,..\, MAX] ofPATIENT$

$file : \mathrm{seq}\, PATIENT \bullet \# file \leq MAX$

**Lecture** $file[i],\ \forall i : 1 \,..\, MAX$

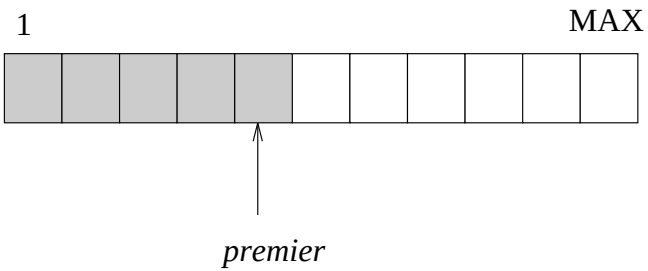
$\forall i : 1 \,..\, MAX \bullet file\ i$

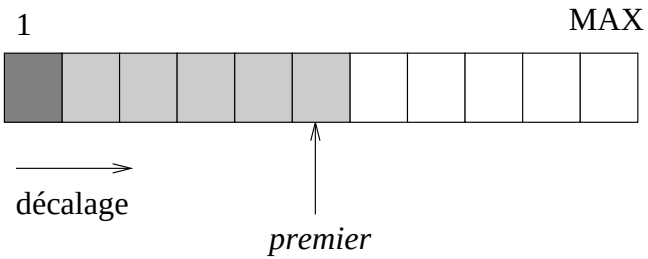
**Affectation** $file[i] := p,\ \forall i : 1 \,..\, MAX; p : PATIENT$

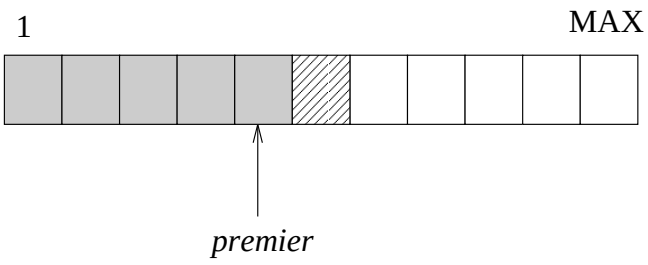
$\forall i : 1 \,..\, MAX \bullet file' = file \oplus \{i \mapsto p\}$

# Cas Hôpital - Raffinage

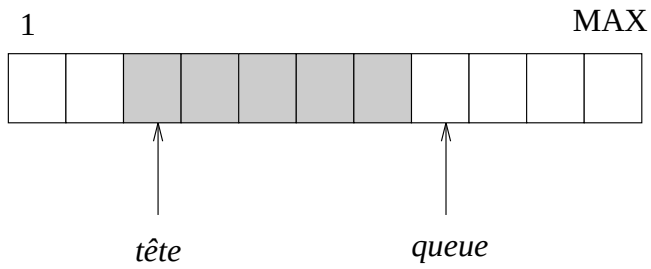
**un pointeur**



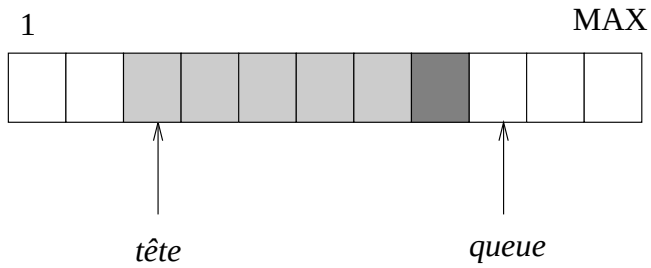



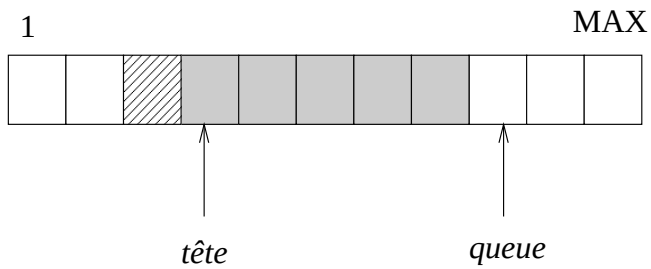

**deux pointeurs**




Ajout




Retrait



# Cas Hôpital - Raffinage

*SalleAttenteHôpital*

$$patients : \mathrm{seq}\ PATIENT$$

---

$$\#patients \leq MAX$$

$$\forall i, j : 1 \,..\, \#patients \mid i \neq j \bullet patients(i) \neq patients(j)$$

*SalleConcret*

$$file : \mathrm{seq}\ PATIENT$$

$$premier : \mathbb{N}$$

---

$$\#file \leq MAX \land premier \leq MAX$$

$$\forall i, j : 1 \,..\, premier \mid i \neq j \bullet file(i) \neq file(j)$$

# Cas Hôpital - Raffinage

*Abs*

*SalleAttenteHôpital*
*SalleConcret*

$$patients = \mathrm{rev}\ ((1\,..\,premier) \lhd file)$$

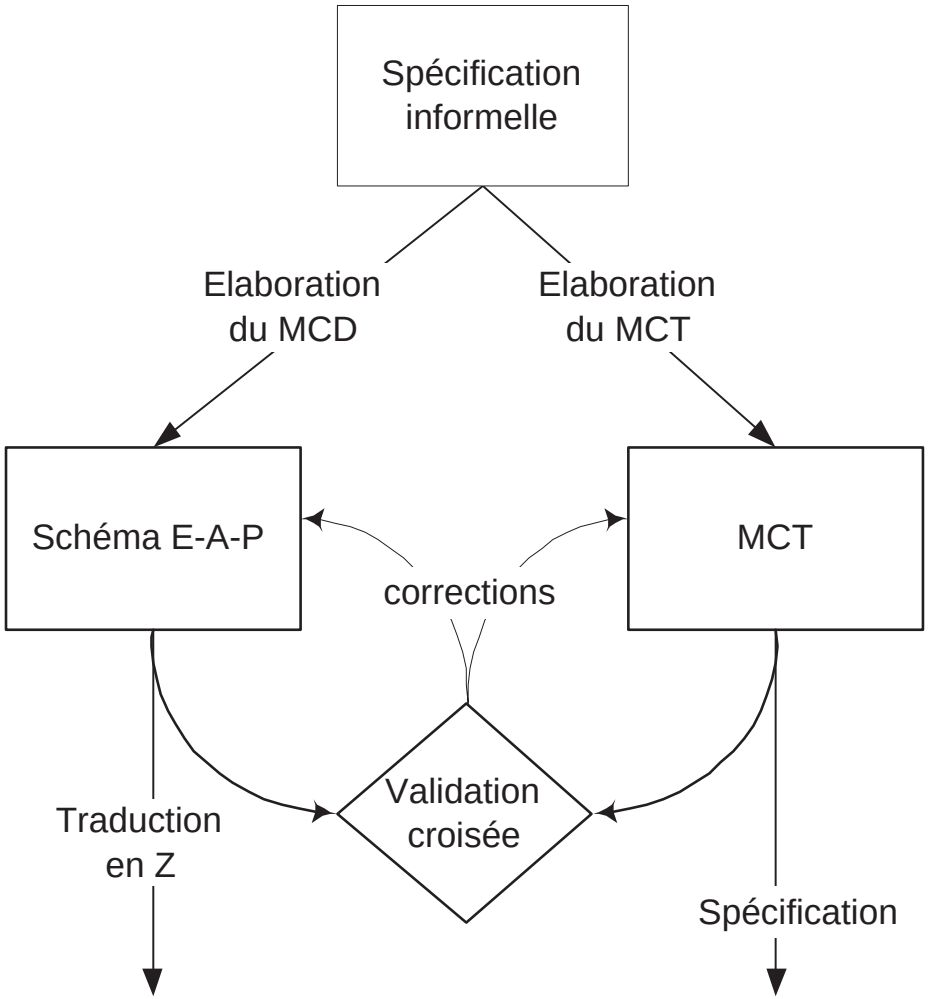
idem pour les opérations

La suite de raffinage établit l'implantation (prouvée) finale.

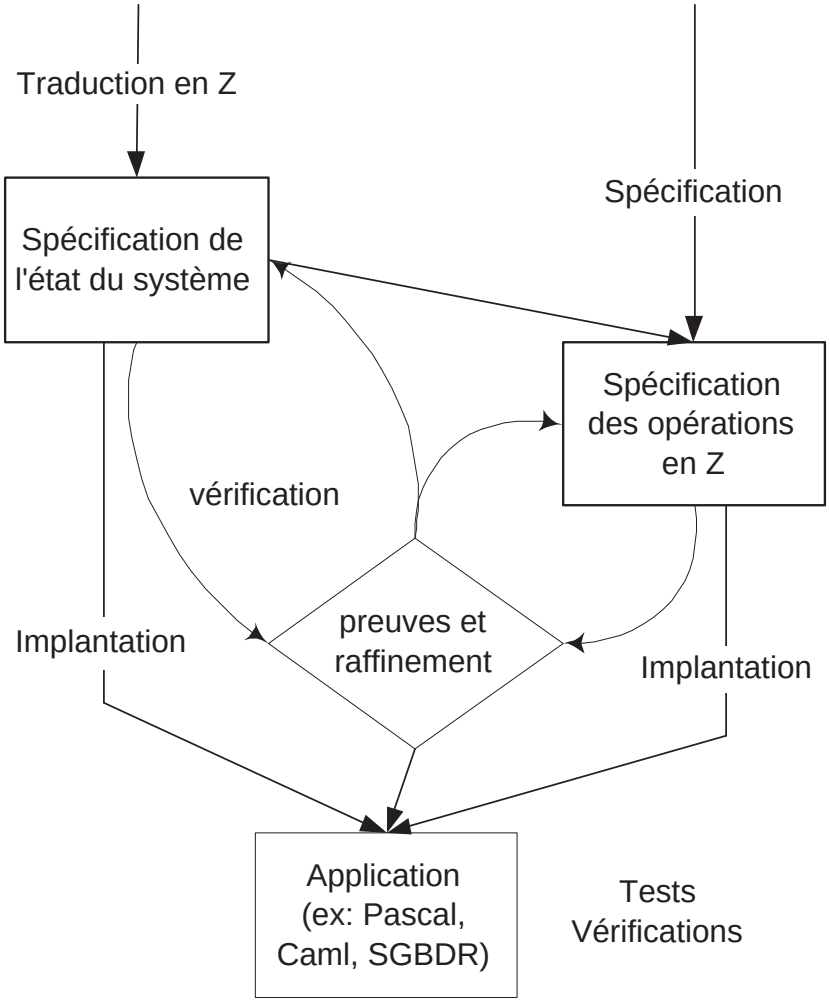
Preuves et détails : voir [AV01], chapitre 3, section 7

# Plan

# Retour sur la démarche de spécification 1/2



# Retour sur la démarche de spécification 2/2



# Traduction du formalisme E-A-P en Z

Plusieurs variantes sont possibles, seule la traduction principale est présentée ici.

- Traduction d'un type d'entité
- Traduction d'un type d'association
- Optimisations

$\Rightarrow$ Détails et exemples : [AV01], chapitre 5, section 2

# Traduction d'un type d'entité en Z

Un TE = un schéma Z pour les données non clé + une fonction

| **Patient** |
| --- |
| no_SecSoc_p |
| nom_p |
| prénom_p |
| adresse_p |

*PatientT2*

$nom\_p : STRING$

$prénom\_p : STRING$

$adresse\_p : STRING$

*PatientExt2*

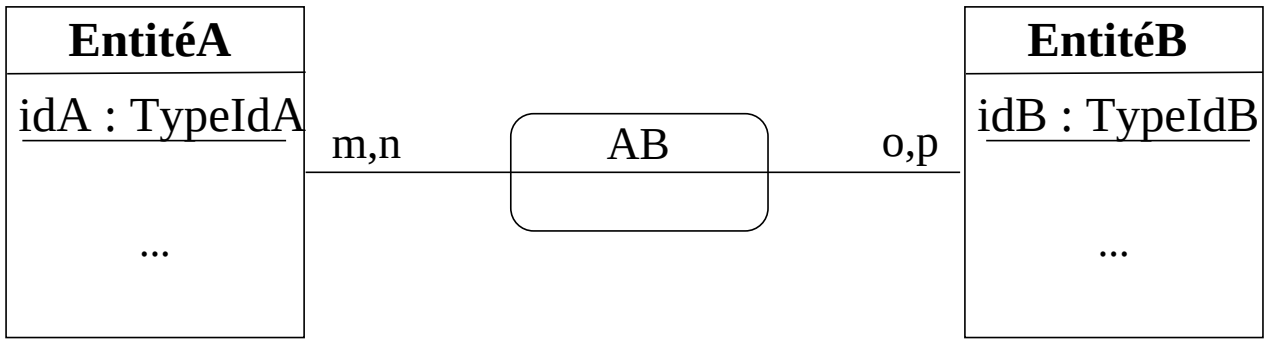
$no\_SecSoc\_p : \mathbb{N} \pfun PatientT2$

identifiant du TE

# Traduction d'un TA en Z

- Identifiant $\Rightarrow$ inclusion des extensions de TE
- Porteur d'informations $\Rightarrow$ représentation similaire au TE pour le TA
  - une seule propriété : pas de TA explicite
  - plusieurs propriétés
- Arité
  - binaire : relation Z ou variante
  - n-aire ($n > 2$) : produit cartésien
- Cardinalités : prédicats, simplifications possibles pour les TA binaires
- Association d'association $\Rightarrow$ personnaliser d'abord puis cas normal

# Traduction d'un TA binaire non porteur en Z




*ABExt*

$$
\begin{array}{l}
\mathit{EntitéAExt} \\
\mathit{EntitéBExt} \\
\\
\mathit{assoc}_{AB} : \mathit{TypeIdA} \leftrightarrow_{gen} \mathit{TypeIdB} \\
\hline
\mathrm{dom}\ \mathit{assoc}_{AB} \subseteq \mathrm{dom}\ \mathit{idA} \land \mathrm{ran}\ \mathit{assoc}_{AB} \subseteq \mathrm{dom}\ \mathit{idB}
\end{array}
$$

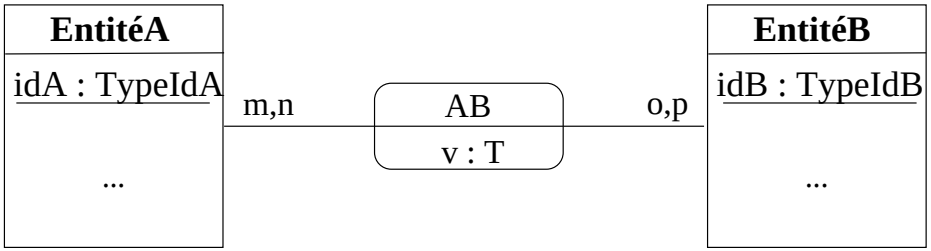
# Traduction d'un TA binaire (cardinalités) 1/2

| **cas** | **m** | **n** | **o** | **p** | $assoc_{AB} : didA \leftrightarrow_{gen} didB$ |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | $assoc_{AB} : didA \rightarrowtail didB$ |
| 2 | 0 | 1 | 1 | 1 | $assoc_{AB} : didA \rightarrowtail\!\!\!\!\rightarrow didB$ |
| 3 | 0 | 1 | 0 | n | $assoc_{AB} : didA \nrightarrow didB$ |
| 4 | 0 | 1 | 1 | n | $assoc_{AB} : didA \nrightarrow\!\!\!\!\rightarrow didB$ |
| 5 | 0 | n | 0 | 1 | $assoc_{AB} : didB \nrightarrow didA$ |
| 6 | 0 | n | 1 | 1 | $assoc_{AB} : didB \rightarrow didA$ |
| 7 | 0 | n | 0 | n | $assoc_{AB} : didA \leftrightarrow didB$ |
| 8 | 0 | n | 1 | n | $assoc_{AB} : didA \leftrightarrow didB$ |

# Traduction d'un TA binaire (cardinalités) 2/2

| **cas** | **m** | **n** | **o** | **p** | $assoc_{AB} : didA \leftrightarrow_{gen} didB$ |
|---|---|---|---|---|---|
| 9 | 1 | 1 | 0 | 1 | $assoc_{AB} : didA \rightarrowtail didB$ |
| 10 | 1 | 1 | 1 | 1 | $assoc_{AB} : didA \rightarrowtail\!\!\!\!\rightarrow didB$ |
| 11 | 1 | 1 | 0 | n | $assoc_{AB} : didA \rightarrow didB$ |
| 12 | 1 | 1 | 1 | n | $assoc_{AB} : didA \rightarrow\!\!\!\!\rightarrow didB$ |
| 13 | 1 | n | 0 | 1 | $assoc_{AB} : didB \leftarrow\!\!\!\!\leftarrow didA$ |
| 14 | 1 | n | 1 | 1 | $assoc_{AB} : didB \rightarrow\!\!\!\!\rightarrow didA$ |
| 15 | 1 | n | 0 | n | $assoc_{AB} : didA \leftrightarrow didB$ |
| 16 | 1 | n | 1 | n | $assoc_{AB} : didA \leftrightarrow didB$ |

# Traduction d'un TA binaire porteur en Z




*ABExt*

$$\begin{array}{l} \mathit{EntitéAExt} \\ \mathit{EntitéBExt} \\ \\ \mathit{assoc}_{AB} : (\mathit{TypeIdA} \leftrightarrow_{gen} \mathit{TypeIdB}) \\ \mathit{prop}_{AB} : (\mathit{TypeIdA} \times \mathit{TypeIdB}) \pfun \mathit{ABT} \\ \hline \\ \mathrm{dom}\ \mathit{assoc}_{AB} \subseteq \mathrm{dom}\ \mathit{idA} \land \mathrm{ran}\ \mathit{assoc}_{AB} \subseteq \mathrm{dom}\ \mathit{idB} \\ \\ \mathrm{dom}\ \mathit{prop}_{AB} = \mathit{assoc}_{AB} \end{array}$$

# Traduction E-A-P en Z (exemple)

$[STRING]$

*PatientT*

$nom\_p : STRING$
$prénom\_p : STRING$
$adresse\_p : STRING$

*PatientExt*

$no\_SecSoc\_p : \mathbb{N} \pfun PatientT$

# Traduction E-A-P en Z (exemple)

*SalleT*

$$nom\_s : STRING$$
$$localisation\_s : STRING$$
$$nb\_places\_s : \mathbb{N}$$

*SalleExt*

$$no\_s : \mathbb{N} \pfun SalleT$$

# Traduction E-A-P en Z

*Attente_normaleT*

$num\_ordre\_an : \mathbb{N}$

*Attente_normaleExt*

*PatientExt*

*SalleExt*

$attente\_normale : \mathbb{N} \pfun \mathbb{N}$

$prop\_attente\_normale : (\mathbb{N} \times \mathbb{N}) \pfun Attente\_normaleT$

---

$\mathrm{dom}\ attente\_normale \subseteq \mathrm{dom}\ no\_SecSoc\_p \land$

$\mathrm{ran}\ attente\_normale \subseteq \mathrm{dom}\ no\_s \land$

$\mathrm{dom}\ prop\_attente\_normale = attente\_normale$

Il s'agit bien d'une fonction partielle $no\_SecSoc\_p \pfun no\_s$.

# Traduction E-A-P en Z (exemple)

*UrgenceT*

$num\_ordre\_u : \mathbb{N}$

*UrgenceExt*

*PatientExt*

*SalleExt*

$urgence : \mathbb{N} \pfun \mathbb{N}$

$prop\_urgence : (\mathbb{N} \times \mathbb{N}) \pfun UrgenceT$

---

$\dom urgence \subseteq \dom no\_SecSoc\_p \land$

$\ran urgence \subseteq \dom no\_s \land$

$\dom prop\_urgence = urgence$

Il s'agit bien d'une fonction partielle $no\_SecSoc\_p \pfun no\_s$.

# Traduction E-A-P en Z (exemple)

Système global

*Hôpital*

*Attente_normaleExt*

*UrgenceExt*

$$\operatorname{dom} attente\_normale \cap \operatorname{dom} urgence = \emptyset$$

# Traduction E-A-P en Z (optimisations)

Les optimisations suivantes peuvent être appliquées.

- Suppression des schémas mono-variable

# Traduction E-A-P en Z (optimisations)

Les optimisations suivantes peuvent être appliquées.

- Suppression des schémas mono-variable
  - substitution du schéma par sa variable

# Traduction E-A-P en Z (optimisations)

Les optimisations suivantes peuvent être appliquées.

- Suppression des schémas mono-variable
  - substitution du schéma par sa variable
  - exemple : *PatientExt*, *Attente_normaleT*

# Traduction E-A-P en Z (optimisations)

Les optimisations suivantes peuvent être appliquées.

- Suppression des schémas mono-variable
  - substitution du schéma par sa variable
  - exemple : *PatientExt*, *Attente_normaleT*
  - et pour les TE renommage explicite par le nom du TE sans majuscule et finissant par 's'

# Traduction E-A-P en Z (optimisations)

Les optimisations suivantes peuvent être appliquées.

- Suppression des schémas mono-variable
  - substitution du schéma par sa variable
  - exemple : *PatientExt*, *Attente_normaleT*
  - et pour les TE renommage explicite par le nom du TE sans majuscule et finissant par 's'
  - exemple : $no\_SecSoc\_p \Rightarrow patients$

# Traduction E-A-P en Z (optimisations)

Les optimisations suivantes peuvent être appliquées.

- Suppression des schémas mono-variable
  - substitution du schéma par sa variable
  - exemple : *PatientExt*, *Attente_normaleT*
  - et pour les TE renommage explicite par le nom du TE sans majuscule et finissant par 's'
  - exemple : $no\_SecSoc\_p \Rightarrow patients$
- Suppression des schémas types

# Traduction E-A-P en Z (optimisations)

Les optimisations suivantes peuvent être appliquées.

- Suppression des schémas mono-variable
  - substitution du schéma par sa variable
  - exemple : *PatientExt*, *Attente_normaleT*
  - et pour les TE renommage explicite par le nom du TE sans majuscule et finissant par 's'
  - exemple : $no\_SecSoc\_p \Rightarrow patients$
- Suppression des schémas types
  - généralisation du cas précédent à plusieurs variables

# Traduction E-A-P en Z (optimisations)

Les optimisations suivantes peuvent être appliquées.

- Suppression des schémas mono-variable
  - substitution du schéma par sa variable
  - exemple : *PatientExt*, *Attente_normaleT*
  - et pour les TE renommage explicite par le nom du TE sans majuscule et finissant par 's'
  - exemple : $no\_SecSoc\_p \Rightarrow patients$
- Suppression des schémas types
  - généralisation du cas précédent à plusieurs variables
  - à éviter : conserver la structure

# Traduction E-A-P en Z (optimisations)

Les optimisations suivantes peuvent être appliquées.

- Suppression des schémas mono-variable
  - substitution du schéma par sa variable
  - exemple : *PatientExt*, *Attente_normaleT*
  - et pour les TE renommage explicite par le nom du TE sans majuscule et finissant par 's'
  - exemple : $no\_SecSoc\_p \Rightarrow patients$
- Suppression des schémas types
  - généralisation du cas précédent à plusieurs variables
  - à éviter : conserver la structure

! Le critère majeur est la lisibilité attendue.
On tient aussi compte de la réutilisabilité.

# Traduction E-A-P en Z (optimisations)

Les types *STRING*, *PatientT* et *SalleT* sont inchangés.

*Hôpital*

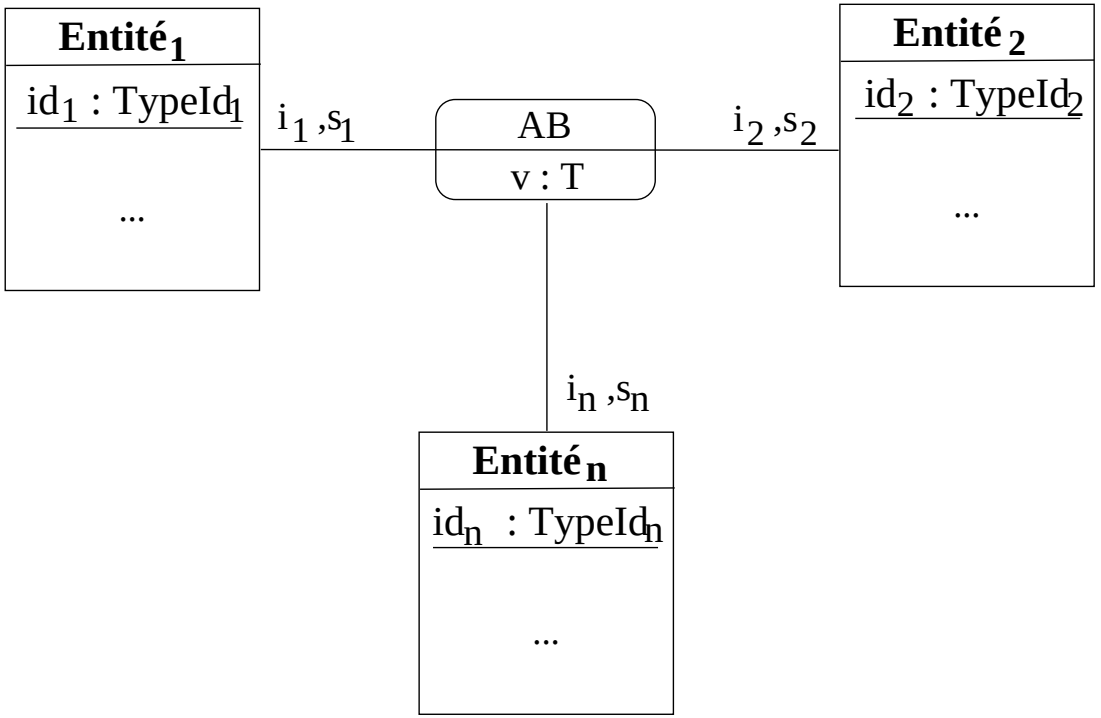
$$
\begin{array}{l}
patients : \mathbb{N} \pfun PatientT \\
salles : \mathbb{N} \pfun SalleT \\
attente\_normale : \mathbb{N} \pfun \mathbb{N} \\
prop\_attente\_normale : (\mathbb{N} \times \mathbb{N}) \pfun Attente\_normaleT \\
urgence : \mathbb{N} \pfun \mathbb{N} \\
prop\_urgence : (\mathbb{N} \times \mathbb{N}) \pfun UrgenceT \\
\hline
\dom attente\_normale \subseteq \dom patients \land \\
\ran attente\_normale \subseteq \dom salles \land \\
\dom prop\_attente\_normale = attente\_normale \land \\
\dom urgence \subseteq \dom patients \land \\
\ran urgence \subseteq \dom salles \land \\
\dom prop\_urgence = urgence \land \\
\dom attente\_normale \cap \dom urgence = \emptyset
\end{array}
$$

La représentation est condensée (plus "appréhendable") mais on a perdu la

# Traduction du formalisme E-A-P en Z

L'ajout de nouvelles contraintes, non formalisées directement en E-A-P, peut être difficile, on doit alors changer de représentation (voir exercices TD).

# Traduction d'un TA n-aire ($n > 2$) en Z




$$ABT$$
$$v : T$$

# Traduction d'un TA n-aire ($n > 2$) en Z

Solution retenue : personnaliser $\Rightarrow$ associations binaires
(l'identifiant est un numéro quelconque)

*ABExt*

$Entité_1 Ext$
$Entité_2 Ext$
...
$Entité_n Ext$
$Entité_{AB} : \mathbb{N} \pfun ABT$

$assoc_{AB_1} : \mathbb{N} \pfun^* TypeId_1$
$assoc_{AB_2} : \mathbb{N} \pfun^* TypeId_2$
...
$assoc_{AB_n} : \mathbb{N} \pfun^* TypeId_n$

---

...

# Traduction d'un TA n-aire ($n > 2$) en Z

## Solution Personnalisation (suite)

*ABExt*

...

(1) $\text{ran } assoc_{AB_1} \subseteq \text{dom } id_1 \wedge$

(2) $\text{dom } assoc_{AB_1} = \text{dom } assoc_{AB_2} \wedge$
$\text{ran } assoc_{AB_2} \subseteq \text{dom } id_2 \wedge$

...

($n$) $\text{dom } assoc_{AB_1} = \text{dom } assoc_{AB_n} \wedge$
$\text{ran } assoc_{AB_n} \subseteq \text{dom } id_n$

+*cardinalités*

* s'affine pour simplifier les cardinalités.

# Traduction du formalisme E-A-P en Z

Autre exemple : la bibliothèque

voir [AV01], chapitre 5

# Plan

# Bilan et extensions

- Bilan
  1. Bon complément d'une modélisation semi-formelle
  2. Documentation rigoureuse
  3. Preuve de programmes
  4. OCL
- Extensions
  1. Outils
  2. Spécification défensive ou offensive
  3. Promotion
  4. Orientation objet