❸ La spécification formelle

- La spécification formelle
- Spécification algébrique

La spécification formelle

- La spécification formelle est une spécification exprimée dans un langage formel. Sa syntaxe et sa sémantique sont définies de manière formelle.
- La spécification formelle est complémentaire de la spécification informelle.
  - ➔ précise et non ambiguë
  - ➔ difficilement lisible et compréhensible par un non spécialiste
  - ➔ vérifiable
  - ➔ possibilité de traitement automatique
- Exemples de techniques de spécification formelle : spécification algébrique, langage de spécification Z.
- La spécification formelle est née dans l'objectif de vérifier formellement les programmes : preuve de programmes.

La spécification formelle

- Les méthodes de spécification formelle reposent sur l'utilisation de prédicats qui ont des assertions sur l'état du système. Un système peut être représenté par un ensemble de fonctions où chaque fonction est spécifiée en utilisant :
  - des pré-conditions : qui spécifient les valeurs en entrée de la fonction
  - des post-conditions : qui spécifient les valeurs en sortie de la fonction
- Exemple :

  Fonction de recherche dans un tableau d'entiers

  fonction Recherche (X : dans Tableau_d_entier ; Clé : dans Entier) retour Entier ;
  Pré : il existe i dans **X'PREMIER**…**X'DERNIER** tel que X(i) = Clé
  Post : **X''** (Recherche (X, Clé)) = Clé et X = X''
  Erreur : Recherche (X,Clé) = X'DERNIER + 1

Le prédicat d'erreur détermine la post-condition à appliquer dans le cas où la pré-condition est fausse

valeur du tableau, une fois la fonction évaluée

bornes du tableau

La spécification algébrique

- La spécification algébrique est une technique qui permet de définir une classe d'objets ou un type, en fonction des relations existant entre les différentes opérations applicables à ce type.

- Utilisée pour la spécification des types abstraits de données, cette technique a ensuite été généralisée à la spécification de systèmes.

La spécification de types abstraits de données

- Une spécification de type abstrait décrit une classe de structure de données non par son implémentation, mais par une liste de services disponibles et par les propriétés formelles de ces services.
- La spécification formelle d'un type de données abstrait consiste en 4 parties
  - types
  - fonctions

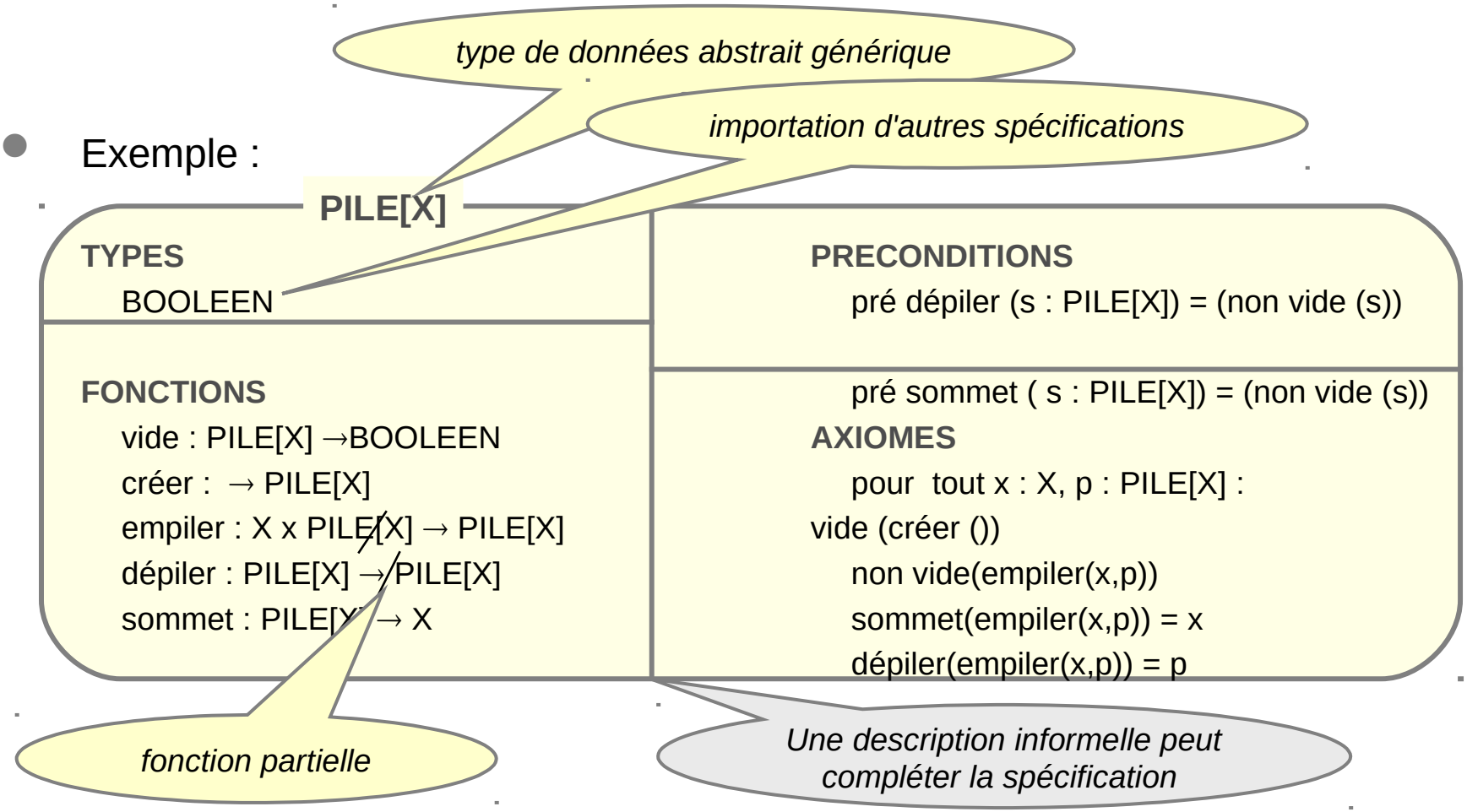
  } **Syntaxe du type**

  - pré-conditions
  - axiomes

  } **Sémantique du type**

La spécification de types abstraits de données

- Exemple :



La spécification de types abstraits de données

- Les fonctions partielles sont des fonctions qui ne sont pas définies pour tout objet du type abstrait

  Exemples : sommet et dépiler

  ➔ définition de pré-conditions

- Il existe 3 catégories de fonctions
  - constructeur
    de type [→ T]
    exemple : créer
  - accesseur
    de type [T →]
    exemples : vide et sommet
  - transformateur
    de type [T → T]
    exemples : empiler et dépiler

La spécification de types abstraits de données

- Des suites d'opérations complexes peuvent être décrites par des expressions mathématiques qui jouissent des propriétés habituelles de l'algèbre
- Le processus d'exécution d'un programme peut être considéré comme un cas de simplification algébrique.

  Exemple : Les axiomes des piles permettent la simplification d'une expression telle que :
  x = sommet (dépiler (empiler (x1, dépiler (empiler (x2, empiler (sommet (dépiler (empiler (x4, empiler (x5, créer()))), dépiler (empiler (x6, empiler (x7, empiler (x8, créer()))))))))

  ➔ **...**

  ➔ x = x5

La spécification de types abstraits de données

- L'enrichissement est une technique clé qui consiste à dériver les spécifications afin que les spécifications simples servent de base pour construire des spécifications plus complexes.

- Le type X enrichit le type Y :
  - il hérite des opérations et axiomes définis sur le type de base Y, de telle sorte qu'ils s'appliquent aussi à X
  - il se peut que de nouvelles opérations de X viennent supplanter des opérations homonymes de Y
  - il se peut qu'il ajoute de nouvelles opérations et de nouveaux axiomes
  - il peut y avoir une partie restrictive qui détermine les opérations qui ne sont pas héritées

La spécification de types abstraits de données

- Spécification d'erreur :

On peut définir une fonction constante Indéfini et dans les cas exceptionnels, on retourne un résultat Indéfini, exemple :

TABLEAU[X: (Indéfini() → X)]

TYPES

ENTIER

FONCTIONS

créer : ENTIER x ENTIER → TABLEAU[X]

affecter : TABLEAU[X] x ENTIER x X → TABLEAU[X]

inf : TABLEAU[X] → ENTIER

sup : TABLEAU[X] → ENTIER

eval : TABLEAU[X] x ENTIER → X

AXIOMES pour tout x, y, i, j : ENTIER, t : TABLEAU[X], v : X

inf (créer(x,y)) = x

sup (créer(x,y)) = y

inf(affecter (t,i,v)) = inf(t)

sup(affecter (t,i,v)) = sup(t)

eval (créer (x, y), i) = Indéfini

eval (affecte (t,i,v), j) = si j < inf(t) ou j > sup(t) alors Indéfini sinon si j = i alors v sinon eval(t,j)

Spécification algébrique d'un problème de facturation de commandes

Un problème de facturation de commandes

- L'objectif est de facturer des commandes.
- Facturer consiste à changer l’état d’une commande (de "en_attente" à "facturée")
- Sur une commande, il y a une seule référence à un produit commandé avec une certaine quantité. La quantité peut varier selon les commandes.
- La même référence peut être commandée sur différentes commandes.
- L’état d’une commande sera changé en "facturée" si la quantité commandée est inférieure ou égale à la quantité stockée (de la référence) du produit commandé.
- Toutes les références commandées sont des références dans le stock.
- L'ensemble des commandes varie lors de l'entrée de nouvelles commandes, ou lors de l'annulation de commandes.
- Le stock varie lors de la facturation de commandes ou de nouvelles entrées de quantités de produits dans le stock

Une solution

COMMANDE

TYPES

BOOLEEN, ENTIERPOS, ETATCOMMANDE,
PRODUIT

FONCTIONS

est_en_attente : COMMANDE → BOOLEEN
est_facturée : COMMANDE → BOOLEEN
état : COMMANDE → ETATCOMMANDE
référence : COMMANDE → PRODUIT
quantité_commandée : COMMANDE → ENTIERPOS

PRECONDITIONS

AXIOMES

pour tout c : COMMANDE
non est_en_attente(c) = est_facturée(c)
est_facturée(c) = état(c)==facturée
est_en_attente(c) = état(c)==attente

Une solution

STOCK

TYPES

ENTIER, ENTIERPOS, BOOLEEN, PRODUIT

FONCTIONS

quantité_en_stock : PRODUIT x STOCK → ENTIER

ajouter : PRODUIT x ENTIERPOS x STOCK ↛ STOCK

enlever : PRODUIT x ENTIERPOS x STOCK → STOCK

en_stock : PRODUIT x STOCK → BOOLEEN

PRECONDITIONS

quantité_en_stock(p,s) = en_stock(p,s)

ajouter(p,n,s) = en_stock(p,s)

enlever(p,n,s) = en_stock(p,s) et quantité(p,s) ≥ n

AXIOMES pour tout p,q : PRODUIT, n : ENTIERPOS, s: STOCK

quantité_en_stock(q,ajouter(p,n,s)) = quantité_en_stock(p,s) + n si q = p
ou quantité_en_stock(q,s) si q ≠ p

quantité_en_stock(q,enlever(p,n,s)) = quantité_en_stock(p,s)-n si q = p
ou quantité_en_stock(q,s) si q ≠ p

Une solution

COMMANDE_STOCK

TYPES

COMMANDE, STOCK, BOOLEEN

FONCTIONS

créer : COMMANDE x STOCK → COMMANDE_STOCK

commande : COMMANDE_STOCK → COMMANDE

stock : COMMANDE_STOCK → STOCK

facturer : COMMANDE_STOCK → COMMANDE_STOCK

quantité_suffisante : COMMANDE_STOCK → BOOLEEN

référencé : COMMANDE_STOCK → BOOLEEN

facturation_possible : COMMANDE_STOCK → BOOLEEN

PRECONDITIONS

AXIOMES pour tout cs : COMMANDE_STOCK

créer(commande(cs),stock(cs))=cs commande(créer(c,s))=c

référencé(cs) = en_stock(référence(commande(cs)),stock(cs)) stock(créer(c,s))=s

quantité_suffisante(cs) = quantité_commandée(commande(cs)) ≤ quantité_en_stock(référence(commande(cs)),stock(cs))

facturation_possible(cs) = est_en_attente(commande(cs)) et référencé(cs) et quantité_suffisante(cs)

stock(facturer(cs)) = enlever(référence(commande(cs)),quantité_commandée(commande(cs)),stock(cs)) si facturation_possible(cs)

facturer(cs) = créer(commande(cs),stock(cs)) si non facturation_possible(cs)

est_facturée(commande(facturer(cs)) = facturation_possible(cs)

référence(commande(facturer(cs)) = référence(commande(cs))

quantité_commandée(commande(facturer(cs)) = quantité_commandée(commande(cs))

Conclusion

- Une spécification algébrique consiste à spécifier les opérations applicables à un objet en fonction des relations entre opérations.
- La spécification peut se faire en assemblant des blocs de spécification plus simples
- La structure naturelle de la spécification algébrique correspond aux types abstraits de données ou aux classes d'objets.