III.2 Type abstrait : spécification fonctionnelle

Définir un *type abstrait de données*

→ définir précisément quelles propriétés, opérations et fonctions seront utiles aux besoins des utilisateurs

Définir précisément ces fonctions nécessite de préciser pour chacune d'elle l'ensemble de leurs caractéristiques.

Ces caractéristiques ont un triple rôle, elles :

- participent à la définition des propriétés et opérations,
- précisent leurs sémantiques,
- sont des éléments pouvant être utilisés pour la preuve d'algorithmes.

III.2 Type abstrait : spécification fonctionnelle

Définir un *type abstrait de données*

→ définir précisément quelles propriétés, opérations et fonctions seront utiles aux besoins des utilisateurs

Définition : *Spécification fonctionnelle*

La *spécification fonctionnelle* d'un *type abstrait* regroupe :

- la définition précise des opérations ou fonctions dont auront besoin les utilisateurs,
- l'ensemble des caractéristiques des fonctions

Pour un types abstrait de données T, elles sont de la forme :

$$f : X_1 \times \ldots \times X_n \rightarrow Y_1 \times \ldots \times Y_m$$

où $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ sont des types

Pour un types abstrait de données T, elles sont de la forme :

$$f : X_1 \times \ldots \times X_n \rightarrow Y_1 \times \ldots \times Y_m$$

où $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ sont des types

- les *fonctions d'accès* qui sont celles où T apparaît seulement à gauche de la flèche : elles permettent d'accéder aux valeurs des informations associées au type T; deux types :

Pour un types abstrait de données T, elles sont de la forme :

$$f : X_1 \times \ldots \times X_n \rightarrow Y_1 \times \ldots \times Y_m$$

où $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ sont des types

- les *fonctions d'accès* qui sont celles où T apparaît seulement à gauche de la flèche : elles permettent d'accéder aux valeurs des informations associées au type T; deux types :
  - les *propriétés* : seul le type apparait à gauche

Pour un types abstrait de données T, elles sont de la forme :

$$f : X_1 \times \ldots \times X_n \rightarrow Y_1 \times \ldots \times Y_m$$

où $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ sont des types

- les *fonctions d'accès* qui sont celles où T apparaît seulement à gauche de la flèche : elles permettent d'accéder aux valeurs des informations associées au type T; deux types :
  - les *propriétés* : seul le type apparait à gauche
  - les *requêtes* : le type + d'autres paramètres sont à gauche

Pour un types abstrait de données T, elles sont de la forme :

$$f : X_1 \times \ldots \times X_n \rightarrow Y_1 \times \ldots \times Y_m$$

où $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ sont des types

- les *fonctions d'accès* qui sont celles où T apparaît seulement à gauche de la flèche : elles permettent d'accéder aux valeurs des informations associées au type T; deux types :
  - les *propriétés* : seul le type apparait à gauche
  - les *requêtes* : le type + d'autres paramètres sont à gauche
- les *fonctions de modification* qui sont celle où T apparaît à droite et à gauche de la flèche : elles permettent de modifier des objets de type T ; Parmi elles, on peut retrouver des propriétés ;

Pour un types abstrait de données T, elles sont de la forme :

$$f : X_1 \times \ldots \times X_n \rightarrow Y_1 \times \ldots \times Y_m$$

où $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ sont des types

- les *fonctions d'accès* qui sont celles où T apparaît seulement à gauche de la flèche : elles permettent d'accéder aux valeurs des informations associées au type T; deux types :
  - les *propriétés* : seul le type apparait à gauche
  - les *requêtes* : le type + d'autres paramètres sont à gauche
- les *fonctions de modification* qui sont celle où T apparaît à droite et à gauche de la flèche : elles permettent de modifier des objets de type T ; Parmi elles, on peut retrouver des propriétés ;
- les *fonctions de création* qui sont celles où T apparaît seulement à droite de la flèche : elles permettent de créer des objets de types T.

Exemple : voiture à vendre

Soit une voiture à vendre, le vendeur doit pouvoir calculer son prix de vente, savoir si une remise est possible, de combien et bien sûr annoncer son prix de vente avec ou sans remise.

Exemple : voiture à vendre

Soit une voiture à vendre, le vendeur doit pouvoir calculer son prix de vente, savoir si une remise est possible, de combien et bien sûr annoncer son prix de vente avec ou sans remise.

Les fonctionnalités dont à besoin le vendeur sont :

- quel est le modèle de la voiture ?
- quel est son prix de vente ?
- une remise est-t-elle possible ?
- quel est montant max de la remise ?
- quel est le prix remisé ?

Exemple : voiture à vendre

Soit une voiture à vendre, le vendeur doit pouvoir calculer son prix de vente, savoir si une remise est possible, de combien et bien sûr annoncer son prix de vente avec ou sans remise.

nom:	`Voiture→Text`	*// nom de la voiture*
prix:	`Voiture→Float`	*// prix de la voiture (non remisé)*
a_remise:	`Voiture→Bool`	*// remise possible ?*
remise_max:	`Voiture→Float`	*// remise maximum (en %)*
prix_remise:	`Voiture x Float→Float`	*// prix de la voiture (avec remise)*
remise_max:	`Voiture x Float→Voiture`	*// modifier la remise max*

Exemple : voiture à vendre

Soit une voiture à vendre, le vendeur doit pouvoir calculer son prix de vente, savoir si une remise est possible, de combien et bien sûr annoncer son prix de vente avec ou sans remise.

nom:	`Voiture→Text`	*(1) nom(v)⇒prix(v)>0*
prix:	`Voiture→Float`	*(2) prix(v)>0*
a_remise:	`Voiture→Bool`	*(3) a_remise(v)⇔remise_max(v)>0*
remise_max:	`Voiture→Float`	*(4) 0 ≤ remise_max(v) < 1*
prix_remise:	`Voiture x Float→Float`	*(5) prix_remise(v,r)* *(6) prix_remise(v,r) ≤ prix(v)-prix(v)*remise_max(v)* *(7) prix_remise(v,r)⇒r<remise_max(v)*
remise_max:	`Voiture x Float→Voiture`	*(4) remise_max(remise_max(v,r))==r*

nom:	`Voiture→Text`	*(1) nom(v)⇒prix(v)>0*
prix:	`Voiture→Float`	*(2) prix(v)>0*
a_remise:	`Voiture→Bool`	*(3) remise(v)⇔remise_max(v)>0*
remise_max:	`Voiture→Float`	*(4) 0 ≤ remise_max(v) < 1*
prix_remise:	`Voiture x Float→Float`	*(5) prix_remise(v,r)* *(6) prix_remise(v,r) ≤ prix(v)-prix(v)*remise_max(v)* *(7) prix_remise(v,r)⇒r<remise_max(v)*
remise_max:	`Voiture x Float→Voiture`	*(4) remise_max(remise_max(v,r))==r*

fonctions d'accès

nom:	`Voiture→Text`	*(1) nom(v)⇒prix(v)>0*
prix:	`Voiture→Float`	*(2) prix(v)>0*
a_remise:	`Voiture→Bool`	*(3) remise(v)⇔remise_max(v)>0*
remise_max:	`Voiture→Float`	*(4) 0 ≤ remise_max(v) < 1*
prix_remise:	`Voiture x Float→Float`	*(5) prix_remise(v,r)* *(6) prix_remise(v,r) ≤ prix(v)-prix(v)*remise_max(v)* *(7) prix_remise(v,r)⇒r<remise_max(v)*
remise_max:	`Voiture x Float→Voiture`	*(4) remise_max(remise_max(v,r))==r*

fonctions d'accès

modification

nom:	`Voiture→Text`	*(1) nom(v)⇒prix(v)>0*
prix:	`Voiture→Float`	*(2) prix(v)>0*
a_remise:	`Voiture→Bool`	*(3) remise(v)⇔remise_max(v)>0*
remise_max:	`Voiture→Float`	*(4) 0 ≤ remise_max(v) < 1*
prix_remise:	`Voiture x Float→Float`	*(5) prix_remise(v,r)* *(6) prix_remise(v,r) ≤ prix(v)-prix(v)*remise_max(v)* *(7) prix_remise(v,r)⇒r<remise_max(v)*
remise_max:	`Voiture x Float→Voiture`	*(4) remise_max(remise_max(v,r))==r*

fonctions d'accès

nom:	`Voiture→Text`	*propriétés*	*(1) nom(v)⇒prix(v)>0*
prix:	`Voiture→Float`		*(2) prix(v)>0*
a_remise:	`Voiture→Bool`		*(3) remise(v)⇔remise_max(v)>0*
remise_max:	`Voiture→Float`		*(4) 0 ≤ remise_max(v) < 1*
prix_remise:	`Voiture x Float→Float`		*(5) prix_remise(v,r)* *(6) prix_remise(v,r) ≤ prix(v)-prix(v)*remise_max(v)* *(7) prix_remise(v,r)⇒r<remise_max(v)*

modification

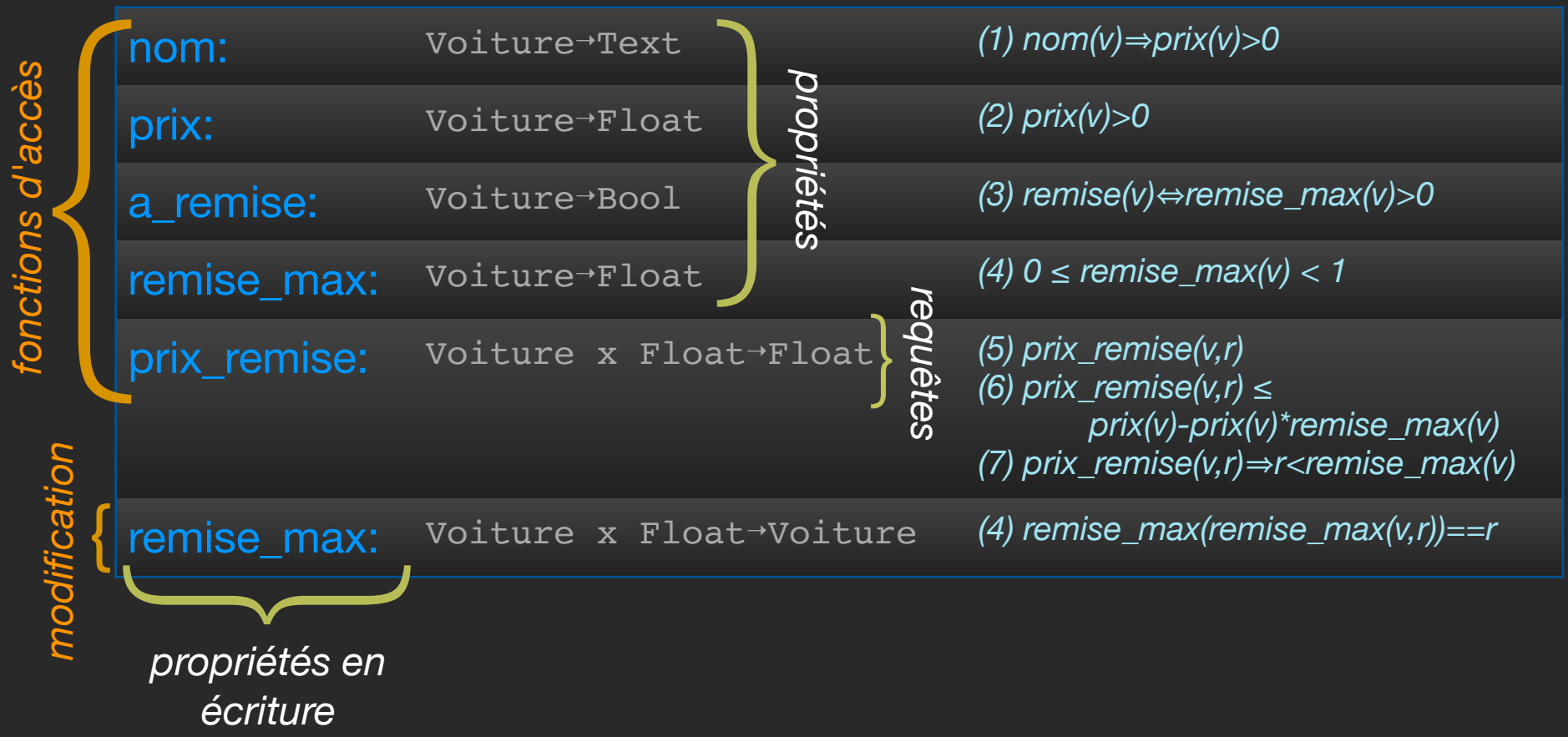
remise_max:	`Voiture x Float→Voiture`	*(4) remise_max(remise_max(v,r))==r*

fonctions d'accès

nom:	`Voiture→Text`	*propriétés*	(1) nom(v)⇒prix(v)>0
prix:	`Voiture→Float`		(2) prix(v)>0
a_remise:	`Voiture→Bool`		(3) remise(v)⇔remise_max(v)>0
remise_max:	`Voiture→Float`		(4) 0 ≤ remise_max(v) < 1
prix_remise:	`Voiture x Float→Float`	*requêtes*	(5) prix_remise(v,r) (6) prix_remise(v,r) ≤ prix(v)-prix(v)*remise_max(v) (7) prix_remise(v,r)⇒r<remise_max(v)

modification

remise_max:	`Voiture x Float→Voiture`	(4) remise_max(remise_max(v,r))==r


fonctions d'accès
modification
nom:
Voiture→Text
prix:
Voiture→Float
a_remise:
Voiture→Bool
remise_max:
Voiture→Float
propriétés
prix_remise:
Voiture x Float→Float
requêtes
(1) nom(v)⇒prix(v)>0
(2) prix(v)>0
(3) remise(v)⇔remise_max(v)>0
(4) 0 ≤ remise_max(v) < 1
(5) prix_remise(v,r)
(6) prix_remise(v,r) ≤
prix(v)-prix(v)*remise_max(v)
(7) prix_remise(v,r)⇒r<remise_max(v)
remise_max:
Voiture x Float→Voiture
(4) remise_max(remise_max(v,r))==r
propriétés en
écriture