Spécifications algébriques

Typage des constantes et opérateurs

- Soit $\mathcal{S}$ un ensemble dénombrable de symboles, les sortes utilisées pour distinguer les termes possédant les mêmes caractéristiques
- Les termes sont séparés selon leur sorte : $\mathcal{T} = \bigcup_{s \in \mathcal{S}} \mathcal{T}_s$
- Les constantes également : $\mathcal{C} = \bigcup_{s \in \mathcal{S}} \mathcal{C}_s$
- Les variables également : $\mathcal{V} = \bigcup_{s \in \mathcal{S}} \mathcal{V}_s$
- L'arité des fonctions prend en compte la sorte des paramètres et du résultat

$$\forall n \in \mathbb{N}. \mathcal{F}_n = \bigcup_{s \in \mathcal{S}, \forall i \in [1, \ldots, n]. s_i \in \mathcal{S}} \mathcal{F}_{(s_1 \times \cdots \times s_n) \mapsto s}$$

- L'arité est donc étendue pour intégrer les sortes

Structure de termes

Vision ensembliste

- Soit $\mathcal{S}$ (resp. $\mathcal{C}$, resp. $\mathcal{F}$) un ensemble dénombrable de sortes (resp. constantes, resp. fonctions)
- L'ensemble des termes $\mathcal{T}$ partitionné selon les sortes est le plus petit ensemble tel que :
  - $\forall s \in \mathcal{S}. \forall c \in \mathcal{C}_s. \, c \in \mathcal{T}_s$
  - $\forall s_1, \ldots s_n, \, s \in \mathcal{S}. \forall f \in \mathcal{F}_{s_1 \times \cdots \times s_n \mapsto s}.$ $\forall t_1 \in \mathcal{T}_{s_1}. \ldots . \forall t_n \in \mathcal{T}_{s_n}. f(t_1, \ldots, t_n) \in \mathcal{T}_s$
- Exemple : Les entiers naturels de Peano

$$
\begin{gathered}
nat \in \mathcal{S} \\
zero \in \mathcal{C}_{nat} \\
successeur \in \mathcal{F}_{nat \mapsto nat}
\end{gathered}
$$

  L'ensemble des termes est la plus petite solution de l'équation :

$$\mathcal{T}_{nat} = \{zero\} \cup \{successeur(n) \mid n \in \mathcal{T}_{nat}\}$$

- Ces définitions peuvent être stratifiées pour éliminer les paradoxes

Structure de termes avec variable

Vision ensembliste

- Soit $\mathcal{V}$ un ensemble dénombrable de variables
- L'ensemble des termes $\mathcal{T}[\mathcal{V}]$ avec variables partitionné selon les sortes est le plus petit ensemble tel que :
  - $\forall s \in \mathcal{S}.\, \forall c \in \mathcal{C}_s.\, c \in \mathcal{T}[\mathcal{V}]_s$
  - $\forall s \in \mathcal{S}.\, \forall x \in \mathcal{V}_s.\, x \in \mathcal{T}[\mathcal{V}]_s$
  - $\forall s_1, \ldots, s_n, s \in \mathcal{S}.\, \forall f \in \mathcal{F}_{s_1 \times \cdots \times s_n \mapsto s}.$
    $\forall t_1 \in \mathcal{T}[\mathcal{V}]_{s_1}, \ldots, t_n \in \mathcal{T}[\mathcal{V}]_{s_n}.\, f(t_1, \ldots, t_n) \in \mathcal{T}[\mathcal{V}]_s$
- Notons qu'une substitution est une fonction de $\mathcal{V}$ vers $\mathcal{T}[\mathcal{V}]$ qui associe à une variable d'une sorte un terme de la même sorte
- Ces définitions peuvent être stratifiées pour éliminer les paradoxes

Structure de termes

Vision déductive

- La construction des termes de sorte s correspond aux règles d'introduction de s :

	$\dfrac{c \in \mathcal{C}_s}{c \in \mathcal{T}[\mathcal{V}]_s}\ I_s^c \qquad \dfrac{x \in \mathcal{V}_s}{x \in \mathcal{T}[\mathcal{V}]_s}\ I_s^x$
$n \in \mathbb{N}^\star$	$\dfrac{f \in \mathcal{F}_{s_1 \times \ldots \times s_n \mapsto s} \qquad t_1 \in \mathcal{T}[\mathcal{V}]_{s_1} \qquad t_n \in \mathcal{T}[\mathcal{V}]_{s_n}}{f(t_1, \ldots, t_n) \in \mathcal{T}[\mathcal{V}]_s}\ I_s^f$

- Exemple : Les entiers naturels de Peano

$$\frac{}{zero \in \mathcal{T}[\mathcal{V}]_{nat}}\ I_{nat}^{zero}$$

$$\frac{n \in \mathcal{T}[\mathcal{V}]_{nat}}{successeur(n) \in \mathcal{T}[\mathcal{V}]_{nat}}\ I_{nat}^{successeur}$$

Modélisation de l'arithmétique

Formulation classique

- Entiers naturels (arithmétique de Peano) : $\mathbb{N}$ modélisé par
  - $zero \in \mathcal{C}_0$ $(\overline{0} = \emptyset)$
  - $successeur \in \mathcal{F}_1$ $(\overline{n+1} = \{\overline{n}\} \cup \overline{n})$
- Entiers relatifs : $\mathbb{Z}$ modélisé par $\mathbb{N}^2$ avec
  - $(n, 0)$ modélise $\mathbb{Z}^+$ (représentant classe équivalence)
  - $(0, n)$ modélise $\mathbb{Z}^-$ (représentant classe équivalence)
- Nombres rationnels : $\mathbb{Q}$ modélisé par $\mathbb{Z} \times \mathbb{N}^\star$ avec $(a, b) \in \mathbb{Z} \times \mathbb{N}^\star$ tel que $a \wedge b = 1$ modélise $a/b \in \mathbb{Q}$ (représentant classe équivalence)
- Nombres réels (coupure de Dedekind) : $\mathbb{R}$ modélisé par $\mathcal{P}(\mathbb{Q})^2$ avec $x \in \mathbb{R} \setminus \mathbb{Q}$ modélisé par la coupure $(\{q \in \mathbb{Q} \mid q < x\}, \{q \in \mathbb{Q} \mid x < q\})$ $(\aleph_0 = \mid \mathbb{N} \mid = \omega, \aleph_1 = 2^{\aleph_0} = \mid \mathbb{R} \mid)$
- Arithmétique décidable de Presburger : arithmétique linéaire (pas de multiplications entre variables)

Equations sur les termes

Propriétés algébriques

- L'égalité $=$ est une relation (prédicat) d'équivalence définie dans la plupart des structures (appelées égalitaires) :
  - Réflexive : $\forall x.\ x = x$
  - Symétrique : $\forall x.\ \forall y.\ x = y \rightarrow y = x$
  - Transitive : $\forall x.\ \forall y.\ \forall z.\ x = y \wedge y = z \rightarrow x = z$
- Mais, en logique des prédicats elle est traitée comme un prédicat quelconque dont les propriétés doivent être formalisées
- Les logiques équationnelles lui accordent un rôle particulier
- Une équation gardée sur la sorte $s \in \mathcal{S}$ est de la forme :

$$\forall x_1 \in \mathcal{T}_{s_1}, \ldots, \forall x_n \in \mathcal{T}_{s_n}.\ \varphi \rightarrow (G = D)$$

  avec $G \in \mathcal{T}[\mathcal{V}]_s$, $D \in \mathcal{T}[\mathcal{V}]_s$, φ une formule bien formée de la logique des prédicats et $\{x_1, \ldots, x_n\} = VL(\varphi) \cup VL(G) \cup VL(D)$
- Elle est notée $\varphi \rightarrow (G = D)$

Equations sur les termes

Exemple

- Définition de l'addition pour les entiers de Peano :

$$
\begin{array}{ll}
 & somme \in \mathcal{F}_{nat \times nat \mapsto nat} \\
(a) & \forall m \in \mathcal{T}_{nat}.\ somme(zero, m) = m \\
(b) & \left[\begin{array}{l} \forall n \in \mathcal{T}_{nat}, \\ \forall m \in \mathcal{T}_{nat}, \\ somme(successeur(n), m) = successeur(somme(n, m)) \end{array}\right]
\end{array}
$$

- Calcul d'une addition par réécriture en utilisant les équations :

$$
\begin{array}{l}
somme(successeur(successeur(zero)), successeur(zero)) \\
\overset{(b)}{=} successeur(somme(successeur(zero), successeur(zero))) \\
\overset{(b)}{=} successeur(successeur(somme(zero, successeur(zero)))) \\
\overset{(a)}{=} successeur(successeur(successeur(zero)))
\end{array}
$$

Spécification algébrique

Sémantique

- Une spécification algébrique est définie par un ensemble dénombrable de sortes, un ensemble dénombrable de constantes, un ensemble dénombrables de fonctions et un ensemble dénombrables d'équations
- La sémantique d'une spécification algébrique est donnée par une interprétation comme pour la logique des prédicats
- L'interprétation d'une spécification algébrique doit valider toutes les équations
- Une interprétation particulière appelée sémantique initiale s'appuie sur la structure d'algèbre de termes partitionnée en classes d'équivalence selon les équations
- Les formules de logique équationnelle sont des formules de logique des prédicats du premier ordre exploitant l'opérateur $=$ sur les termes, c'est-à-dire contenant des équations

Déduction naturelle

Logique équationnelle

$$\frac{}{\Gamma \vdash t = t}\ \text{Réflexivité} \qquad \frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash t_2 = t_1}\ \text{Symétrie}$$

$$\frac{\Gamma \vdash t_1 = t_2 \qquad \Gamma \vdash t_2 = t_3}{\Gamma \vdash t_1 = t_3}\ \text{Transitivité}$$

$$\frac{f \in \mathcal{F}_{s_1 \times \ldots \times s_n \mapsto s} \qquad \Gamma \vdash t_1 = t_1' \qquad \Gamma \vdash t_n = t_n'}{\Gamma \vdash f(t_1, \ldots, t_n) = f(t_1', \ldots, t_n')}\ \text{Congruence}$$

$$\frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash \sigma\, t_1 = \sigma\, t_2}\ \text{Substitution} \qquad \frac{\Gamma \vdash t_1 = t_2 \qquad \Gamma \vdash [t_2/t_1]\,\varphi}{\Gamma \vdash \varphi}\ \text{Réécriture}$$

- En déduction naturelle “pure”, seules Réflexivité et Réécriture sont nécessaires (resp. introduction et élimination de l’égalité)
- La substitution $[t_2/t_1]\,\varphi$ est une généralisation de la substitution d’une variable au remplacement d’un terme par un autre terme. Les contraintes sur les variables libres sont similaires.

Preuve par induction

Formulation classique

- Rappel: Récurrence simple

$$(\forall n \in \mathbb{N}.\ \varphi) \leftrightarrow \begin{cases} [0/n]\,\varphi \\ \wedge \forall m \in \mathbb{N}.\ ([m/n]\,\varphi \rightarrow [m+1/n]\,\varphi) \end{cases}$$

- Rappel: Récurrence généralisée

$$(\forall n \in \mathbb{N}.\ \varphi) \leftrightarrow \begin{cases} [0/n]\,\varphi \\ \wedge \forall p \in \mathbb{N}.\ ((\forall q \in \mathbb{N}.\ q < p \rightarrow [q/n]\,\varphi) \rightarrow [p/n]\,\varphi) \end{cases}$$

- Induction bien fondée: la relation d'ordre strict $< : \mathcal{E} \times \mathcal{E}$ est bien fondée s'il n'existe pas de chaîne de $\mathcal{E}$ infiniment décroissante

$$(\forall x \in \mathcal{E}.\ \varphi) \leftrightarrow \forall y \in \mathcal{E}.\ ((\forall z \in \mathcal{E}.\ z < y \rightarrow [z/x]\,\varphi) \rightarrow [y/x]\,\varphi)$$

- Exemple : $<$ est bien fondée sur $\mathbb{N}$. Une telle relation sur les termes peut être définie par plongement sur $\mathbb{N}$.

Preuve par induction

Formulation déductive

- Récurrence et induction sont des règles d'élimination
- Rappel : Récurrence simple

$$\frac{\Gamma \vdash n \in \mathbb{N} \qquad \Gamma \vdash [0/n]\,\varphi \qquad \Gamma,\ m \in \mathbb{N}, [m/n]\,\varphi \vdash [m+1/n]\,\varphi}{\Gamma \vdash \varphi}\ E^{RS}_{nat}$$

- Rappel : Récurrence généralisée

$$\frac{\Gamma \vdash n \in \mathbb{N} \qquad \Gamma \vdash [0/n]\,\varphi \qquad \Gamma,\ p \in \mathbb{N}, (\forall q \in \mathbb{N}.\ q < p \to [q/n]\,\varphi) \vdash [p/n]\,\varphi}{\Gamma \vdash \varphi}\ E^{RG}_{nat}$$

- Induction bien fondée

$$\frac{\Gamma \vdash x \in \mathcal{E} \qquad \Gamma,\ y \in \mathcal{E}, (\forall z \in \mathcal{E}.\ z < y \to [z/x]\,\varphi) \vdash [y/x]\,\varphi}{\Gamma \vdash \varphi}\ E^{BF}_{nat}$$

Preuve par induction

Formulation classique

- Induction structurelle:

$$\forall s \in \mathcal{S}.\\ (\forall t \in \mathcal{T}_s.\, \varphi)\\ \leftrightarrow\\ \left(\begin{array}{c} \forall c \in \mathcal{C}_s.\, [c/t]\,\varphi \\ \wedge \\ \forall f \in \mathcal{F}_{s_1 \times \cdots \times s_n \mapsto s}.\, \forall t_1 \in \mathcal{T}_{s_1}, \ldots, \forall t_n \in \mathcal{T}_{s_n}. \\ (\bigwedge_{j \in [1 \ldots n]} (s_j = s \rightarrow [t_j/t]\,\varphi)) \rightarrow [f(t_1, \ldots, t_n)/t]\,\varphi \end{array} \right)$$

- Remarque : Hypothèse d'induction uniquement lorsque $s_i = s$
- Exemple : Entier naturel de Peano

$$(\forall t \in \mathcal{T}_{nat}.\, \varphi)\\ \leftrightarrow\\ ([zero/t]\,\varphi \wedge \forall p \in \mathcal{T}_{nat}.\, [p/t]\,\varphi \rightarrow [successeur(p)/t]\,\varphi)$$

- Identique dans ce cas à la récurrence simple

Preuve par induction structurelle

Exemple

- Exemple : $\forall n_1 \in T_{nat}.\ \forall n_2 \in T_{nat}.\ \exists r \in T_{nat}.\ r = somme(n_1, n_2)$
- Induction sur n_1 :
  - $n_1 = zero$: Prouvons $\forall n_2 \in T_{nat}.\ \exists r \in T_{nat}.\ r = somme(zero, n_2)$
    Nous avons $somme(zero, n_2) = n_2$
    Prenons donc $r = n_2$
  - $n_1 = successeur(n_3)$ avec $n_3 \in T_{nat}$ et
    $\forall n_2 \in T_{nat}.\ \exists r_2 \in T_{nat}.\ r_2 = somme(n_3, n_2)$:
    Prouvons $\forall n_2 \in T_{nat}.\ \exists r_1 \in T_{nat}.\ r_1 = somme(successeur(n_3), n_2)$
    Nous avons $somme(successeur(n_3), n_2) =$
    $successeur(somme(n_3, n_2)) = successeur(r_2)$
    Prenons donc $r_1 = successeur(r_2)$

Preuve par induction

Formulation déductive

- Induction structurelle:
- Prenons $\{c_1, \ldots, c_p\} = \mathcal{C}_s$
- Prenons $\{f_1, \ldots, f_q\} = \bigcup\limits_{\substack{n\in\mathbb{N}\\ s_1,\ldots,s_n\in\mathcal{S}}} \mathcal{F}_{s_1\times\cdots\times s_n\mapsto s}$
- La règle ci-dessous possède $p+q$ prémisses $c_i \in \mathcal{C}_s$ et $f_j \in \mathcal{F}_{s_1\times\cdots\times s_{n_j}\mapsto s}$

$$\frac{\Gamma \vdash t \in \mathcal{T}_s \qquad \begin{array}{cc} \Gamma \vdash [c_1/t]\,\varphi & \Gamma,\ t_i \in \mathcal{T}_{s_i}, (s_i = s \rightarrow [t_i/t]\,\varphi) \vdash [f_{n,1}(t_1,\ldots,t_n)/t]\,\varphi \\ \vdots & \vdots \\ \Gamma \vdash [c_p/t]\,\varphi & \Gamma,\ t_i \in \mathcal{T}_{s_i}, (s_i = s \rightarrow [t_i/t]\,\varphi) \vdash [f_{n,q}(t_1,\ldots,t_n)/t]\,\varphi \end{array}}{\Gamma \vdash \varphi}\ E_s$$

- Remarque : Hypothèse d'induction uniquement lorsque $s_k = s$
- Exemple : Entier naturel de Peano

$$\frac{\Gamma \vdash t \in \mathcal{T}_{nat} \qquad \Gamma \vdash [zero/t]\,\varphi \qquad \Gamma,\ p \in \mathcal{T}_{nat}, [p/t]\,\varphi \vdash [successeur(p)/t]\,\varphi}{\Gamma \vdash \varphi}\ E_{nat}$$

- Identique à la récurrence simple

Induction structurelle et Déduction naturelle

Exemple

- Exemple : $\forall n_1 \in T_{nat}.\ \forall n_2 \in T_{nat}.\ \exists r \in T_{nat}.\ r = somme(n_1, n_2)$

$$
\dfrac{\dfrac{\begin{array}{c}
\dfrac{}{n_1 \in \mathcal{T}_{nat} \vdash n_1 \in \mathcal{T}_{nat}}\ Hyp(\emptyset;\ n_1 \in \mathcal{T}_{nat}) \\
\vdots \\
\hline
n_1 \in \mathcal{T}_{nat} \vdash \forall n_2 \in T_{nat}.\ \exists r \in T_{nat}.\ r = somme(zero, n_2) \\
\vdots \\
\hline
\begin{array}{l} n_1 \in \mathcal{T}_{nat}, \\ n_3 \in \mathcal{T}_{nat} \\ \left[\begin{array}{l} \forall n_2 \in T_{nat}, \\ \exists r_2 \in T_{nat}, \\ r_2 = somme(n_3, n_2) \end{array}\right] \end{array} \vdash \left[\begin{array}{l} \forall n_2 \in T_{nat}, \\ \exists r \in T_{nat}, \\ r = somme(successeur(n_3), n_2) \end{array}\right]
\end{array}}{n_1 \in \mathcal{T}_{nat} \vdash \forall n_2 \in T_{nat}.\ \exists r \in T_{nat}.\ r = somme(n_1, n_2)}\ E_{nat}}{\vdash \forall n_1 \in T_{nat}.\ \forall n_2 \in T_{nat}.\ \exists r \in T_{nat}.\ r = somme(n_1, n_2)}\ I_{\forall}
$$

Preuves de programmes fonctionnels

- Un programme fonctionnel pur est semblable à une fonction mathématique
- Pas de modification de l'état de la mémoire (effet de bord)
- Chaque appel avec les mêmes valeurs de paramètres calcule le même résultat
- Langages appropriés : Haskell, Coq, Isabelle, ...; restrictions d'OCaML, de $F^\sharp$, de Lisp, de Scheme, de Clojure, ...
- La spécification du programme est une propriété des valeurs du résultat, en fonction des valeurs des paramètres
- Les données typées sont représentées par des termes
- Les programmes sont des ensembles d'équations sur les termes