Chapitre 2

Récursivité et induction

2.1 Motivation

Les définitions récursives sont omniprésentes en informatique. Elles sont présentes à la fois dans les langages de programmation, mais aussi présentes dans de nombreux concepts que l'on manipule.

Exemple 2.1 (Listes en JAVA) *En JAVA, avec*

```
class Liste {
  int contenu;
  Liste suivant;
  }
Liste lst;
```

on définit la classe Liste de façon récursive (inductive) : en utilisant dans la définition de la classe, le champ " suivant " du type de la classe Liste elle même.

Exemple 2.2 (Arbres ordonnés) *Nous avons défini les arbres ordonnés dans le chapitre précédent en passant par la notion de graphe. Une alternative naturelle serait de présenter les arbres ordonnés par une définition récursive : un arbre ordonné est soit vide, soit réduit à un sommet (une racine), soit constitué d'un sommet (une racine) et une liste (ordonnée) d'arbres ordonnés (ses fils).*

Dans ce chapitre, nous nous attardons sur les définitions inductives d'ensembles et de fonctions, qui permettent de donner un sens à des définitions récursives.

Nous discutons, par ailleurs comment il est possible de faire des preuves sur des structures définies inductivement, en introduisant les preuves par induction structurelle.

2.2 Raisonnement par récurrence sur $\mathbb{N}$

L'induction structurelle est une généralisation de la preuve par récurrence : revenons sur cette dernière pour avoir les idées au clair.

Lorsque l'on raisonne sur les entiers, le *premier principe d'induction* aussi appelé *principe de récurrence mathématique* est un mode de raisonnement particulièrement utile.

Théorème 2.1 *Soit $P(n)$ un prédicat (une propriété) dépendant de l'entier n. Si les deux conditions suivantes sont vérifiées :*

(B) $P(0)$ est vrai ;

(I) $P(n)$ implique $P(n+1)$ pour tout n ;

alors pour tout entier n, $P(n)$ est vrai.

Démonstration : Le raisonnement se fait par l'absurde. Considérons $X = \{k \in \mathbb{N} | P(k) \text{ est faux}\}$. Si X est non vide, il admet un plus petit élément n. D'après la condition (B), $n \neq 0$, et donc $n-1$ est un entier, et $P(n-1)$ est vrai par définition de X. On obtient une contradiction avec la propriété (I) appliquée pour l'entier $n-1$. □

Pour faire une preuve par récurrence, on établit donc une propriété en 0 (cas de base), et on établit que la propriété est *héréditaire*, ou *inductive* : $P(n)$ implique $P(n+1)$ pour tout n.

Le concept de preuve inductive généralise cette idée à d'autres ensembles que les entiers, à savoir aux ensembles qui se définissent inductivement.

Exercice 2.1. *On considère $S_n = 1^3 + 2^3 + \cdots + (2n-1)^3$. Prouver par récurrence que $S_n = 2n^4 - n^2$.*

Exercice 2.2. *Prouver par récurrence que $\sum_{k=1}^{n} \frac{1}{4k^2-1} = \frac{n}{2n+1}$.*

Exercice 2.3 (corrigé page ??). *Le théorème plus haut est parfois nommé "premier principe d'induction". Prouver le "second principe d'induction" : soit $P(n)$ une propriété dépendant de l'entier n. Si la propriété suivante est vérifiée : si pour tout $n \in \mathbb{N}$, si en supposant pour tout entier $k < n$ la propriété $P(k)$ on déduit $P(n)$, alors pour tout $n \in \mathbb{N}$, la propriété $P(n)$ est vraie.*

Exercice 2.4 (corrigé page ??). *On fixe un alphabet Σ. On rappelle qu'un langage sur Σ est une partie de Σ^*. Si L_1 et L_2 sont deux langages de Σ^*, on définit leur concaténation par $L_1.L_2 = \{u.v | u \in L_1, v \in L_2\}$. La concaténation est une opération associative admettant $\{\epsilon\}$ comme élément neutre. On peut alors définir les puissances d'un langage L de la façon suivante : $L^0 = \{\epsilon\}$, et pour un entier $n > 0$, $L^{n+1} = L^n.L = L.L^n$. L'étoile d'un langage L est défini par $L^* = \bigcup_{n \in \mathbb{N}} L^n$.*

Soient L et M deux langages sur Σ, avec $\epsilon \notin L$. Montrer que dans $\mathcal{P}(\Sigma^*)$ (les langages sur Σ), l'équation $X = L.X \cup M$ admet pour unique solution le langage $L^*.M$.

2.3 Définitions inductives

Les définitions inductives visent à définir des parties d'un ensemble E.

Remarque 2.1 *Cette remarque est pour les puristes. Elle peut être évitée lors de la première lecture de ce document.*

Nous nous restreignons dans ce document au cadre où l'on souhaite définir par induction des objets qui correspondent à des parties d'un ensemble déjà connu E. Nous faisons cela pour éviter les subtilités et paradoxes de la théorie des ensembles.

Le lecteur très attentif pourra observer que l'on considérera dans la suite très souvent l'écriture syntaxique des objets plutôt que les objets eux-mêmes. En effet, en faisant ainsi, on garantit que l'on se place sur l'ensemble $E = \Sigma^*$ pour un certain alphabet Σ, et on évite de se poser la question de l'existence de l'ensemble E sous-jacent dans les raisonnements qui suivent.

Par exemple, pour formaliser complètement l'exemple 2.1 plus haut, on chercherait plutôt à définir une représentation syntaxique des listes plutôt que les listes.

Lorsque l'on veut définir un ensemble, ou une partie d'un ensemble, une façon de faire est de le définir de façon *explicite*, c'est-à-dire en décrivant précisément quels sont ses éléments.

Exemple 2.3 *Les entiers pairs peuvent se définir par $P = \{n | \exists k \in \mathbb{N}\ n = 2 * k\}$.*

Malheureusement, ce n'est pas toujours aussi facile, et il est souvent beaucoup plus commode de définir un ensemble par une définition *inductive*. Un exemple typique de définition inductive est une définition comme celle-ci :

Exemple 2.4 *Les entiers pairs correspondent aussi au plus petit ensemble qui contient 0 et tel que si n est pair, alors $n + 2$ est pair.*

Remarque 2.2 *Observons que l'ensemble des entiers $\mathbb{N}$ vérifie bien que 0 est un entier, et que si n est un entier $n + 2$ aussi. Il y a donc besoin de dire que c'est le plus petit ensemble avec cette propriété.*

2.3.1 Principe général d'une définition inductive

Intuitivement, une partie X se définit inductivement si on peut la définir avec la donnée explicite de certains éléments de X et de moyens de construire de nouveaux éléments de X à partir d'éléments de X.

De façon générique, dans une définition inductive,

— certains éléments de l'ensemble X sont donnés explicitement (c'est-à-dire on se donne un ensemble B d'éléments de X, qui constitue l'ensemble de base de la définition inductive) ;

— les autres éléments de l'ensemble X sont définis en fonction d'éléments appartenant déjà à l'ensemble X selon certaines règles (c'est-à-dire on se donne des règles R de formation d'éléments, qui constituent les étapes inductives de la définition récursive).

On considère alors le plus petit ensemble qui contient B et qui est *stable* (on dit aussi parfois *clos* ou *fermé*) par les règles de R.

2.3.2 Formalisation : Premier théorème du point fixe

Formellement, tout cela se justifie par le théorème qui suit.

Définition 2.1 (Définition inductive) *Soit E un ensemble. Une* définition inductive *d'une partie X de E consiste à se donner*

- *un sous ensemble non vide B de E (appelé* ensemble de base*)*
- *et d'un ensemble de* règles *R : chaque règle $r_i \in R$ est une fonction (qui peut être partielle) r_i de $E^{n_i} \to E$ pour un certain entier $n_i \geq 1$.*

Théorème 2.2 (Théorème du point fixe) *A une définition inductive correspond un plus petit ensemble qui vérifie les propriétés suivantes :*

(B) *il contient B : $B \subset X$;*

(I) *il est stable par les règles de R : pour chaque règle $r_i \in R$, pour tout $x_1, \cdots, x_{n_i} \in X$, on a $r_i(x_1, \cdots, x_{n_i}) \in X$.*

On dit que cet ensemble est alors défini inductivement*.*

Démonstration : Soit $\mathcal{F}$ l'ensemble des parties de E vérifiant (B) et (I). L'ensemble $\mathcal{F}$ est non vide car il contient au moins un élément : en effet, l'ensemble E vérifie les conditions (B) et (I) et donc $E \in \mathcal{F}$.

On peut alors considérer X défini comme l'intersection de tous les éléments de $\mathcal{F}$. Formellement,

$$X = \bigcap_{Y \in \mathcal{F}} Y. \tag{2.1}$$

Puisque B est inclus dans chaque $Y \in \mathcal{F}$, B est inclus dans X. Donc X vérifie la condition (B).

L'ensemble obtenu vérifie aussi (I). En effet, considérons une règle $r_i \in R$, et des $x_1, \cdots, x_{n_i} \in X$. On a $x_1, \cdots, x_{n_i} \in Y$ pour chaque $Y \in \mathcal{F}$. Pour chaque tel Y, puisque Y est stable par la règle r_i, on doit avoir $r(x_1, \cdots, x_{n_i}) \in Y$. Puisque cela est vrai pour tout $Y \in \mathcal{F}$, on a aussi $r(x_1, \cdots, x_{n_i}) \in X$, ce qui prouve que X est stable par la règle r_i.

X est le plus petit ensemble qui vérifie les conditions (B) et (I), car il est par définition inclus dans tout autre ensemble vérifiant les conditions (B) et (I).

□

2.3.3 Différentes notations d'une définition inductive

Notation 2.1 *On note souvent une définition inductive sous la forme*

(B) $x \in X$
avec une ligne comme celle là pour chaque $x \in B$
(ou éventuellement on écrit $B \subset X$);

(I) $x_1, \cdots x_{n_i} \in X \Rightarrow r_i(x_1, \cdots, x_{n_i}) \in X$
avec une telle ligne pour chaque règle $r_i \in R$.

Exemple 2.5 *Selon cette convention, la définition inductive des entiers pairs (de l'exemple 2.4) se note*

(B) $0 \in P$;
(I) $n \in P \Rightarrow n+2 \in P$.

Exemple 2.6 *Soit $\Sigma = \{(,)\}$ l'alphabet constitué de la parenthèse ouvrante et de la parenthèse fermante. L'ensemble $D \subset \Sigma^*$ des parenthésages bien formés, appelé langage de Dyck, est défini inductivement par*
(B) $\epsilon \in D$;
(I) $x \in D \Rightarrow (x) \in D$;
(I) $x, y \in D \Rightarrow xy \in D$.

Notation 2.2 *On préfère parfois écrire une définition inductive sous la forme de* règles de déduction *:*

$$\frac{}{B \subset X} \qquad \frac{x_1 \in X \quad \dots \quad x_{n_i} \in X}{r_i(x_1, \dots, x_{n_i}) \in X}$$

Le principe de cette notation est qu'un trait horizontal — signifie une règle de déduction. Ce qui est écrit au dessus est une hypothèse. Ce qui est écrit en dessous est une conclusion. Si ce qui est au dessus est vide, alors c'est que la conclusion est valide sans hypothèse.

Notation 2.3 *On écrit aussi parfois directement*

$$\frac{}{b} \qquad \frac{x_1 \quad \dots \quad x_{n_i}}{r_i(x_1, \dots, \ x_{n_i})}$$

pour chaque $b \in B$,
ou

$$\frac{}{b \in B} \qquad \frac{x_1 \quad \dots \quad x_{n_i}}{r_i(x_1, \dots, \ x_{n_i})}$$

ou encore plus rarement parfois

$$\frac{}{B} \qquad \frac{x_1 \quad \dots \quad x_{n_i}}{r_i(x_1, \dots, \ x_{n_i})}$$

2.4 Applications

2.4.1 Quelques exemples

Exemple 2.7 ($\mathbb{N}$) *La partie X de $\mathbb{N}$ définie inductivement par*

$$\frac{}{0} \qquad \frac{n}{n+1}$$

n'est autre que $\mathbb{N}$ tout entier.

Exemple 2.8 (Σ^*) *La partie X de Σ^*, où Σ est un alphabet, définie inductivement par*
(B) $\epsilon \in X$;
(I) $w \in X \Rightarrow wa \in X$, *pour chaque $a \in \Sigma$;*

n'est autre que Σ^* tout entier.

Exemple 2.9 (Langage $\{a^n bc^n\}$) *Le langage L sur l'alphabet $\Sigma = \{a, b, c\}$ des mots de la forme $a^n bc^n$, $n \in \mathbb{N}$, se définit inductivement par*

(B) $b \in L$;
(I) $w \in L \Rightarrow awc \in L$.

Exercice 2.5 (corrigé page ??). *Définir inductivement l'ensemble des expressions entièrement parenthésées formées à partir d'identificateurs pris dans un ensemble A et des opérateurs $+$ et $\times$.*

2.4.2 Arbres binaires étiquetés

Reprenons le texte suivant du polycopié de INF421 (version 2010-2011) : "la notion d'arbre binaire est assez différente des définitions d'arbre libre, arbre enraciné et arbre ordonné. Un *arbre binaire* sur un ensemble fini de sommets est soit vide, soit l'union disjointe d'un sommet appelé sa *racine*, d'un arbre binaire appelé *sous-arbre gauche*, et d'un arbre binaire appelé *sous-arbre droit*. Il est utile de représenter un arbre binaire non vide sous la forme d'un triplet $A = (A_g, r, A_d)$."

On obtient immédiatement une définition inductive de l'écriture des arbres binaires étiquetés à partir de ce texte.

Exemple 2.10 (Arbres binaires étiquetés) *L'ensemble AB des arbres binaires étiquetés par l'ensemble A est la partie de Σ^*, où Σ est l'alphabet $\Sigma = A \cup \{\emptyset, (,), ,\}$, définie inductivement par*

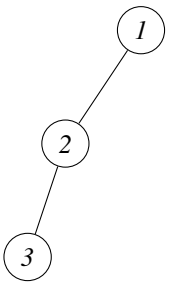
(B) $\emptyset \in AB$;
(I) $g, d \in AB \Rightarrow (g, a, d) \in AB$, *pour chaque $a \in A$.*

Remarque 2.3 *Dans l'écriture plus haut, (g, a, d) désigne la concaténation du mot de longueur 1 $($, du mot g, du mot , de longueur 1, du mot a, du mot , de longueur 1, du mot d et du mot $)$ de longueur 1. Tous ces mots sont bien des mots sur l'alphabet Σ qui contient tous les symboles nécessaires.*

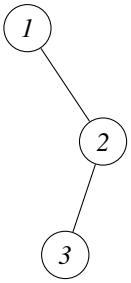
Remarque 2.4 $g, d \in AB \Rightarrow (g, a, d) \in AB$, *pour chaque $a \in A$ désigne le fait que l'on répète, pour chaque $a \in A$, la règle $g, d \in AB \Rightarrow (g, a, d) \in AB$. C'est donc en réalité pas une règle, mais une famille de règles, une pour chaque élément a de A.*

Remarque 2.5 *Attention : un arbre binaire n'est pas un arbre ordonné dont tous les nœuds sont d'arité au plus 2.*

Exemple 2.11 *Par exemple, l'arbre binaire étiqueté*



et l'arbre binaire étiqueté



ne sont pas les mêmes, car le premier correspond au mot $(((\emptyset, 3, \emptyset), 2, \emptyset), 1, \emptyset)$ *et le second au mot* $(\emptyset, 1, ((\emptyset, 3, \emptyset), 2, \emptyset))$*. Pourtant si on considère ces arbres comme des arbres ordonnés, ce sont les mêmes.*

Exercice 2.6 (corrigé page ??). *Soit* A *un alphabet. On définit récursivement la suite d'ensemble* $(AB_n)_{n\in\mathbb{N}}$ *par*

— $AB_0 = \{\emptyset\}$.

— $AB_{n+1} = AB_n \cup \{(a, g, d) | a \in A, g, d \in AB_n\}$

Montrer que $X = \bigcup_{n\in\mathbb{N}} AB_n$ *correspond aussi à l'ensemble* AB *des arbres binaires étiquetés par l'ensemble* A*.*

2.4.3 Expressions arithmétiques

On peut définir les expressions arithmétiques bien formées sur l'alphabet Σ_{exp} de l'exemple **??**. Rappelons que l'on avait défini l'alphabet

$$\Sigma_{exp} = \{0, 1, 2, \cdots, 9, +, -, *, /, (,)\}.$$

Commençons par définir ce qu'est un nombre écrit en base 10. A priori on écrit un entier en base 10 sans débuter par un 0 (sauf pour 0). Par exemple 000192 n'est pas autorisé. Par contre 192 est bien une écriture valide.

On obtient la définition inductive suivante.

Exemple 2.12 *L'ensemble* $\mathcal{N}$ *des nombres entiers non-nuls écrits en base* 10 *est la partie de* ${\Sigma_{exp}}^*$*, définie inductivement par*

(B) $a \in \mathcal{N}$ *pour chaque* $a \in \{1, 2, \ldots, 9\}$ *;*
(I) $g \in \mathcal{N} \Rightarrow ga \in \mathcal{N}$*, pour chaque* $a \in \{0, 1, 2, \ldots, 9\}$.

On peut ensuite définir les expressions arithmétiques de la façon suivante :

Exemple 2.13 *L'ensemble* $Arith$ *des expressions arithmétiques est la partie de* ${\Sigma_{exp}}^*$*, définie inductivement par*
(B) $0 \in Arith$*;*
(B) $\mathcal{N} \subset Arith$*;*
(I) $g, d \in Arith \Rightarrow g + d \in Arith$*;*
(I) $g, d \in Arith \Rightarrow g * d \in Arith$*;*
(I) $g, d \in Arith \Rightarrow g/d \in Arith$*;*
(I) $g, d \in Arith \Rightarrow g - d \in Arith$*;*
(I) $g \in Arith \Rightarrow (g) \in Arith$*;*

Ainsi, on a $(1 + 2 * 4 + 4 * (3 + 2)) \in Arith$ qui correspond bien à une expression valide. Par contre $+1 - /2($ n'est pas dans $Arith$.

2.4.4 Termes

Les termes sont des arbres ordonnés étiquetés particuliers. Ils jouent un rôle essentiel dans beaucoup de structures en informatique.

Soit $F = \{f_0, f_1, \cdots, f_n, \cdots\}$ un ensemble de symboles, appelés *symboles de fonctions*. A chaque symbole f est associé un entier $a(f) \in F$, que l'on appelle *arité de* f et qui représente le nombre d'arguments du symbole de fonction f. On note F_i pour le sous-ensemble des symboles de fonctions d'arité i. Les symboles de fonctions d'arité 0 sont appelés des *constantes*.

Soit Σ l'alphabet $\Sigma = F \cup \{(,), ,\}$ constitué de F et de la parenthèse ouvrante, de la parenthèse fermante, et de la virgule.

Définition 2.2 (Termes sur F**)** *L'ensemble* T *des termes construit sur* F *est la partie de* Σ^* *définie inductivement par :*
(B) $F_0 \subset T$
(c'est-à-dire : les constantes sont des termes)
(I) $t_1, t_2, \cdots, t_n \in T \Rightarrow f(t_1, t_2, \cdots, t_n) \in T$
pour chaque entier n*, pour chaque symbole* $f \in F_n$ *d'arité* n*.*

Remarque 2.6 *Dans la définition plus haute, on parle bien de mots sur l'alphabet* Σ *:* $f(t_1, t_2, \cdots, t_n)$ *désigne le mot dont la première lettre est* f*, la seconde* $($*, les suivantes celle de* t_1*, etc.*

Exemple 2.14 *Par exemple, on peut fixer* $F = \{0, 1, f, g\}$*, avec* 0 *et* 1 *d'arité* 0 *(ce sont des constantes),* f *d'arité* 2 *et* g *d'arité* 1*.*

$f(0, g(1))$ *est un terme sur* F*.* $f(g(g(0)), f(1, 0))$ *est un terme sur* F*.* $f(1)$ *n'est pas un terme sur* F*.*

Les termes sur F correspondent à des arbres ordonnés étiquetés particuliers : les sommets sont étiquetés par les symboles de fonctions de F, et un sommet étiqueté par un symbole d'arité k possède exactement k fils.

2.5 Preuves par induction

On va devoir régulièrement prouver des propriétés sur les éléments d'un ensemble X défini inductivement. Cela s'avère possible en utilisant ce que l'on appelle la *preuve par induction*, parfois appelée *preuve par induction structurelle*, qui généralise le principe de la preuve par récurrence.

Théorème 2.3 (Preuve par induction) *Soit $X \subset E$ un ensemble défini inductivement à partir d'un ensemble de base B et de règles R. Soit $\mathcal{P}$ un* prédicat *exprimant une propriété d'un élément $x \in E$: c'est-à-dire une propriété $\mathcal{P}(x)$ qui est soit vraie soit fausse en un élément $x \in E$.*

Si les conditions suivantes sont vérifiées :

- (B) *$\mathcal{P}(x)$ est vérifiée pour chaque élément $x \in B$;*
- (I) *$\mathcal{P}$ est* héréditaire*, c'est-à-dire stable par les règles de R : Formellement, pour chaque règle $r_i \in R$, pour chaque $x_1, \cdots, x_{n_i} \in E$, $\mathcal{P}(x_1), \cdots, \mathcal{P}(x_{n_i})$ vraies impliquent $\mathcal{P}(x)$ vraie en $x = r_i(x_1, \cdots, x_{n_i})$.*

Alors $\mathcal{P}(x)$ est vraie pour chaque élément $x \in X$.

Démonstration : On considère l'ensemble Y des éléments $x \in E$ qui vérifient le prédicat $\mathcal{P}(x)$. Y contient B par la propriété (B). Y est stable par les règles de R par propriété (I). L'ensemble X, qui est le plus petit ensemble contenant B et stable par les règles de R, est donc inclus dans Y. □

Remarque 2.7 *La preuve par induction généralise bien la preuve par récurrence. En effet, $\mathbb{N}$ se définit inductivement comme dans l'exemple 2.7. Une preuve par induction sur cette définition inductive de $\mathbb{N}$ correspond à une preuve par récurrence, c'est-à-dire aux hypothèses du théorème 2.1.*

Exemple 2.15 *Pour prouver par induction que tous les mots du langage défini inductivement dans l'exemple 2.9 possèdent autant de a que de c, il suffit de constater que c'est vrai pour le mot réduit à une lettre b, qui possède 0 fois la lettre a et la lettre c, et que si cela est vrai pour le mot w alors le mot awc possède aussi le même nombre de fois la lettre a que la lettre c, à savoir exactement une fois de plus que dans w.*

Exercice 2.7 (corrigé page ??). *On considère le sous-ensemble ABS des arbres binaires stricts défini comme le sous-ensemble du langage AB (des arbres binaires étiquettés par A) défini inductivement par :*

(B) *$(\emptyset, a, \emptyset) \in ABS$, pour chaque $a \in A$.*

(I) *$g, d \in ABS \Rightarrow (g, a, d) \in ABS$, pour chaque $a \in A$.*

Montrer

- *qu'un élément de ABS est toujours non-vide et sans sommet avec un seul fils non-vide.*
- *que dans un arbre binaire strict, le nombre de sommets n vérifie $n = 2f - 1$, où f est le nombre de feuilles.*

Exercice 2.8. *Montrer que tout mot du langage de Dyck possède autant de parenthèses fermantes qu'ouvrantes.*

Exercice 2.9. *Montrer que toute expression arithmétique, c'est-à-dire tout mot du langage $Arith$, possède autant de parenthèses fermantes qu'ouvrantes.*

Exercice 2.10. *Un arbre binaire est dit* équilibré *si pour chaque sommet de l'arbre, la différence entre la hauteur de son sous-arbre droit et la hauteur de son sous-arbre gauche vaut soit -1, 0 ou 1 (i.e. au plus un en valeur absolue).*

- *Donner une définition inductive de l'ensemble AVL des arbres binaires équilibrés.*
- *On définit la suite $(u_n)_{n\in\mathbb{N}}$ par $u_0 = 0$, $u_1 = 1$, et pour tout $n \geq 2$, $u_{n+2} = u_{n+1} + u_n + 1$.*
  Montrer que pour tout $x \in AVL$, $n \geq u_{h+1}$ où h et n sont respectivement la hauteur et le nombre de sommets d'un arbre.

2.6 Dérivations

2.6.1 Écriture explicite des éléments : Second théorème du point fixe

Nous avons vu jusque-là plusieurs exemples d'ensembles X définis inductivement. L'existence de chaque ensemble X découle du théorème 2.2, et en fait de l'équation (2.1) utilisée dans la preuve de celui-ci.

On parle de *définition de X par le haut*, puisque l'équation (2.1) définit X à partir de sur-ensembles de celui-ci. Cela a clairement l'avantage de montrer facilement l'existence d'ensembles définis inductivement, ce que nous avons abondamment utilisé jusque-là.

Cependant, cela a le défaut de ne pas dire quels sont exactement les éléments des ensembles X obtenus.

Il est en fait aussi possible de définir chaque ensemble X défini inductivement par *le bas*. On obtient alors une définition explicite des éléments de X, avec en prime une façon de les obtenir explicitement.

C'est ce que dit le résultat suivant :

Théorème 2.4 (Définition explicite d'un ensemble défini inductivement) *Chaque ensemble X défini inductivement à partir de l'ensemble de base B et des règles R s'écrit aussi*

$$X = \bigcup_{n\in\mathbb{N}} X_n,$$

où $(X_n)_{n\in\mathbb{N}}$ est la famille de parties de E définie par récurrence par

- $X_0 = B$
- $X_{n+1} = X_n \cup \{r_i(x_1, \cdots, x_{n_i}) | x_1, \cdots, x_{n_i} \in X_n \text{ et } r_i \in R\}$.

Autrement dit, tout élément de X est obtenu en partant d'éléments de B et en appliquant un nombre fini de fois les règles de R pour obtenir des nouveaux éléments.

Démonstration : Il suffit de prouver que cet ensemble est le plus petit ensemble qui contient B et qu'il est stable par les règles de R.

D'une part, puisque $X_0 = B$, B est bien dans l'union des X_n. D'autre part, si l'on prend une règle $r_i \in R$, et des éléments $x_1, \cdots, x_{n_i}$ dans l'union des X_n, par définition chaque x_j est dans un X_{k_j} pour un entier k_j. Puisque les ensembles X_i sont croissants (i.e. $X_i \subset X_{i+k}$ pour tout k, ce qui se prouve facilement par récurrence sur k), tous les $x_1, \cdots, x_{n_i}$ sont dans X_{n_0} pour $n_0 = \max(k_1, \cdots, k_{n_i})$. On obtient immédiatement que $r(x_1, \cdots, x_{n_i})$ est dans X_{n_0+1}, ce qui prouve qu'il est bien dans l'union des X_n.

Enfin, c'est le plus petit ensemble, car tout ensemble qui contient B et qui est stable par les règles de R doit contenir chacun des X_n. Cela se prouve par récurrence sur n. C'est vrai au rang $n = 0$, car un tel ensemble doit contenir X_0 puisqu'il contient B. Supposons l'hypothèse au rang n, c'est-à-dire X contient X_n. Puisque les éléments de X_{n+1} sont obtenus à partir d'éléments de $X_n \subset X$ en appliquant une règle $r_i \in R$, X contient chacun de ces éléments. □

2.6.2 Arbres de dérivation

La définition *par le bas* de X du théorème précédent invite à chercher à garder la trace de comment chaque élément est obtenu, en partant de X et en appliquant les règles de R.

Exemple 2.16 *Le mot 1+2+3 correspond à une expression arithmétique. En voici une preuve.*

$$\dfrac{\dfrac{1 \in \mathcal{N} \qquad 2 \in \mathcal{N}}{1+2 \in Arith} \qquad 3 \in Arith}{1+2+3 \in Arith}$$

Ce n'est pas la seule possible. En effet, on peut aussi écrire

$$\dfrac{1 \in Arith \qquad \dfrac{2 \in \mathcal{N} \qquad 3 \in \mathcal{N}}{2+3 \in Arith}}{1+2+3 \in Arith}$$

Pour coder chaque trace, la notion naturelle qui apparaît est celle de terme, sur un ensemble de symboles F bien choisi : on considère que chaque élément b de la base B est un symbole d'arité 0. A chaque règle $r_i \in R$ on associe un symbole d'arité n_i. Un terme t sur cet ensemble de symboles est appelé une *dérivation*.

A chaque dérivation t est associé un élément $h(t)$ comme on s'y attend : si t est d'arité 0, on lui associe l'élément b de B correspondant. Sinon, t est de la forme $r_i(t_1, \cdots, t_{n_i})$, pour une règle $r_i \in R$ et pour des termes $t_1, \cdots, t_{n_i}$, et on associe à t le résultat de la règle r_i appliquée aux éléments $h(t_1), \cdots, h(t_{n_i})$.

Exemple 2.17 *Pour les expressions arithmétiques, notons par le symbole $+$ d'arité 2, la règle $g, d \in Arith \Rightarrow g + d \in Arith$;*

La première preuve de l'exemple 2.16 correspond à la dérivation $+(+(1,2),3)$. *La seconde à la dérivation* $+(1,+(2,3))$. *L'image par la fonction* h *de ces dérivations est le mot* $1+2+3$.

On peut alors reformuler le théorème précédent de la façon suivante.

Proposition 2.1 *Soit un ensemble* X *défini inductivement à partir de l'ensemble de base* B *et des règles de* R. *Soit* D *l'ensemble des dérivations correspondant à* B *et à* R. *Alors*

$$X = \{h(t) | t \in D\}.$$

Autrement dit, X est précisément l'ensemble des éléments de E qui possèdent une dérivation.

On voit dans l'exemple précédent, qu'un élément de X peut à priori avoir plusieurs dérivations.

Définition 2.3 *On dit qu'une définition inductive de* X *est* non ambiguë *si la fonction* h *précédente est injective.*

Intuitivement, cela signifie qu'il n'existe qu'une unique façon de construire chaque élément de X.

Exemple 2.18 *La définition suivante de* $\mathbb{N}^2$ *est ambigüe.*

(B) $(0,0) \in \mathbb{N}^2$;
(I) $(n,m) \in \mathbb{N}^2 \Rightarrow (n+1,m) \in \mathbb{N}^2$;
(I) $(n,m) \in \mathbb{N}^2 \Rightarrow (n,m+1) \in \mathbb{N}^2$.

En effet, on peut par exemple obtenir $(1,1)$ *en partant de* $(0,0)$ *et en appliquant la deuxième puis la troisième règle, mais aussi en appliquant la troisième règle puis la deuxième règle.*

Exemple 2.19 *La définition de* $Arith$ *de l'exemple 2.13 est ambigüe puisque* $1+2+3$ *possède plusieurs dérivations.*

Exemple 2.20 *Ce problème est intrinsèque aux expressions arithmétiques, puisque lorsqu'on écrit* $1+2+3$, *on ne précise pas dans l'écriture si l'on veut parler du résultat de l'addition de* 1 *à* $2+3$ *ou de* 3 *à* $1+2$, *l'idée étant que puisque l'addition est associative, cela n'est pas important.*

Exemple 2.21 *Pour éviter ce problème potentiel, définissons l'ensemble* $Arith'$ *des expressions arithmétiques parenthésées comme la partie de* ${\Sigma_{exp}}^*$, *définie inductivement par*

(B) $0 \in Arith$;
(B) $\mathcal{N} \subset Arith'$;
(I) $g, d \in Arith' \Rightarrow (g+d) \in Arith'$;
(I) $g, d \in Arith' \Rightarrow (g*d) \in Arith'$;
(I) $g, d \in Arith' \Rightarrow (g/d) \in Arith'$;
(I) $g, d \in Arith' \Rightarrow (g-d) \in Arith'$;
(I) $g \in Arith' \Rightarrow (g) \in Arith'$;

Cette fois, $1+2+3$ *n'est pas un mot de* $Arith'$. *Par contre,* $(1+(2+3)) \in Arith'$ *et* $((1+2)+3) \in Arith'$.

L'intérêt de cette écriture est que l'on a cette fois des règles non-ambigües.

2.7 Fonctions définies inductivement

Nous aurons parfois besoin de définir des fonctions sur des ensembles X définis inductivement. Cela peut se faire facilement lorsque X admet une définition non ambigüe.

Théorème 2.5 (Fonction définie inductivement) *Soit $X \subset E$ un ensemble défini inductivement de façon non ambigüe à partir de l'ensemble de base B et des règles R. Soit Y un ensemble.*

Pour qu'une application f de X dans Y soit parfaitement définie, il suffit de se donner :

(B) *la valeur de $f(x)$ pour chacun des éléments $x \in B$;*

(I) *pour chaque règle $r_i \in R$, la valeur de $f(x)$ pour $x = r_i(x_1, \cdots, x_{n_i})$ en fonction de la valeur x_1, ..., x_{n_i}, $f(x_1)$, ..., et $f(x_{n_i})$.*

Autrement dit, informellement, si l'on sait "programmer récursivement", c'est-à-dire "décrire de façon récursive la fonction", alors la fonction est parfaitement définie sur l'ensemble inductif X.

Démonstration : On entend par l'énoncé, qu'il existe alors une unique application f de X dans Y qui satisfait ces contraintes. Il suffit de prouver que pour chaque $x \in X$, la valeur de f en x est définie de façon unique. Cela se prouve facilement par induction : c'est vrai pour les éléments $x \in B$. Si cela est vrai en $x_1, \cdots, x_{n_i}$, cela est vrai en $x = r_i(x_1, \cdots, x_{n_i})$: la définition de X étant non ambigüe, x ne peut être obtenu que par la règle r_i à partir de $x_1, \cdots, x_{n_i}$. Sa valeur est donc parfaitement définie par la contrainte pour la règle r_i. □

Exemple 2.22 *La fonction factorielle $Fact$ de $\mathbb{N}$ dans $\mathbb{N}$ se définit inductivement par*

(B) $Fact(0) = 1$;

(I) $Fact(n+1) = (n+1) * Fact(n)$.

Exemple 2.23 *La hauteur h d'un arbre binaire étiqueté se définit inductivement par*

(B) $h(\emptyset) = 0$;

(I) $h((g, a, d)) = 1 + \max(h(g), h(d))$.

Exemple 2.24 *La valeur v d'une expression arithmétique de $Arith'$ se définit inductivement par (v est une fonction qui va des mots vers les rationnels)*

(B) $v(0) = 0$;

(B) $v(x) = h(x)$ *pour* $x \in \mathcal{N}$;

(I) $v((g + d)) = v(g) + v(d)$;

(I) $v((g * d)) = v(g) * v(d)$;

(I) $v((g/d)) = v(g)/v(d)$, *si* $v(d) \neq 0$;

(I) $v((g - d)) = v(g) - v(d)$;

(I) $v((g)) = v(g)$;

où h est la fonction qui à un mot de $\mathcal{N}$ associe sa valeur en tant que rationnel : h se définit inductivement par

(B) $h(a) = a$ *pour chaque* $a \in \{1, 2, \ldots, 9\}$;

(I) $h(ga) = 10 * h(g) + a$ *pour chaque* $a \in \{0, 1, 2, \ldots, 9\}$.

On observera que ces définitions ne sont essentiellement que la traduction de comment on peut les programmer de façon récursive. L'utilisation d'une définition non-ambigüe évitant toute ambiguïté sur l'évaluation.

Remarque 2.8 *Pour les expressions arithmétiques,* $1+2*3 \in Arith$ *est aussi ambigüe à priori. Un évaluateur qui prendrait en entrée un mot de* $Arith$ *devrait aussi gérer les priorités, et comprendre que* $1 + 2 * 3$ *n'est pas le résultat de la multiplication de* $1 + 2$ *par* 3*. En utilisant la définition de* $Arith'$*, on évite complétement aussi cette difficulté, puisque les expressions codent explicitement comment les évaluer, et une définition inductive devient possible. A priori, la valeur d'une expression de* $Arith$ *ne se définit pas aussi simplement inductivement, ne serait-ce qu'en raison du problème que la valeur de* $x + y * z$ *ne s'obtient pas directement à partir de celle de* $x + y$ *et de* z *uniquement.*

2.8 Notes bibliographiques

Lectures conseillées Pour aller plus loin sur les notions évoquées dans ce chapitre, nous suggérons la lecture de [Arnold and Guessarian, 2005]. Pour une présentation plus générale des définitions inductives et des théorèmes de points fixes, nous suggérons la lecture de [Dowek, 2008].

Bibliographie Ce chapitre a été rédigé grâce à l'ouvrage [Dowek, 2008] et à l'ouvrage [Arnold and Guessarian, 2005].

Index

Bibliographie

[Arnold and Guessarian, 2005] Arnold, A. and Guessarian, I. (2005). *Mathématiques pour l'informatique*. Ediscience International.

[Dowek, 2008] Dowek, G. (2008). *Les démonstrations et les algorithmes*. Polycopié du cours de l'Ecole Polytechnique.