Aspects mathématiques des langages de programmation

Syntaxe et sémantique

Adrien Guatto
guatto@irif.fr

29 avril 2024

Avant-propos

Ces notes proposent une introduction à la théorie des langages de programmation. Elles sont en cours de rédaction, et contiennent indubitablement leur lot de fautes et d'erreurs plus ou moins graves. Merci de signaler celles-ci à *guatto@irif.fr*. Merci également de ne pas redistribuer le document.

1. Introduction

Ces notes visent à fournir une introduction à la théorie des langages de programmation. Elles prennent donc le sujet à son début. Toutefois, leur objectif est d'offrir un chemin aussi direct que possible aux questions de recherche contemporaines. Comme beaucoup de ces recherches, elles adoptent le point de vue de l'informatique mathématique. On y étudiera donc des langages de programmation très idéalisés, épurés de leurs caractéristiques contingentes. Cette idéalisation éloigne nécessairement des aspects les plus pratiques de l'activité de programmation. En contrepartie, elle donne accès à toute la précision et à toute la rigueur permises par les mathématiques.

Un langage de programmation est un langage symbolique dont les phrases sont obtenues par l'application répétée d'une collection d'opérations formelles fixée. Ses phrases, appelées programmes, sont équipés d'une mécanique d'exécution plus ou moins abstraite qui définit leur sens. Ce sens contribue notamment à déterminer quels programmes doivent être considérés comme équivalents. L'étude mathématique des langages de programmation se rapproche donc de l'algèbre, à ceci près que les opérations offertes par un langage de programmation prennent des formes plus riches que celles mises en jeu lors de la définition de la structure de groupe, d'anneau, de corps et des autres objets traditionnels du cours d'algèbre. La perspective adoptée au sein des présentes notes est donc celle de l'algèbre universelle, sous-discipline des mathématiques dédiée à l'étude du format général des structures algébriques plutôt qu'à celle d'une famille de structures algébriques particulière.

Le contenu des notes est découpé en trois parties. La première partie constitue une introduction au pendant le plus informatique de l'algèbre universelle classique conçue pour être compatible avec les développement ultérieurs. La seconde partie traite du système T, un prototype de langage fonctionnel dont tous les programmes terminent. La troisième partie traite du langage PCF, un langage fonctionnel dont les programmes peuvent diverger.

Ces notes s'adressent à des étudiants de quatrième ou cinquième année d'université ayant suivi un cours de logique mathémathique de premier cycle. La connaissance d'un langage de programmation fonctionnel est également la bienvenue. Les questions plus informatiques, notamment d'implémentation, seront mentionnées lorsqu'elles sont susceptibles d'éclairer les développements mathématiques. Nous espérons ainsi aider lectrices et lecteurs informaticiens à s'appuyer sur leurs intuitions pratiques.

Première partie

Généralités mathématiques

2. Rappels de logique

L'objectif de ce chapitre est de rappeler dans un style informel les opérations de base nécessaires à la construction des mathématiques, telles que nécessaires dans les chapitres suivants. Il n'a pas pour ambition de se substituer à un cours de logique formelle ou de fondements des mathématiques, bien que ces sujets puissent être abordés d'une manière étonamment proche de l'étude des langages de programmation.

2.1. Le langage informel de la logique et des preuves

La définition et la manipulation des objets mathématiques passe par l'énonciation de *propositions*. D'un point de vue formaliste, une proposition est un énoncé écrite dans un langage symbolique complètement codifié. La grammaire du langage détermine les propositions de base, ainsi que les règles permettant de construire de nouvelles propositions. Ces règles de construction sont les *connecteurs* logiques bien connus, comme la conjonction. Elles sont gouvernées par d'autres règles qui caractérisent la forme des preuves valides d'une proposition donnée.

Au delà de son contenu logique, une proposition porte sur les objets qui intéressent le mathématicien. Ces objets peuvent être de différentes *sortes*. Ils sont soumis à certaines *opérations* et *prédicats* logiques. Les opérations et prédicats sont liées par une collection de propositions admises appelées *axiomes*. Un système de sortes, opérations, relations et axiomes forme une *théorie axiomatique*. Par exemple, la géométrie euclidienne peut être présentée comme une théorie axiomatique où l'on manipule des points, des droites et des plans liés par les relations d'inclusion, d'incidence et de congruence. Ses axiomes expriment par exemple que pour tous deux points il existe une droite qui les contient tous les deux.

Depuis la première moitié du XX^{e} siècle, il semble que la vaste majorité des mathématiciens aient adopté une théorie axiomatique particulière, la théorie de Zermelo-Fraenkel avec axiome du choix, en abrégé ZFC. Cette théorie s'est révélée très expressive, au sens où la plupart des objets mathématiques usuels peuvent être construits en utilisant les objets primitifs de ZFC. Les objets de ZFC sont des "ensembles", et certains axiomes de cette théorie reflètent effectivement nos intuitions pré-mathématiques vis-à-vis des collections de choses rencontrées dans la vie ordinaire. D'autres axiomes sont toutefois d'une signification nettement moins intuitive. Par analogie avec l'informatique, on peut donc penser à ZFC comme à une théorie "de bas niveau" : à l'image du langage machine des processeurs, c'est un langage difficile à manier mais au dessus duquel il est possible de bâtir les abstractions qui constitueront des couches de plus haut niveau. Comme en informatique, l'existence d'une couche de bas niveau partagée assure la compatibilité des couches supérieures ; ainsi, un théorème d'analyse peut être utilisé en arithmétique

sans difficulté si besoin est, sans modification de sa preuve. Cette approche garantit la cohérence des mathématiques, non pas au sens logique de l'absence de contradiction, mais au sens où les développements au sein d'un certain pan de mathématiques sont directement utilisable dans d'autres, puisque tous sont ultimement traduisibles dans le langage de ZFC, sans changer de théorie axiomatique.

Le reste de ce chapitre contient un rappel des règles élémentaires de la logique, ainsi que des constructions ensemblistes usuelles. Ces dernières sont présentées dans un style informel, les règles précises de la théorie des ensembles de Zermelo-Fraenkel dépassant de loin le cadre de ces notes.

2.2. Propositions et prédicats

L'objectif de cette section est de faciliter la rédaction des démonstrations, et de clarifier les règles autorisées pour l'étudiant qui manquerait de bagage mathématique. Pour ce faire, on va adopter un point de vue très mécanique, et donc très détaillée, de l'activité démonstratoire, ramenée à un petit jeu de règles élémentaires. Il est évident qu'une preuve lisible n'explicite pas l'utilisation de chacune des règles, mais cherche plutôt à mettre en valeur les étapes essentielles. Toutefois, avoir en tête l'existence de ces règles, en particulier lorsqu'on est peu familier avec la pratique des mathématiques, peut aider à rédiger et surtout à éviter les erreurs de raisonnement.

À tout moment durant une preuve, le mathématicien cherche à démontrer un *but* formé d'une proposition appelée *conclusion* et d'un nombre fini de propositions présupposées appellées *hypothèses*. Les règles qui suivent permettent de progresser en remplaçant le but courant par une collection finie de nouveaux buts. Éventuellement, on atteint un point où cette collection de nouveaux buts est vide, ce qui signifie que la preuve est terminée. Les règles sont de plusieurs natures : certaines sont purement structurelles

2.2.1. Règles structurelles

Ces règles sont indépendantes du choix des connecteurs, et décrivent la gestion des hypothèses. Elles sont au nombre de deux. La première stipule qu'un but φ est immédiatement démontrable si φ apparaît également comme hypothèse. La seconde stipule qu'il est toujours possible d'introduire une nouvelle hypothèse φ pour prouver une conclusion ψ. On doit pour cela commencer par prouver φ sous la même liste d'hypothèse.

2.2.2. Connecteurs

Pour chaque connecteur, on donne les règles de démonstration qui permettent de l'*introduire*, c'est-à-dire de prouver un but dont il forme la racine de la conclusion, et de l'*éliminer*, c'est-à-dire d'utiliser une hypothèse dont il forme la racine.

2.2.2.1. Conjonction.

Étant donnée deux propositions φ et ψ, on peut former leur *conjonction*, notée $\varphi \wedge \psi$. Pour prouver $\varphi \wedge \psi$, il suffit de prouver φ et ψ. Pour prouver la conclusion τ en utilisant l'hypothèse $\varphi \wedge \psi$, on prouve la conclusion τ sous les hypothèses additionnelles φ et ψ.

2.2.2.2. Disjonction.

Étant donnée deux propositions φ et ψ, on peut former leur *disjonction*, notée $\varphi \vee \psi$. Pour prouver $\varphi \vee \psi$, il suffit de prouver φ, ou bien de prouver ψ. Pour prouver la conclusion τ en utilisant l'hypothèse $\varphi \vee \psi$, on prouve indépendamment la conclusion τ sous l'hypothèse φ et sous l'hypothèse ψ.

2.2.2.3. Implication.

Étant donnée deux propositions φ et ψ, on peut former leur *implication*, notée $\varphi \implies \psi$. Pour prouver $\varphi \implies \psi$, il suffit de prouver ψ sous l'hypothèse φ. Pour prouver le but τ en utilisant l'hypothèse $\varphi \wedge \psi$, on prouve le but φ sous les mêmes hypothèses, puis on prouve τ sous l'hypothèse additionnelle ψ.

2.2.2.4. Trivialité.

La proposition *triviale* est notée $\top$. Prouver le but $\top$ est immédiat. Aucune règle ne permet d'utiliser une hypothèse $\top$.

2.2.2.5. Contradiction.

La proposition *contradictoire* est notée $\bot$. Aucune règle ne permet de l'introduire. Tout but τ est prouvé immédiatement sous l'hypothèse $\bot$.

Remarque 2.2.1. Une théorie axiomatique qui démontre $\bot$ sans hypothèse est dite *contradictoire*. Dans ce cas, la règle citée ci-dessus entraîne immédiatement que la théorie démontre toute proposition.

2.2.2.6. Négation.

La *négation* d'une formule φ est notée $\neg\varphi$. La négation n'est pas un connecteur primitif mais est définie par $\neg\varphi \overset{def}{\iff} \varphi \implies \bot$. L'avantage de cette approche est qu'elle ne nécessite pas d'introduire de nouvelles règles. On peut dériver les règles suivantes à partir de cette définition. Pour introduire $\neg\varphi$, il faut démontrer $\bot$ sous l'hypothèse additionnelle φ. Pour prouver le but ψ en présence de l'hypothèse $\neg\varphi$, il suffit de démontrer la conclusion φ.

2.2.3. Le tiers exclu

Enfin, on ajoute aux règles ci-dessus l'axiome suivant, appelé *tiers exclu* :

$$\text{pour toute proposition } \varphi, \text{ on a } \varphi \vee \neg\varphi.$$

En la présence de cet axiome, appelé *tiers exclu*, on peut dériver les résultats bien connus de logique classique, notamment l'involutivité de la négation

$$\neg\neg\varphi \quad \iff \quad \varphi,$$

la formule bien connue de l'implication

$$\varphi \implies \psi \quad \iff \quad \neg\varphi \vee \psi,$$

ainsi que les lois de distributivité dites de *de Morgan* reproduites ci-dessous.

$$\neg(\varphi \wedge \psi) \iff \neg\varphi \vee \neg\psi \qquad \neg(\varphi \vee \psi) \iff \neg\varphi \wedge \neg\psi \qquad \neg\top \iff \bot \qquad \neg\bot \iff \top$$

L'involutivité de la négation permet notamment le *raisonnement par l'absurde*. Celui-ci prend la forme de la règle de preuve suivante : pour prouver une conclusion φ, il suffit de supposer $\neg\varphi$ et de dériver une contradiction, c'est-à-dire de prouver la conclusion $\bot$.

2.2.4. Prédicats

La logique pure telle que décrite précédemment ne suffit pas pour construire les mathématiques. On lui ajoute donc une notion d'*objets* définie axiomatiquement par un jeu d'opérations et de prédicats, et sur lesquels il est possible de quantifier logiquement. Les opérations et prédicats autorisés dépendent de la théorie axiomatique adoptée. Dans ces notes, comme il est d'usage en mathématiques, on adoptera une théorie des ensembles informelle. Nos objets seront donc des ensembles notés A, B, C et parfois t, u, v.

2.2.4.1. Égalité.

Le prédicat d'égalité entre deux individus, noté $=$, est le plus basique de tous, et joue un rôle un peu particulier. La seule façon d'introduire ce prédicat est via la proposition $A = A$, prouvable sous n'importe quelles hypothèses. On peut l'utiliser via une règle logique ad-hoc appelée *transport* : dès lors qu'on dispose d'une hypothèse de la forme $A = B$, alors on peut remplacer A par B et B par A n'importe où dans le but courant, que ce soit dans les hypothèses ou la conclusion. En particulier, on peut dériver la symétrie et la transitivité de l'égalité à partir de la règle de transport.

2.2.4.2. Appartenance.

La théorie des ensembles est une théorie axiomatique dénuée d'opérations et munie d'un seul prédicat, hormis l'égalité. Il s'agit du prédicat binaire appelé *appartenance* et noté $t \in A$. Intuitivement, il signifie que t est un élément de A. Ce prédicat est soumis à un certain nombre d'axiomes qui permettent de manipuler efficacement la notion d'ensemble, et qui ont permis de bâtir le reste des mathématiques. On décrit un certain nombre d'entre eux plus bas.

Remarque 2.2.2. Le prédicat d'appartenance relie des objets de même nature, c'est-à-dire que t et A sont tous deux des ensembles. Autrement dit, les éléments d'un ensemble sont d'autres ensembles. Ainsi, en théorie des ensembles, l'ensemble des entiers naturels, le nombre pi, la droite réelle, ou l'entier 42 sont tous des ensembles. Remarquons la similarité avec les langages informatiques de bas niveau où toutes les données manipulées sont vues comme des suites de bits. Cette économie de concepts a la conséquence contre-intuitive de permettre l'expression de prédicats dénués de sens intuitifs, comme par exemple $\mathbb{N} \in \pi$. Il est même possible que ces prédicats soient valides, selon le codage employé.

2.2.4.3. Quantification.

Étant donné un ensemble A et une proposition $\varphi(x)$ qui mentionne x, on peut former les propositions $\forall x \in A, \varphi(x)$ et $\exists x \in A, \varphi(x)$. On peut introduire la proposition $\forall x \in A, \varphi(x)$ en démontrant $\varphi(x)$ sous l'hypothèse d'existence d'un nouvel individu $x \in A$. On peut utiliser la proposition $\forall x \in A, \varphi(x)$ en hypothèse en spécifiant un objet $t \in A$, ce qui fournit l'hypothèse $\varphi(t)$. On peut introduire la proposition $\exists x \in A, \varphi(x)$ en spécifiant un objet $u \in A$ et en démontrant la conclusion $\varphi(u)$. On peut utiliser la proposition $\exists x \in A, \varphi(x)$ en spécifiant un objet $u \in A$ et en démontrant la conclusion $\varphi(u)$. Symétriquement, on peut utiliser la proposition $\exists x \in A, \varphi(x)$ pour obtenir un objet $x \in A$ et une nouvelle hypothèse $\varphi(x)$. À noter, on ne sait rien sur l'objet x en dehors du fait qu'il satisfait $\varphi(x)$.

Remarque 2.2.3. Formellement, la quantification n'est pas restreinte aux éléments d'un ensemble fixé. Sa forme primitive est $\forall x, \varphi(x)$, où x quantifie donc sur tous les ensembles. Toutefois, cet usage est moins intuitif et n'est utile que lors de certaines constructions d'usage assez rare dont on signalera les quelques apparitions dans ces notes. La quantification $\forall x \in A, \varphi(x)$ que nous employons ci-dessus est dite *bornée* et constitue simplement un raccourci pour $\forall x, x \in A \implies \varphi(x)$. De même pour $\exists x \in A, \varphi(x)$ qui abrège $\exists x, x \in A \wedge \varphi(x)$.

2.2.4.4. Inclusion.

Le prédicat d'*inclusion*, noté $A \subseteq B$, n'est pas primitif mais défini comme suit à partir du prédicat d'appartenance.

$$A \subseteq B \overset{def}{\iff} \forall x \in A, x \in B.$$

2.2.5. Quelques axiomes ensemblistes

On rappelle ici quelques axiomes et résultats élémentaires de théorie des ensembles. L'exposé complet des axiomes de ZFC est hors.

Extensionalité. L'axiome d'*extensionalité* exprime que deux ensembles sont égaux si et seulement s'ils ont les mêmes éléments. Autrement dit, il exprime que deux ensembles A

et B sont égaux si $A \subseteq B$ et $B \subseteq A$. Notons que la direction "seulement si" est une conséquence automatique du transport d'égalité.

Compréhension. L'axiome de *compréhension* exprime qu'étant donné un ensemble A et un prédicat φ sur les éléments de A, il existe un ensemble A' tel que $x \in A'$ si et seulement si $x \in A$ et $\varphi(x)$. L'axiome d'extensionalité implique que cet ensemble est unique. On le note $\{x \in A \mid \varphi(x)\}$.

Remarque 2.2.4. Contrairement à la quantification, la compréhension est nécessairement restreinte aux éléments d'un ensemble fixé. En l'absence de cette restriction, on obtient facilement une théorie paradoxale. Le plus célèbre de ces paradoxes est dû à Bertrand Russel. Supposons qu'on puisse définir un ensemble en compréhension sans restriction sur le domaine de variation de x. On définit alors $\Omega \coloneqq \{x \mid x \notin x\}$ et on raisonne par cas sur $\Omega \in \Omega$ en utilisant le tiers-exclu pour dériver une contradiction. Si $\Omega \in \Omega$, alors par définition $\Omega \notin \Omega$, ce qui est contradictoire. De même, si $\Omega \notin \Omega$, alors $\Omega \in \Omega$. Dans les deux cas, on aboutit à une contradiction.

Ensemble des parties. L'axiome de l'*ensemble des parties* exprime que pour tout ensemble A, il existe un ensemble A' tel que les éléments de A' sont les parties de A. L'axiome d'extensionalité implique que cet ensemble est unique. On le note $\mathbb{P}A$.

Réunion. L'axiome de la *réunion* exprime que pour tout ensemble d'ensembles A, il existe un ensemble A' tel qu'un élément de A' est un élément d'un élement de A. L'axiome d'extensionalité implique que cet ensemble est unique. On le note $\bigcup A$. Lorsque l'ensemble A ne comprend que deux éléments A_1 et A_2, on écrit $A_1 \cup A_2$ pour sa réunion.

Infini. Il existe un ensemble infini. Sans énoncer précisément le sens de cet axiome, il entraîne l'existence de l'ensemble des entiers naturels, noté $\mathbb{N}$. On reviendra sur les propriétés de cet ensemble ultérieurement.

Quelques ensembles utiles. On définit l'ensemble vide, noté $\emptyset$, par compréhension comme $\{x \in \mathbb{N} \mid \bot\}$. On obtient un ensemble à un seul élément, noté $\mathbb{1}$, comme $\mathbb{P}\emptyset$. Son seul élement est donc $\emptyset$, que l'on notera $*$ dans ce contexte pour éviter toute confusion. À son tour, l'ensemble $\mathbb{P}\mathbb{1}$ a deux éléments, à savoir $\emptyset$ et $\{*\}$. Par convention, on note le premier élement 0 et le second 1, ce qui revient à définir l'ensemble $\mathbb{B}$ des booléens comme $\mathbb{P}\mathbb{1}$.

Étant donnés deux ensembles A et B, on peut définir leur produit cartésien $A \times B$ comme l'ensemble de toutes les paires (a, b) telles que $a \in A$ et $b \in B$.

Remarque 2.2.5. La définition du produit cartésien présuppose la possibilité de former les paires ordonnés, qui est assurée par les axiomes de ZFC que nous avons omis.

2.3. Relations et fonctions

2.3.1. Définitions de base

Définition 2.3.1 (Relation)**.** Une *relation* R est un triplet (A, B, G) où A et B sont des ensembles et G est un sous-ensemble du produit cartésien $A \times B$.

On appelle respectivement *domaine* et *codomaine* de R l'ensemble A et l'ensemble B. Par extension, on appelle *relation de A dans B* une relation de domaine A et codomaine B, et on écrit $R : A \nrightarrow B$ pour signifier que R est une telle relation. On note $\mathbf{Rel}(A, B)$ l'ensemble de toutes les relations de A dans B.

Définition 2.3.2 (Composition des relations)**.** Soient $R : A \nrightarrow B$ et $S : B \nrightarrow C$ deux relations. La *composée* de R et S, notée $R\,;S$, est la relation de domaine A et codomaine C définie par

$$R\,;S \coloneqq \{(x, z) \in A \times C \mid \exists y \in B, (x, y) \in R \wedge (y, z) \in S\}.$$

Définition 2.3.3 (Relation identité)**.** La relation *identité* sur un ensemble A, notée Id_A, est la relation de domaine et codomaine A définie par

$$Id_A \coloneqq \{(x, x') \in A \times A \mid x = x'\}.$$

La composition des relations est associative et admet la relation identité comme élément neutre à gauche et à droite, c'est-à-dire que

$$R\,;(S\,;T) = (R\,;S)\,;T \qquad \text{et} \qquad R\,;Id_B = R = Id_A\,;R$$

pour toutes relations $R : A \nrightarrow B$, $S : B \nrightarrow C$ et $T : C \nrightarrow D$.

Définition 2.3.4 (Relation pleine et relation vide)**.** Soient A et B deux ensembles. La relation *pleine* et la relation *vide* de A dans B, notées respectivement $\mathbf{1}_{A,B}$ et $\mathbf{0}_{A,B}$, sont définies comme suit.

$$\mathbf{1}_{A,B} \coloneqq A \times B \qquad\qquad \mathbf{0}_{A,B} \coloneqq \emptyset$$

La relation vide est absorbante pour la composition, c'est-à-dire que pour toute relation $R : A \nrightarrow B$ et tout ensemble C on a $\mathbf{0}_{C,A}\,;R = \mathbf{0}_{C,B}$ et $R\,;\mathbf{0}_{B,C} = \mathbf{0}_{A,C}$.

Définition 2.3.5 (Relation réciproque)**.** La *réciproque* d'une relation $R : A \nrightarrow B$, notée R°, est la relation de domaine B et codomaine A définie par

$$R^\circ \coloneqq \{(y, x) \in A \times B \mid (x, y) \in R\}.$$

La réciproque, qui renverse une relation, est une opération essentielle. Elle satisfait une propriété importante de *fonctorialité*, c'est-à-dire de compatibilité avec la composition. Cette propriété signifie que pour tous A, B, C et toutes relations $R : A \nrightarrow B$ et $S : B \nrightarrow C$, on a $(R\,;S)^\circ = S^\circ\,;R^\circ$ et $Id_A^\circ = Id_A$. De plus, la réciproque est involutive, c'est-à-dire que $R^{\circ\circ} = R$ pour toute relation $R : A \nrightarrow B$.

Les relations de domaine A et codomaine B fixées sont des parties du produit cartésien $A \times B$, et sont donc ordonnées par l'inclusion. Il est clair que la relation vide est la plus petite relation de A dans B, et la relation pleine la plus grande. La composition des relations est croissante, au sens où si $R_1 \subseteq R_2$ et $S_1 \subseteq S_2$ sont des relations de A dans B, alors $R_1 ; S_1 \subseteq R_2 ; S_2$. De même pour la réciproque : on aura $R_1^\circ \subseteq R_2^\circ$ dès lors que $R_1 \subseteq R_2$. On reviendra au caractère ordonné des relations au chapitre 3.

Soit $(R_i)_{i\in I}$ et $(R_j)_{j\in J}$ deux familles de relations avec $R_i : A \rightarrow\!\!\!\!\!\rightarrow B$ pour tout $i \in I$ et $S_j : B \rightarrow\!\!\!\!\!\rightarrow C$ pour tout $j \in J$. On démontre facilement l'égalité

$$(\bigcup_{i\in I} R_i) ; (\bigcup_{j\in J} S_j) \quad = \quad \bigcup_{i\in I}\bigcup_{j\in J} R_i ; S_j$$

par manipulation des quantificateurs existentiels. En revanche, la composition ne distribue pas sur l'intersection en général. Par exemple, considérons les relations $R = \{(a, b_1), (a, b_2)\}$ ainsi que $S = \{(b_1, c)\}$ et $T = \{(b_2, c)\}$. La relation $S \cap T$ est vide et par conséquent $R ; (S \cap T)$ aussi, tandis que $(R ; S) \cap (R ; T)$ est $\{(a, c)\}$. La composition étant croissante, on a tout de même l'inclusion $R ; (S \cap T) \subseteq (R ; S) \cap (R ; T)$ en général.

2.3.2. Fonctions

On dira qu'une relation $R : A \rightarrow\!\!\!\!\!\rightarrow A$ est *réflexive* si elle contient la relation identité sur A et *co-réflexive* si elle est contenue dans la relation identité. Notons qu'une relation co-réflexive sur un ensemble A est essentiellement la même chose qu'une partie de A.

Définition 2.3.6. Une relation $R : A \rightarrow\!\!\!\!\!\rightarrow B$ est :

- *surjective* si $R^\circ ; R$ est réflexive ;
- *injective* si $R ; R^\circ$ est co-réflexive ;
- *entière* si $R ; R^\circ$ est réflexive ;
- *déterministe* si $R^\circ ; R$ est co-réflexive ;

Si on développe les définitions ci-dessus pour aboutir à une définition explicite en termes de quantificateurs, on obtient ce qui suit.

$$\begin{aligned}
R \text{ est surjective} &\iff \forall y \in B, \exists x \in A, (x, y) \in R \\
R \text{ est injective} &\iff \forall x, x' \in A, (\exists y \in B, (x, y) \in R \wedge (x', y) \in R) \implies x = x' \\
R \text{ est entière} &\iff \forall x \in A, \exists y \in B, (x, y) \in R \\
R \text{ est déterministe} &\iff \forall y, y' \in B, (\exists x \in A, (x, y) \in R \wedge (x, y') \in R) \implies y = y'
\end{aligned}$$

L'intérêt de la formulation relationelle apparaît cependant rapidement dans les preuves, comme l'illustre la propriété qui suit dans un cas très simple.

Propriété 2.3.1. *Si $R : A \rightarrow\!\!\!\!\!\rightarrow B$ et $S : B \rightarrow\!\!\!\!\!\rightarrow C$ sont des relations surjectives (resp. injectives, entières, déterministes) alors leur composée $R ; S$ est surjective (resp. injective, entière, déterministe).*

Démonstration. On ne traite que du cas de la surjectivité, les trois autres étant parfaitement similaires. Supposons que R et S sont surjectives. On a alors

$$\begin{aligned}
(R\,;S)^\circ\,;R\,;S &= S^\circ\,;R^\circ\,;R\,;S && \text{fonctorialité de la réciproque}\\
&\subseteq S^\circ\,;Id_B\,;S && R \text{ surjective}\\
&= S^\circ\,;S && \text{neutralité de l'identité}\\
&\subseteq Id_C && S \text{ surjective}
\end{aligned}$$ □

La preuve précédente a implicitement utilisé l'associativité de la composition relationnelle, qui permet d'omettre les parenthèses lors des séquences de composées, ainsi que sa croissance. Cet usage est typique et nous ne le remarquerons plus.

Du point de vue de ce cours, les fonctions constituent un cas particulier des relations. Plus précisément, une fonction est une relation qui lie tout élément de son domaine à un unique élément de son codomaine, comme l'exprime formellement la définition qui suit. On peut également caractériser la propriété duale dite de « co-fonctionnalité », qui consiste à associer à tout élément du codomaine un unique élément du domaine.

Définition 2.3.7. Une relation $R : A \nrightarrow B$ est :

- *fonctionnelle* si elle est entière et déterministe ;
- *co-fonctionnelle* si elle est surjective et injective ;
- *bijective* si elle est fonctionnelle et co-fonctionnelle.

On peut également décrire la (co)-fonctionnalité avec des quantificateurs. Il est avantageux d'utiliser le quantificateur $\exists!x \in A, \varphi(x)$, qui est une notation pour

$$\exists!x \in A, \varphi(x) \quad \overset{def}{\iff} \quad \exists x \in A, \big(\varphi(x) \wedge \forall x' \in A, \varphi(x') \Rightarrow x' = x\big)\,.$$

On peut alors expliciter les définitions et obtenir les formules attendues.

$$\begin{aligned}
R \text{ est fonctionnelle} &\iff \forall x \in A, \exists!y \in A, (x,y) \in R\\
R \text{ est co-fonctionnelle} &\iff \forall y \in B, \exists!x \in A, (x,y) \in R
\end{aligned}$$

On parlera souvent de *fonction*, *co-fonction* ou *bijection* plutôt que de relation fonctionnelle ou co-fonctionnelle ou bijective. Par fidélité envers la terminologie la plus répandue, on appellera respectivement *surjection* et *injection* les fonctions surjectives et les fonctions injectives. De plus, on appelera *fonction partielle* une relation déterministe mais pas nécessairement entière — une fonction partielle n'est donc pas, à proprement parler, une fonction. On parlera parfois de *fonction entière* pour insister sur le fait qu'une fonction est définie sur tout son domaine A.

La propriété 2.3.1 implique que les fonctions, fonctions partielles, co-fonctions, bijections, surjections et injections sont closes par composition. On dénotera souvent les fonctions par f, g ou h plutôt que R, S ou T. On écrit $f : A \to B$ lorsque f est une fonction de A dans B. On note $\mathbf{Ens}(A, B)$ ou B^A l'ensemble des fonctions de A dans B.

Propriété 2.3.2. *Soit $R : A \nrightarrow B$ une relation. Il existe au plus une relation $S : B \nrightarrow A$ telle que $R ; S$ soit réflexive et $S ; R$ co-réflexive. Qui plus est, une telle relation existe si et seulement si R est fonctionnelle.*

Démonstration. Soient $R : A \nrightarrow B$ et $S_1, S_2 : B \nrightarrow A$ des relations telles que $Id \subseteq R ; S_i$ et $S_i ; R \subseteq Id$ pour $i \in \{1, 2\}$. On prouve que S_1 et S_2 sont égales par extensionalité. On a $Id \subseteq R ; S_1$ et donc $S_2 = S_2 ; Id \subseteq S_2 ; R ; S_1 \subseteq Id ; S_1 = S_1$. On prouve $S_2 \subseteq S_1$ par un raisonnement identique, donc S_1 et S_2 sont égales.

Si R est fonctionnelle, alors R° satisfait les hypothèses du théorème par définition. Réciproquement, supposons que $S : B \nrightarrow A$ satisfasse les hypothèses du théorème. On prouve facilement par manipulation des quantificateurs que $S = R^\circ$, et donc que la relation R est fonctionnelle. □

Propriété 2.3.3. *Soit R une relation surjective et déterministe de A dans B et soient S et T des relations de B dans C. Si $R ; S$ est incluse dans (resp. égale à) $R ; T$ alors S est incluse dans (resp. égale à) T.*

Démonstration. Supposons $R ; S \subseteq R ; T$. Soit (y, z) dans S. Puisque S est déterministe, il existe au moins un x dans A tel que (x, y) soit dans R. On voit que (x, z) appartient à $R ; S$, et donc à $R ; T$. Par définition, il doit exister un certain y' tel que (x, y') soit dans R et (y', z) soit dans T. Mais puisque R est déterministe, on doit avoir $y = y'$. Donc (y, z) est dans T. On vient de montrer $S \subseteq T$. On en déduit en particulier que si $R ; S = R ; T$, alors on a l'inclusion de S dans T et réciproquement, et donc que $S = T$. □

2.3.3. Éléments et parties

Comme on l'a vu dans la preuve précédente, les relations sont un concept très utile pour manipuler commodément les définitions ensemblistes en minimisant la manipulation des quantificateurs. Elles permettent également de s'abstraire dans une certaine mesure de la notion de partie, puisque toute partie B d'un ensemble A définit une relation notée $\langle B \rangle$ de $\mathbb{1}$ dans A par $\langle B \rangle \coloneqq \{(*, x) \in \mathbb{1} \times A \mid x \in B\}$. Réciproquement, toute relation $R : \mathbb{1} \nrightarrow A$ définit une partie de A par $\{x \in A \mid (*, x) \in R\}$. Cette correspondance définit une bijection entre $\mathbf{Rel}(\mathbb{1}, A)$ et $\mathbb{P}A$, tout comme entre $\mathbb{P}A$ et $\mathbf{Rel}(A, \mathbb{1})$.

La relation $\langle B \rangle$ est toujours entière, puisque $*$ est le seul élément de l'ensemble $\mathbb{1}$. Elle est déterministe si et seulement si B est de la forme $\{x\}$. La bijection entre $\mathbf{Rel}(\mathbb{1}, A)$ et $\mathbb{P}A$ se restreint à une bijection entre $\mathbf{Ens}(\mathbb{1}, A)$ et A. Autrement dit, les éléments de A sont en bijection avec les fonctions de $\mathbb{1}$ dans A.

Dans la suite de ces notes, par abus de terminologie, on appellera indifféremment « partie » de A un sous-ensemble de A et une relation $1 \nrightarrow A$, quitte à utiliser implicitement la bijection susmentionnée. En particulier, on écrira parfois $A : \mathbb{1} \nrightarrow A$ pour l'ensemble A considéré comme partie de lui-même, relation qui n'est autre que la relation pleine $\mathbf{1}_{\mathbb{1},A}$.

Il nous arrivera aussi de confondre volontairement les élements de A et les fonctions de $\mathbb{1}$ dans A. Par exemple, si x est un élément de A, on note $f(x)$ pour $x ; f$, abbréviation de $\langle \{x\} \rangle ; f$. On note que $(f ; g)(x) = g(f(x))$.

Définition 2.3.8. L'*image directe* d'une partie $P : \mathbb{1} \nrightarrow A$ de A le long d'une relation $R : A \nrightarrow B$ est la partie de B définie par $P\,;R$. L'*image inverse* d'une partie $Q : \mathbb{1} \nrightarrow B$ de B le long de R est la partie de A définie comme l'image directe de Q le long de R°.

On peut facilement reformuler les définitions ci-dessus pour obtenir une définition classique avec quantificateurs, plus explicite mais aussi plus longue. Ainsi, l'image directe de $P \subseteq A$ le long de $R : A \nrightarrow B$ est simplement le sous-ensemble

$$\{y \in B \mid \exists x \in P, (x, y) \in R\} \subseteq B.$$

Similairement, l'image inverse de $Q \subseteq B$ le long de $R : A \nrightarrow B$ est le sous-ensemble

$$\{x \in A \mid \exists y \in Q, (x, y) \in R\} \subseteq A.$$

2.3.4. Réindexation d'une relation par des fonctions

Soient $R : C \nrightarrow D$ une relation et $f : A \to C$ et $g : B \to D$ des fonctions. On appelle *réindexation de R par (f, g)* la relation de A dans B notée $R[f \mid g]$ et définie par

$$R[f \mid g] \coloneqq \{(x, y) \in A \times B \mid (f(x), g(y)) \in R\}.$$

Cette opération est fonctorielle, au sens où pour toute relation S de C dans D telle que $R \subseteq S$, on a

$$R[f \mid g] \subseteq S[f \mid g].$$

De plus, $R[Id \mid Id] = R$.

Soient $R : A_3 \nrightarrow B_3$ une relation et $f_1 : A_1 \to A_2$, $f_2 : A_2 \to A_3$, $g_1 : B_1 \to B_2$ et $g_2 : B_2 \to B_3$ des fonctions. On vérifie facilement l'égalité

$$R[f_2 \mid g_2][f_1 \mid g_1] \quad = \quad R[f_1\,; f_2 \mid g_1\,; g_2].$$

2.3.5. Bijections et isomorphismes

Les bijections jouent un rôle important en mathématiques, puisqu'elles permettent de donner un sens précis à l'idée intuitive de taille d'un ensemble. Deux ensembles sont *isomorphes* lorsqu'il existe une bijection de A dans B. Autrement dit, deux ensembles sont isomorphes lorsqu'on peut mettre leurs éléments en correspondance un-à-un. En particulier, deux ensembles finis A et B sont isomorphes si et seulement s'ils ont le même nombre d'éléments.

Remarque 2.3.1. Deux ensembles isomorphes sont parfois dits *équipotents*.

Propriété 2.3.4. *Une relation f est bijective si et seulement si f et f° sont toutes deux fonctionnelles.*

Démonstration. Par définition, une relation R est surjective si et seulement si R° est entière, et injective si et seulement si R° est déterministe. □

Si f est bijective, alors f° est fonctionnelle, et $f\,;f^\circ$ tout comme $f^\circ\,;f$ sont égales à l'identité. La réciproque est vraie.

Propriété 2.3.5. *Soient $f : A \to B$ et $g : B \to A$ deux fonctions telles que $f\,;g = Id_A$ et $g\,;f = Id_b$. Alors f est une bijection et $g = f^\circ$.*

Démonstration. Par la propriété 2.3.2, on doit avoir $g = f^\circ$ et $f = g^\circ$. En particulier, g est co-fonctionnelle, et f est donc une bijection. □

2.3.6. Exercices

* **Ex. 1** — Donner un exemple de relation entre ensembles finis qui soit uniquement : surjective ; injective ; entière ; déterministe. Donner un exemple de relation qui soit fonctionnelle mais pas co-fonctionnelle, puis co-fonctionnelle mais pas fonctionnelle. Donner les deux bijections de $\mathbb{B}$ distinctes.

* **Ex. 2** — Démontrer que si $R : A \rightarrowtail B$ est une relation surjective et déterministe, alors pour toutes relations S et T dans $\mathbf{Rel}(B, C)$, si $R\,;S \subseteq R\,;T$ alors $S \subseteq T$. En déduire que $R\,;S = R\,;T$ entraîne $S = T$.

* **Ex. 3** — Démontrer que si $R : A \rightarrowtail B$ est une relation injective et entière, alors pour toutes relations S et T dans $\mathbf{Rel}(C, A)$, si $S\,;R \subseteq T\,;R$ alors $S \subseteq T$. En déduire que $S\,;R = T\,;R$ entraîne $S = T$.

* **Ex. 4** — Démontrer que pour tout ensemble I on a $\left(\bigcup_{i\in I} R_i\right)^\circ = \bigcup_{i\in I} R_i^\circ$.

* **Ex. 5** — Soient $R : A \rightarrowtail B$ et $S_1, S_2 : B \rightarrowtail C$ des relations.

1. A-t-on $R\,;(S_1 \cap S_2) \subseteq (R\,;S_1) \cap (R\,;S_2)$? Si oui, le démontrer. Si non, donner un contre-exemple.
2. Même question pour $(R\,;S_1) \cap (R\,;S_2) \subseteq R\,;(S_1 \cap S_2)$.

2.3.7. Extensions et relèvements relationnels

Soient $R : A \rightarrowtail B$ et $S : A \rightarrowtail C$ deux relations. Il peut arriver qu'on cherche une relation $U : C \rightarrowtail B$ telle que $S\,;U = R$. Cependant, une telle relation n'existe pas forcément, par exemple si R n'est pas vide mais que S l'est. On va donc relâcher un peu la propriété désirée en remplaçant l'égalité par l'inclusion de $S\,;U$ dans R en recherchant la plus grande relation U qui vérifie cette inclusion. Elle existe toujours, et peut facilement être définie en utilisant la quantification universelle.

Définition 2.3.9. Soit $R : A \rightarrowtail B$ et $S : A \rightarrowtail C$ des relations. L'*extension* de R le long de S, notée $S \rhd R$, est la relation de C dans B définie par

$$S \rhd R \quad \coloneqq \quad \{(z, y) \in C \times B \mid \forall x \in A, (x, z) \in S \implies (x, y) \in R\}.$$

Une question voisine consiste à fixer $R : A \rightarrowtail B$ et $T : B \rightarrowtail C$ et à chercher la plus grande relation $U : A \rightarrowtail B$ telle que $U\,;R \subseteq T$. On y répond de manière similaire.

Définition 2.3.10. Soit $R : A \nrightarrow B$ et $T : C \nrightarrow B$ des relations. Le *relèvement* de R le long de T, notée $R \lhd T$, est la relation de A dans C définie par

$$R \lhd T \quad \coloneqq \quad \{(x, z) \in A \times C \mid \forall y \in B, (z, y) \in T \implies (x, y) \in R\}.$$

Dans ce qui suit, l'extension et le relèvement sont plus prioritaires que la composition, et une expression comme $S ; S \rhd R$ signifie donc $S ; (S \rhd R)$.

Soit P une partie d'un ensemble A et S est une endorelation de A. On peut considérer P comme une relation de $\mathbb{1}$ dans A et calculer la relation $P \lhd S$ de $\mathbb{1}$ dans A. Celle-ci, vue commme une partie de A, est caractérisée par

$$x \in P \lhd S \quad \iff \quad \forall x' \in A, (x, x') \in S \implies x' \in P.$$

Autrement dit, $P \lhd S$ décrit la partie de A dont les éléments sont tels que toutes leurs images par S sont dans P. Dualement, si l'on considère P comme une relation de A dans $\mathbb{1}$ et qu'on calcule la relation $S \rhd P$ de A dans $\mathbb{1}$, on obtient

$$x \in S \rhd P \quad \iff \quad \forall x' \in A, (x', x) \in S \implies x' \in P.$$

On voit que $S \rhd P$ décrit la partie de A dont les éléments sont tels que tous leurs antécédents par S sont dans P. Puisque les antécédents par S sont les images par S°, l'observation précédente s'écrit formellement $P \lhd S^\circ = S^\circ \rhd P^\circ$. Il s'agit d'une propriété générale, non restreinte aux parties d'un ensemble fixé.

Propriété 2.3.6. *On a $(S \rhd R)^\circ = R^\circ \lhd S^\circ$ et $(R \lhd T)^\circ = T^\circ \rhd R^\circ$.*

Soient $R : A \nrightarrow B$, $S : A \nrightarrow C$ et $T : C \nrightarrow B$ des relations. Le théorème suivant caractérise extensions et relèvements par une propriété universelle. On le démontre facilement par manipulation des quantificateurs.

Théorème 2.3.1. *On a $T \subseteq S \rhd R$ si et seulement si $S ; T \subseteq R$ si et seulement si $S \subseteq R \lhd T$.*

Corollaire 2.3.1. *On a $S ; S \rhd R \subseteq R$ et $R \lhd T ; T \subseteq R$.*

Corollaire 2.3.2. *On a $T \subseteq S \rhd (S ; T)$ et $S \subseteq (S ; T) \lhd T$.*

Corollaire 2.3.3. *Soit $R' : A \nrightarrow B$ une relation telle que $R \subseteq R'$.*

1. *Pour toute relation $S' : A \nrightarrow C$ telle que $S \subseteq S'$ on a $S' \rhd R \subseteq S \rhd R'$.*
2. *Pour toute relation $T' : B \nrightarrow C$ telle que $T \subseteq T'$ on a $R \lhd T' \subseteq R' \lhd T$.*

Démonstration. Soient $R \subseteq R' : A \nrightarrow B$ et $S \subseteq S' : A \nrightarrow C$. On a

$$R \subseteq R' \implies S' ; S' \rhd R \subseteq R' \implies S ; S' \rhd R \subseteq R' \iff S' \rhd R \subseteq S \rhd R'.$$

On démontre similairement la deuxième partie de l'énoncé. □

2.3.8. Complémentaire

Le *complémentaire* d'une relation $R : A \nrightarrow B$ est la relation de même domaine et codomaine notée $\overline{R}$ et définie comme

$$\overline{R} \coloneqq \{(a,b) \in A \times B \mid (a,b) \notin R\}.$$

Le complémentaire est donc le pendant relationnel de la négation logique. Il s'agit d'une opération décroissante, au sens où si R est incluse dans R' alors $\overline{R'}$ est incluse dans $\overline{R}$. Elle interagit avec les autres opérations relationnelles, en particulier avec la réciproque.

Théorème 2.3.2 (De Morgan)**.** *Les opérations relationnelles vérifient les équations ci-dessous, valides pour toutes relations R, S, T des domaines et codomaines attendus.*

$$\overline{\mathbf{0}} = \mathbf{1} \qquad \overline{\mathbf{1}} = \mathbf{0} \qquad \overline{R \cup S} = \overline{R} \cap \overline{S} \qquad \overline{R \cap S} = \overline{R} \cup \overline{S} \qquad \overline{\overline{R}} = R \qquad \overline{R^\circ} = \overline{R}^\circ$$

$$\overline{R\,;S} = R^\circ \rhd \overline{S} = \overline{R} \lhd S^\circ \qquad \overline{S \rhd R} = S^\circ\,;\overline{R} = \overline{R}^\circ\,;S \qquad \overline{R \lhd T} = \overline{R}\,;T^\circ = T\,;\overline{R}^\circ$$

Démonstration. Les équations de la première ligne sont obtenues par manipulation élémentaire des connecteurs logiques à l'aide des lois bien connues de De Morgan. Les cas de la composition, du relèvement et de l'extension ont une forme différente, produite par la possibilité de réaliser un choix dans l'expression retenue pour l'implication qui caractérise soit le rélèvement soit l'extension. On explicite ces choix ci-dessous.

$$\begin{aligned}
\overline{R\,;S} &= \{(x,z) \in A \times C \mid \forall y \in B, (x,y) \notin R \vee (y,z) \notin S\} = R^\circ \rhd \overline{S} = \overline{R} \lhd S^\circ \\
\overline{S \rhd R} &= \{(y,z) \in B \times C \mid \exists x \in A, (x,y) \in S \wedge (x,z) \notin R\} = S^\circ\,;\overline{R} = \overline{R}^\circ\,;S \\
\overline{R \lhd T} &= \{(x,y) \in A \times B \mid \exists z \in C, (y,z) \in T \wedge (x,z) \notin R\} = \overline{R}\,;T^\circ = T\,;\overline{R}^\circ
\end{aligned}$$ □

La propriété ci-dessous montre que la réciproque d'un extension ou d'un relèvement peut aussi s'exprimer à l'aide du complémentaire, en sus des résultats exprimés à la propriété 2.3.6.

Propriété 2.3.7. *On a $(S \rhd R)^\circ = \overline{R} \rhd \overline{S}$ et $(R \lhd T)^\circ = \overline{T} \lhd \overline{R}$.*

2.4. Constructions ensemblistes

2.4.1. Familles d'ensembles

Définition 2.4.1. Une *famille d'ensembles*, ou *famille*, est une paire $(I, (A_i)_{i \in I})$ formée d'un ensemble I appelé *indices* de la famille et, pour chaque élément i de I, d'un ensemble A_i. Une *famille indicée par I* est une famille dont l'ensemble d'indices est I.

Par abus de langage, on identifie souvent la famille $(I, (A_i)_{i \in I})$ à la collection A notée sans indice, l'ensemble d'indices I étant fixé. Ainsi, on parlera d'une *famille A indicée par I* pour désigner la paire $(I, (A_i)_{i \in I})$.

Il n'est pas immédiat d'interpréter la notation indicée $(A_i)_{i \in I}$ dans la théorie des ensembles formelle. On pourrait être tenté de définir cette collection comme une fonction

de l'ensemble I dans un *ensemble de tous les ensembles*, ou peut-être de *tous les ensembles pas trop gros*. Mais un tel objet n'existe pas nativement dans la théorie des ensembles de Zermelo-Fraenkel. On va donc plutôt identifier la collection $(A_i)_{i\in I}$ d'une famille indicée par I à une paire (E, f) formée d'un ensemble E et d'une fonction f de E dans I. L'ensemble A_i correspond alors au sous-ensemble de E donné par $f^\circ(i)$. On appelle E *l'espace total* de la famille. Les constructions qu'on peut écrire sur l'objet en notation indicée se traduisent naturellement en constructions sur la paire (E, f), si bien qu'on gardera toujours celle-ci implicite.

2.4.2. Produits cartésiens

Définition 2.4.2. Le *produit cartésien* d'une famille A indicée par un ensemble I, noté $\prod_{i\in I} A_i$, est l'ensemble dont les éléments sont les I-uplets, c'est-à-dire les fonctions x qui envoient tout élément i de I dans un élément $f(i)$ dans A_i.

Si x est un élément du produit cartésien de la famille $(I, (A_i)_{i\in I})$, on note x_i la $i$ème composante $x(i)$ de x, qui est un élément de A_i. Pour tout i dans I, on note π_i la fonction de $\prod_{i\in I} A_i$ dans A_i qui envoie x dans x_i.

Il est facile de vérifier que cette construction généralise le produit cartésien binaire habituel. À une paire d'ensemble (A_1, A_2) on associe la famille $(1, 2, (A_i)_{i\in\{1,2\}})$ indicée par l'ensemble $1, 2$. Tout élément x du produit cartésien $\prod_{i\in\{1,2\}} A_i$ donne lieu à un couple (x_1, x_2), et vice-versa. Cette correspondance établit une bijection entre le produit cartésien de la famille et le produit cartésien binaire $A_1 \times A_2$.

Propriété 2.4.1 (Propriété universelle du produit cartésien)**.** *Soit A une famille indicée par un ensemble I une famille d'ensembles, B un ensemble, et $(f_i)_{i\in I}$ une famille de fonctions telle que f_i soit de domaine B et codomaine A_i. Alors il existe une unique fonction h de B dans $\prod_{i\in I} A_i$ telle que $f_i = h\,;\pi_i$.*

Démonstration. La fonction h envoie l'élément y de B dans le I-uplet dont la i-ème composante est $f_i(y)$. On a bien $\pi_i(h(y)) = f_i(y)$ par définition. L'unicité est conséquence de l'extensionalité des fonctions. □

Soient A et B deux familles d'ensembles indicées par I. Soit $(R_i)_{i\in I}$ une famille de relations indicée par I telle que R_i soit de domaine A_i et codomaine B_i. Le produit cartésien de la famille $(R_i)_{i\in I}$ est la relation notée $\prod_{i\in I} R_i$ et définie par

$$\prod_{i\in I} R_i : \prod_{i\in I} A_i \nrightarrow \prod_{i\in I} B_i \quad \coloneqq \quad \{(x, y) \in \prod_{i\in I} A_i \times \prod_{i\in I} B_i \mid \forall i \in I, (x_i, y_i) \in R_i\}.$$

Si toutes les relations R_i sont fonctionnelles, alors le produit cartésien de leur famille l'est également.

2.4.3. Coproduits

Définition 2.4.3. Le *coproduit* d'une famille A indicée par un ensemble I, noté $\coprod_{i\in I} A_i$, est l'ensemble dont les éléments sont les paires (i, x) formées d'un élément i de I et d'un élément a de A_i.

Pour tout i dans I, on note ι_i la fonction de A_i dans $\coprod_{i\in I} A_i$ qui envoie x dans (i,x).

Propriété 2.4.2 (Propriété universelle du coproduit)**.** *Soit A une famille indicée par un ensemble I une famille d'ensembles, B un ensemble, et $(f_i)_{i\in I}$ une famille de fonctions telle que f_i soit de domaine A_i et codomaine B. Alors il existe une unique fonction h de $\coprod_{i\in I} A_i$ dans B telle que $f_i = \iota_i \,;h$.*

Démonstration. La fonction h envoie l'élément (i,x) de $\coprod_{i\in I} A_i$ dans l'élément $f_i(x)$ de B. On a bien $h(\iota_i(x)) = f_i(x)$ est immédiate par définition. L'unicité est conséquence de l'extensionalité des fonctions. □

Soient A et B deux familles d'ensembles indicées par I. Soit $(R_i)_{i\in I}$ une famille de relations indicée par I telle que R_i soit de domaine A_i et codomaine B_i. Le coproduit de la famille $(R_i)_{i\in I}$ est la relation notée $\coprod_{i\in I} R_i$ et définie par

$$\coprod_{i\in I} R_i : \coprod_{i\in I} A_i \nrightarrow \coprod_{i\in I} B_i \quad := \quad \{((i,x),(i,y)) \in \coprod_{i\in I} A_i \times \coprod_{i\in I} B_i \mid i \in I, (x,y) \in R_i\}.$$

Si toutes les relations R_i sont fonctionnelles, alors le coproduit de leur famille l'est également.

2.4.4. Identités ensemblistes remarquables

On rappelle que $\mathbf{Ens}(A,B)$ désigne l'ensemble des fonctions de A dans B.

Propriété 2.4.3. *Pour toute famille A indicée par I et tout ensemble B, on a les isomorphismes suivants.*

$$\mathbf{Ens}(B, \prod_{i\in I} A_i) \cong \prod_{i\in I} \mathbf{Ens}(B, A_i) \qquad \mathbf{Ens}(\coprod_{i\in I} A_i, B) \cong \prod_{i\in I} \mathbf{Ens}(A_i, B)$$

Propriété 2.4.4. *Pour toute famille A indicée par $I \times J$, la fonction*

$$h \quad : \quad \coprod_{i\in I} \prod_{j\in J} A_{i,j} \to \prod_{j\in J} \coprod_{i\in I} A_{i,j} \quad := \quad (i,x) \mapsto (j \mapsto (i, x_j))$$

est un isomorphisme.

On trouvera parfois dans la littérature la notation B^A pour l'ensemble des fonctions de A dans B. Cette notation est particulièrement suggestive si on spécialise les isomorphismes précédents aux cas binaires. On obtient alors les isomorphismes suivants, qui évoquent les identités arithmétiques élémentaires.

Lemme 2.4.1 (Identités ensemblistes remarquables)**.** *Pour tous ensembles A, B et C, on a les isomorphismes suivants.*

$$(A + B) + C \cong A + (B + C) \tag{2.1}$$
$$A + B \cong B + A \tag{2.2}$$
$$A + \mathbb{0} \cong A \tag{2.3}$$
$$(A \times B) \times C \cong A \times (B \times C) \tag{2.4}$$
$$A \times B \cong B \times A \tag{2.5}$$
$$A \times \mathbb{1} \cong A \tag{2.6}$$
$$A \times (B + C) \cong A \times B + A \times C \tag{2.7}$$
$$A \times \mathbb{0} \cong \mathbb{0} \tag{2.8}$$
$$C^{\mathbb{1}} \cong C \tag{2.9}$$
$$\mathbb{1}^C \cong \mathbb{1} \tag{2.10}$$
$$(A \times B)^C \cong A^C \times B^C \tag{2.11}$$
$$C^{\mathbb{0}} \cong \mathbb{1} \tag{2.12}$$
$$\mathbb{0}^C \cong \mathbb{0} \tag{2.13}$$
$$C^{A+B} \cong C^A \times C^B \tag{2.14}$$

2.5. Relations d'équivalence et ensembles quotients

Il arrive fréquemment qu'on tente de définir un ensemble d'objets mathématiques d'une façon qui expose des détails superflus. Par exemple, on pourrait essayer de définir naïvement un nombre rationnel comme une paire d'entiers formée d'un numérateur et d'un dénominateur, mais cela conduirait à distinguer la paire $(1, 2)$ et la paire $(2, 4)$ qui représentent pourtant la même fraction. Autre exemple, on pourrait décréter que, mathématiquement parlant, un programme est un arbre de syntaxe abstraite, mais cela conduit à distinguer l'expression fonctionnelle `fun x -> x` de `fun y -> y` alors qu'elles ne se distinguent que par un choix arbitraire de nom du paramètre.

Pour traiter ce genre de situation, on va imposer aux éléments d'un ensemble une relation dite *d'équivalence* et s'astreindre à ne jamais distinguer entre deux éléments liés par cette relation. La notion d'*ensemble quotient* va nous permettre de combiner l'ensemble et sa relation d'équivalence en un nouvel ensemble qui, informellement parlant, ne contient plus que les éléments qui ne sont *pas* équivalents au sens de la relation fournie. L'objectif de cette section est de développer le minimum de théorie nécessaire à la définition et manipulation de ces notions.

2.5.1. Réflexivité, symétrie, transitivité

On appelle *endorelation* une relation d'un ensemble A dans lui-même. On a vu qu'une telle endorelation $R : A \nrightarrow A$ est réflexive lorsque $Id_A \subseteq R$. On dira qu'elle est *symé-*

trique si $R^\circ \subseteq R$, et qu'elle est *transitive* si $R\,;R \subseteq R$. C'est une *relation d'équivalence* si elle est réflexive, symétrique et transitive.

Si E est une relation d'équivalence sur A, on dit que deux éléments x et y de A sont équivalents par E, ou E-équivalents, lorsque $(x, y) \in E$. Lorsque E peut être déduite du contexte, on parlera souvent d'équivalence plutôt que de E-équivalence.

La relation identité sur A est évidemment une relation d'équivalence. De plus, par réflexivité toute relation d'équivalence sur A contient Id_A. Donc, la relation identité sur A est la plus petite relation d'équivalence sur cet ensemble. La relation pleine $\mathbf{1}_{A,A} = A \times A$ est une relation d'équivalence, et *a fortiori* la plus grande de celles-ci.

Si R est une endo-relation de A et n un entier, on définit la n^e *itérée de* R, notée R^n, comme l'endorelation de R donnée par récurrence sur n par $R^0 = Id_A$ et $R^{n+1} = E^n\,;E$.

Définition 2.5.1 (Clôtures)**.** Soit $R : A \nrightarrow A$.

- La *clôture réflexive* de R est $R^? \coloneqq R \cup Id_A$.
- La *clôture symétrique* de R est $R^\leftrightarrow \coloneqq R \cup R^\circ$.
- La *clôture transitive* de R est $R^+ \coloneqq \bigcup_{n\geq 1} R^n$.
- La *clôture réflexive-transitive* de R est $R^* \coloneqq \bigcup_{n\geq 0} R^n = R^? \cup R^+ = R^{+?}$.
- La *clôture symétrique-réflexive-transitive* de R est $R^= \coloneqq R^{\leftrightarrow *}$.

Les clôtures de R définies ci-dessus sont donc les plus petites relations incluant R et respectivement réflexive, symétrique, transitive et réflexive-transitive. Ces quatre opérations partagent trois propriétés importantes. Si on écrit $F(-)$ pour l'opération qui envoie une endorelation R dans sa clôture réflexive, symétrique, transitive ou bien réflexive-transitive, on peut vérifier que, dans chaque cas :

1. F est *croissante*, au sens où $R \subseteq S$ entraîne $F(R) \subseteq F(S)$;
2. F est *inflationnaire*, au sens où $R \subseteq F(R)$;
3. F est *idempotente*, au sens où $F(F(R)) = F(R)$.

On définira ultérieurement la notion d'*opérateur de clôture*, qui regroupe ces propriétés dans le contexte plus général des ensembles préordonnés ou ordonnés.

Propriété 2.5.1. *La clôture réflexive d'une relation transitive est transitive. La clôture transitive d'un relation réflexive est réflexive. La clôture transitive d'une relation symétrique est symétrique. En particulier, $R^=$ est la plus petite relation d'équivalence qui contient R.*

En revanche, la clôture symétrique d'une relation transitive n'est pas nécessairement transitive. Par exemple, la relation $R = \{(a_1, a_2)\}$ est transitive mais sa clôture symétrique $R^\leftrightarrow = \{(a_1, a_2), (a_2, a_1)\}$ n'est pas transitive puisqu'elle ne contient ni (a_1, a_1) ni (a_2, a_2). On a tout de même l'inclusion qui suit.

Propriété 2.5.2. $R^{*\leftrightarrow} \subseteq R^{\leftrightarrow *}$.

2.5.2. Fonctions respectueuses d'une relation

Une fonction $f : A \to B$ *respecte* une relation R sur A lorsque $f^\circ ; R ; f \subseteq Id$, autrement dit lorsque pour tous (x, x') dans R on a $f(x) = f'(x)$. En utilisant les notations de la section 2.3.4, on peut aussi écrire formellement cette propriété $R \subseteq Id(f(-), f(=))$.

Notons que si $f : A \to B$ respecte R alors pour toute fonction $g : B \to C$ la composée $f ; g : A \to C$ respecte R. C'est la conséquence du fait que toute fonction g respecte l'identité.

Propriété 2.5.3. *Soit f une fonction d'un ensemble A dans un ensemble B et R une endorelation de A. Si f respecte R, alors f respecte la clôture symétrique et la clôture réflexive-transitive de R.*

Démonstration. Supposons $f^\circ ; R ; f \subseteq Id$. On a donc $f^\circ ; R^\circ ; f = f^\circ ; R ; f^\circ \subseteq Id^\circ = Id$, et f respecte la réciproque de R et donc sa clôture symétrique. En ce qui concerne la clôture réflexive-transitive de R, on a

$$\begin{aligned} f^\circ ; R^* ; f \subseteq Id &\Leftrightarrow R^* \subseteq f^\circ \rhd Id \lhd f \\ &\Leftarrow R \cup Id \cup f^\circ \rhd Id \lhd f ; f^\circ \rhd Id \lhd f \subseteq f^\circ \rhd Id \lhd f. \end{aligned}$$

Cette dernière inclusion se scinde en trois inclusions à démontrer. L'inclusion $R \subseteq f^\circ \rhd Id \lhd f$ exprime que f respecte R. L'inclusion $Id \subseteq f^\circ \rhd Id \lhd f$ est conséquence de $f^\circ ; f \subseteq Id$, qui exprime le déterminisme de f. Pour la troisième inclusion, on a

$$\begin{aligned} f^\circ \rhd Id \lhd f ; f^\circ \rhd Id \lhd f \subseteq f^\circ \rhd Id \lhd f &\Leftrightarrow f^\circ ; f^\circ \rhd Id \lhd f ; f^\circ \rhd Id \lhd f ; f \subseteq Id \\ &\Leftarrow Id \lhd f ; f^\circ \rhd Id \subseteq Id \\ &\Leftarrow Id \lhd f ; f ; f^\circ ; f^\circ \rhd Id \subseteq Id \\ &\Leftarrow Id ; Id \subseteq Id \\ &\Leftrightarrow Id \subseteq Id \end{aligned}$$

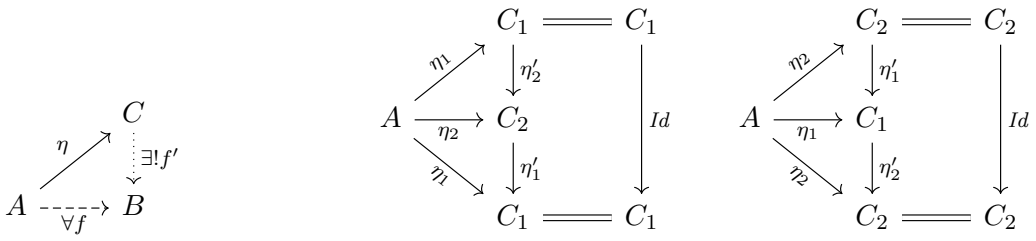
où on a utilisé le fait que f soit entière, c'est-à-dire l'inclusion $Id \subseteq f ; f^\circ$. □

2.5.3. Ensemble quotient

2.5.3.1. Définition abstraite.

Fixons un ensemble A et E une relation d'équivalence sur celui-ci. On s'intéresse à la production d'un ensemble dont les éléments sont, en quelque sorte, les mêmes que ceux de A mais où l'on n'a conservé que les éléments inéquivalents au sens de E. La définition qui suit capture abstraitement cette idée.

Définition 2.5.2. Un *quotient* de A par E est une paire (C, η), où C est un ensemble et η une fonction de A dans C respectant E et telle que pour tout ensemble B et toute fonction $f : A \to B$ qui respecte E, il existe une unique fonction $f' : C \to B$ telle que $f = \eta ; f'$. Cette situation est résumée à la figure 2.1a.



(a) Propriété universelle

(b) Unicité à bijection près

FIG. 2.1. : Quotients

L'énoncé ci-dessus est appelé *propriété universelle* du quotient. On peut y penser comme à un mode d'emploi abstrait : il ne dit pas comment construire un quotient, mais comment l'utiliser. De plus, il caractérise le quotient, comme on va le montrer.

Propriété 2.5.4. *Le quotient de A par E est unique à bijection près.*

Démonstration. Soient (C_1, η_1) et (C_2, η_2) deux quotients de A par E. On va construire une bijection de C_1 dans C_2.

Puisque η_2 est une fonction de A dans C_2 respectant E, et que (C_1, η_1) est un quotient de A par E, on obtient une unique fonction $\eta'_2 : C_1 \to C_2$ telle que $\eta_1 ; \eta'_2 = \eta_2$. On applique le même raisonnement à η_1 pour obtenir une unique fonction $\eta'_1 : C_2 \to C_1$ telle que $\eta_2 ; \eta'_1 = \eta_1$. Or, la fonction $\eta'_2 ; \eta'_1$ vérifie $\eta_1 ; \eta'_2 ; \eta'_1 = \eta_2 ; \eta'_1 = \eta_1$, elle doit donc être égale à l'identité Id_{C_1} par la propriété universelle de C_1. De même pour la fonction $\eta'_1 ; \eta'_2$ qui doit être égale à l'identité Id_{C_2} par la propriété universelle de C_2. On vient donc de construire une bijection (η'_1, η'_2) entre C_1 et C_2. Les deux situations sont présentées sous forme diagrammatique à la figure 2.1b. □

En vertu de ce résultat, on parlera donc *du quotient* de A par E plutôt que *d'un quotient* de A par E, puisqu'on ne distingue jamais des ensembles isomorphes. Toutefois, il nous reste à montrer que le quotient existe toujours, ce qu'on va maintenant faire.

Remarque 2.5.1. La bijection construite dans la preuve de la propriété 2.5.4 préserve la structure de quotient, au sens où $\eta_1 = \eta_2 ; \eta'_1$ et $\eta_2 = \eta_1 ; \eta'_2$. Si (f, g) est une autre bijection de C_1 dans C_2 qui préserve la structure en ce sens, alors on a forcément $f = \eta'_2$ et $g = \eta'_1$ par les propriétés universelles respectives de (C_1, η_1) et (C_2, η_2). On résume parfois cette observation en disant que le quotient de A par E est unique à une *unique* bijection près. Toutefois, cette unicité ne considère que les bijections qui préservent la structure. Par exemple, considérons le quotient de l'ensemble des booléens par la relation identité. Il est facile de montrer que l'ensemble des booléens lui-même, muni de la fonction identité, en est le quotient. Pourtant, il existe deux bijections distinctes de $\mathbb{B}$, à savoir la fonction identité et la négation logique. Seule la première préserve la structure de quotient.

2.5.3.2. Construction de l'ensemble quotient.

Pour montrer qu'un ensemble quotient existe, on commence par définir *classe d'équivalence* par E d'un élément x de A, notée $[x]_E$. Il s'agit de l'ensemble des éléments de A qui sont équivalents à x d'après E.

$$[x]_E \quad \coloneqq \quad \{x' \in A \mid (x, x') \in E\}$$

Remarquons que cet ensemble n'est jamais vide puisque E est réflexive. On va construire un quotient de A par la relation E, noté A/E, en posant

$$A/E \quad \coloneqq \quad \bigcup_{x \in A} \{[x]_E\}.$$

On note $[-]_E$ pour la relation de A dans A/E qui relie tout x de A à tout $P \in A/E$ tel que $x \in P$. Cette relation est surjective par réflexivité de E.

Propriété 2.5.5. *Soient x, y et z des éléments de A tels que x appartienne à la classe d'équivalence de y ainsi qu'à celle de z. Alors $[y]_E = [z]_E$.*

Corollaire 2.5.1. *La relation $[-]_E$ est fonctionnelle.*

Propriété 2.5.6. *Deux éléments de A appartiennent à la même classe d'équivalence si et seulement s'ils sont équivalents.*

Démonstration. Supposons, $x, x' \in [y]_E$. On a alors $(x, y) \in E$ et $(x', y) \in E$. On conclut $(x, x') \in E$ par symétrie et transitivité de E.

Maintenant, supposons que $(x, x') \in E$. Alors $x \in [x']_E$ par définition, et $x' \in [x']_E$ par réflexivité. Donc x et x' appartiennent à la même classe d'équivalence. □

Propriété 2.5.7. *Deux éléments x et y de A sont E-équivalents si et seulement si leurs classes d'équivalence par E sont égales.*

La propriété 2.5.7 exprime que la fonction $[-]_E$ qui envoie tout élement x de A dans sa classe d'équivalence par E respecte E. Il nous reste à démontrer que l'ensemble construit vérifie la propriété universelle est bien un quotient.

Théorème 2.5.1. *La paire $(A/E, [-]_E)$ est un quotient de A par E.*

Démonstration. Soit $f : A \to B$ une fonction respectant E. On doit montrer qu'il existe une unique fonction $f_E : A/E \to B$ telle que $[-]_E ; f_E = f$. On définit la relation

$$f_E : A/E \nrightarrow B \quad \coloneqq \quad \{(C, y) \in A/E \times B \mid \exists x \in C, y = f(x)\}.$$

Cette relation est clairement entière. Elle est déterministe puisque si (C, y_1) et (C, y_2) alors il existe x_1 et x_2 dans C tels que $y_1 = f(x_1)$ et $y_2 = f(x_2)$, mais alors $(x_1, x_2) \in E$ par la propriété 2.5.6, et donc $y_1 = y_2$ puisque f respecte E. Donc f_E est bien une fonction.

Soit x un élément de A. On a $x \in [x]_E$ par réflexivité, donc $([x]_E, f(x)) \in f_E$ et, par définition, $f_E([x]_E) = f(x)$.

Soit $g : A/E \to B$ telle que $[-]_E ; g = f$. On a donc $[-]_E ; g = [-]_E ; f_E$. Mais $[-]_E$ est surjective et déterministe, donc $g = f_E$ par le propriété 2.3.3. □

La manipulation des ensembles quotients est essentielle en mathématique. Ceux-ci permettent par exemple de définir l'ensemble des nombres réels comme classes d'équivalences des suites de Cauchy, l'anneau $\mathbb{Z}/n\mathbb{Z}$ des entiers congrus modulo n, ou encore les expressions d'un langage de programmation modulo l'équivalence induite par la sémantique du langage. Cependant, leur usage peut être laborieux, puisque manipuler l'ensemble quotient exige de prouver que toutes les fonctions respectent la relation d'équivalence. De plus, la définition générique de A/E ne donne aucune information concrète sur la structure des classes d'équivalence. On va voir dans le chapitre suivant qu'il est parfois possible d'éviter la manipulation du quotient en définissant des *formes canoniques*, qui remplacent les classes d'équivalences abstraites par des choix calculables de représentants. C'est là que l'informatique entre en jeu.

2.5.4. Exercices

* **Ex. 6** — Soit $R : A \nrightarrow A$.

1. Démontrer que si R est transitive, alors $\bigcup_{n\geq 1} R$ est incluse dans R.
2. Démontrer que R^+ est transitive.
3. Démontrer que si $S : A \nrightarrow A$ est transitive et contient R alors elle contient R^+.

* **Ex. 7** — Démontrer la propriété 2.5.1.

* **Ex. 8** — Soit A un ensemble quelconque. Donner la plus grande relation d'équivalence sur A. Quel est le quotient de A par cette relation ?

* **Ex. 9** — Soient $R, S : A \nrightarrow B$ deux relations d'équivalence.

1. La relation $R \cap S$ est-elle une relation d'équivalence ? Si oui, le démontrer. Si non, donner un contre-exemple.
2. Même question pour la relation $R \cup S$.

* **Ex. 10** — Démontrer la propriété 2.5.7.

* **Ex. 11** — Soit $R : A \nrightarrow A$. On définit $R^{\leq n}$ comme $(R^?)^n$. Montrer que $R^* = \bigcup_{n\geq 0} R^{\leq n}$.

3. Ensembles ordonnés

3.1. Définitions de base

Une endorelation $R : A \nrightarrow A$ est *anti-symétrique* lorsque $R \cap R^\circ \subseteq Id_A$. En d'autres termes, si $(x, x') \in R$ et $(x', x) \in R$ alors $x = x'$ quels que soient x et x' dans A.

Définition 3.1.1. Une *relation de préordre* sur un ensemble A est une relation $R : A \nrightarrow A$ réflexive et transitive. Une *relation d'ordre* sur A est une relation de préordre sur A qui est antisymétrique.

Par abus de langage, on abrègera utilisera souvent les termes *préordre* et *ordre* pour désigner une relation de préordre ou d'ordre, respectivement.

Définition 3.1.2. Un *ensemble préordonné* X est une paire $(|X|, \leq_X)$ formée d'un ensemble $|X|$ appelé *support* de X et d'une relation de préordre $\leq_X$ sur ce support. L'ensemble est *ordonné* lorsque $\leq_X$ est une relation d'ordre.

Dans le reste de cette sous-section, X, Y et Z désignent des ensembles préordonnés. On écrira $\leq$ plutôt que $\leq_X$ lorsque X peut être déduit du contexte. On écrira souvent $x \leq_X y$ plutôt que $(x, y) \in \leq$; notons que $f(x) \leq_Y g(y)$ est par définition équivalent à $x \leq_X [f \mid g]$ y. Par abus de langage, on tend à identifier un ensemble préordonné ou ordonné et son suport, et on parlera d'un élément de X ou d'une partie de X pour désigner un élément ou une partie de $|X|$.

Exemple 3.1.1. L'ensemble des entiers naturels munis de la relation d'ordre habituelle est un ensemble ordonné. L'ensemble des entiers naturels munis de la relation de divisibilité est un ensemble ordonné distinct du précédent.

Exemple 3.1.2. Soit A un ensemble quelconque. L'ensemble des parties de A ordonnées par l'inclusion forme une ensemble ordonné. L'ensemble des parties de A muni de la relation R définie par $R \coloneqq \{(P, Q) \in \mathbb{P}A \times \mathbb{P}A \mid \exists f : P \to Q, f \text{ injective}\}$ est un ensemble préordonné.

Exemple 3.1.3. Notons A^* l'ensemble des mots finis sur un ensemble A. L'endorelation R sur A^* définie par $w_1 \leq w_2$ si et seulement si la longueur de w_1 est plus petite que celle de w_2 est une relation de préordre.

Exemple 3.1.4. L'exemple 3.1.3 est une instance d'un phénomène plus général. Plus généralement, si $f : B \to A$ est une fonction quelconque et $S : A \nrightarrow A$ une relation de préordre, la relation $S[f \mid f]$ est une relation de préordre sur B. Notons que $S[f \mid f]$ n'est pas antisymétrique en général, même si S l'est !

Définition 3.1.3 (Fonction croissante)**.** Une fonction $f : X \to Y$ est *croissante* si

$$\leq_X \quad \subseteq \quad \leq_Y [f \mid f].$$

autrement dit lorsque pour tous $x, x' \in X$ tels que $x \leq_X x'$ on a $f(x) \leq_Y f(x')$.

Les fonctions croissantes sont aussi appelées des fonctions *croissantes* dans la littérature anglophone [3, p. ex.]. En français ce terme désigne les fonctions qui sont soit croissantes, soit décroissantes. Une fonction $f : X \to Y$ est *décroissante* lorsqu'elle est croissante pour $|X|$ muni de la relation réciproque $\leq_X^\circ$.

Propriété 3.1.1. *La fonction identité $id : X \to X$ est croissante. La composée $f ; g : X \to Z$ de deux fonctions $f : X \to Y$ et $g : Y \to Z$ croissantes est croissante.*

Démonstration. Immédiat par application des résultats de la section 2.3.4. □

3.2. Constructions sur les ensembles préordonnés et ordonnés

3.2.1. Relation d'équivalence induite par un préordre

Pour tout préordre $(|X|, \leq_X)$, la relation $\equiv_X$ définie comme $\leq_X \cap \leq_X^\circ$ est une relation d'équivalence. Il s'agit même de la plus grande relation d'équivalence contenue dans $\leq_X$. De plus, cette relation est compatible avec la relation de préordre, au sens suivant.

Propriété 3.2.1. *On a $\equiv_X ; \leq_X ; \equiv_X \subseteq \leq_X$.*

On appelle parfois les classes d'équivalences de la relation $\equiv_X$ les *composantes fortement connexes* de X. Un graphe peut être vu comme une relation $G : |X| \nrightarrow |X|$ sans propriété particulière. Les classes d'équivalences de la clôture réflexive-transitive de G sont exactement ses composantes fortement connexes.

3.2.2. Relation d'ordre induite par un préordre

Étant donnée une relation de préordre, on peut se demander comment lui associer une relation d'ordre. Il existe en général de nombreuses solutions, et on cherche une solution canonique, c'est-à-dire caractérisée comme la meilleure solution à un certain problème donné. Considérons la définition suivante.

Définition 3.2.1. Soit X un ensemble préordonné. On définit l'ensemble ordonné des *composantes fortement connexes* de X, dénoté $\mathsf{CC}\, X$, comme suit.

- Les éléments de $\mathsf{CC}\, X$ sont les classes d'équivalence d'éléments de $|X|$ pour la relation d'équivalence $\equiv_X$ définie à la section 3.2.1.
- La relation d'ordre $\leq_{\mathsf{CC}\, X}$ entre ces classes d'équivalences est héritée de X, autrement dit $P \leq_{\mathsf{CC}\, X} Q$ dès lors qu'il existe x dans P et y dans Q tels que $x \leq_X y$.

La propriété 3.2.1 assure que la relation posée sur les composantes fortement connexes est bien une relation d'ordre.

Cette construction est canonique au sens où $\mathsf{CC}\, X$ est, informellement, le plus petit ensemble ordonné dans lequel X s'envoie par une fonction croissante. Il s'agit là de la *propriété universelle* de $\mathsf{CC}\, X$, qu'on exprime rigoureusement ci-dessous.

Propriété 3.2.2. *Soit X un ensemble préordonné et Y un ensemble ordonné. Soit $f : X \to Y$ une fonction croissante. Il existe une unique fonction croissante $f^{\#} : \mathsf{CC}\, X \to Y$ telle que $f = \iota; f^{\#}$, où $\iota : X \to \mathsf{CC}\, X$ est la fonction qui envoie x dans $[x]_{\equiv}$.*

Corollaire 3.2.1. *L'ensemble des fonctions croissantes de X dans Y, où Y est un ordre, est en bijection avec l'ensemble des fonctions croissantes de $\mathsf{CC}\, X$ dans Y.*

3.2.3. Ordre dual

La construction suivante est aussi simple qu'utile. Elle permet notamment de ne pas dupliquer certains définitions, comme on le verra ultérieurement.

Définition 3.2.2. L'ensemble préordonné *dual* de X, noté X°, a le même ensemble d'éléments que X et la relation de préordre donnée par $\leq_{X^\circ} = \leq_X{}^s$.

Propriété 3.2.3. *Si $f : X \to Y$ est une fonction croissante alors f est aussi une fonction croissante de X° dans Y°.*

On écrit f° lorsqu'on veut insister sur le fait qu'on voit f comme une fonction de X° dans Y°, même si formellement f et f° sont identiques.

3.2.4. Constructions libres

On se pose la question de munir un ensemble S quelconque d'une relation de préordre. Il existe deux choix canoniques : on peut décider de munir S de la plus petite relation de préordre possible, ou bien de la plus grande possible. Le premier choix correspond au préordre *discret*, le second au préordre *codiscret*, parfois aussia appelé *chaotique*.

Définition 3.2.3. Soit S un ensemble. La relation d'ordre *discrète* (resp. *codiscrète*) sur S est la relation identité sur S (resp. la relation totale sur S, c'est-à-dire $S \times S$). On écrit $\mathsf{Disc}\, S$ (resp. $\mathsf{CoDisc}\, S$) pour l'ensemble préordonné dont les éléments sont ceux de l'ensemble S et la relation de préordre est la relation discrète (resp. codiscrète).

La différence entre ces deux constructions est essentiellement résumée par les propriétés universelles suivantes.

Propriété 3.2.4. *L'ensemble des fonctions de S dans $|X|$ est en bijection avec l'ensemble des fonctions croissantes de $\mathsf{Disc}\, S$ dans X. L'ensemble des fonctions de $|X|$ dans S est en bijection avec l'ensemble des fonctions croissantes de X dans $\mathsf{CoDisc}\, S$.*

On remarque que la relation de préordre discrète est une relation d'ordre. Ce n'est pas vrai de la relation codiscrète, loin d'être antisymétrique. De plus, $\mathsf{CC}(\mathsf{CoDisc}\, S)$ est l'ensemble ordonné trivial a un seul élément ! Cet exemple montre l'intérêt des préordres.

3.2.4.1. Soulèvement.

On voudra fréquemment s'assurer qu'un ensemble ordonne possède un élément *minimal*, c'est-à-dire plus petit que tous les autres.

Définition 3.2.4. L'ensemble préordonné $\uparrow X$ a pour ensemble d'éléments $|\uparrow X| = \langle\,\rangle \uplus \{\langle x\rangle \mid x \in |X|\}$ et pour relation de préordre la plus petite relation de préordre telle que $\langle\,\rangle \leq_{\uparrow X} a$ pour tout $a \in |\uparrow X|$ et $\langle x\rangle \leq_{\uparrow X} \langle x'\rangle$ pour tous $x, x' \in |X|$.

Propriété 3.2.5. *Si X est un ensemble ordonné alors $\uparrow X$ l'est aussi.*

Propriété 3.2.6. *La fonction $\langle -\rangle : X \to \uparrow X$ qui envoie x dans $\langle x\rangle$ est croissante.*

Propriété 3.2.7. *Soit $f : X \to Y$ une fonction croissante. La fonction $\langle f\rangle : \uparrow X \to \uparrow Y$ qui envoie $\langle\,\rangle$ sur $\langle\,\rangle$ et $\langle x\rangle$ sur $\langle f(x)\rangle$ est croissante.*

3.2.4.2. Produits cartésiens.

Définition 3.2.5 (Préordre produit)**.** Le produit cartésien de X et Y, noté $X \times Y$, a pour ensemble d'élément $|X| \times |Y|$, et pour relation d'ordre la relation telle que $(x, y) \leq_{X\times Y} (x', y')$ si et seulement si $x \leq_X y$ et $y \leq_Y y'$.

Propriété 3.2.8. *Si X et Y sont des ensembles ordonnés alors $X \times Y$ l'est aussi.*

Propriété 3.2.9. *La fonction π_1 (resp. π_2) de $|X| \times |Y|$ dans $|X|$ (resp. $|Y|$) qui envoie la paire (x, y) sur l'élément x (resp. y) est croissante.*

Propriété 3.2.10. *Soient $f : Z \to X$ et $g : Z \to Y$ deux fonctions croissantes. La fonction $\langle f, g\rangle : Z \to X \times Y$ qui envoie $z \in Z$ dans $(f(z), g(z)) \in X \times Y$ est croissante.*

On appelle les fonctions π_i les *projections* du produit cartésien, et la fonction $\langle f, g\rangle$ le *pairage* de f et g.

Propriété 3.2.11. *Soient $f : Z \to X$, $g : Z \to Y$ et $h : Z \to X \times Y$ des fonctions croissantes. On a les égalités suivantes.*

$$\pi_1 \circ \langle f, g\rangle = f \tag{3.1}$$
$$\pi_2 \circ \langle f, g\rangle = f \tag{3.2}$$
$$h = \langle \pi_1 \circ h, \pi_2 \circ h\rangle \tag{3.3}$$

Ces équations sont faciles à démontrer, et ne dépendent en réalité pas de la monotonie de f, g et h. On aurait pu énoncer les mêmes pour le modèle ensembliste de T.

3.2.4.3. Préordres fonctionnels.

Définition 3.2.6. Le préordre des fonctions croissantes de X et Y, noté $[X, Y]$, a pour ensemble d'éléments les fonctions croissantes de X dans Y, et la relation d'ordre telle que $f \leq_{[X,Y]} g$ si et seulement si pour tout $x \leq_X x'$ on a $f(x) \leq_Y g(x')$.

Propriété 3.2.12. *Si X et Y sont des ensembles ordonnés alors $[X, Y]$ l'est aussi.*

Propriété 3.2.13. *La fonction dite d'*évaluation *de X dans Y, dénotée $ev_{X,Y} : [X, Y] \times X \to Y$, qui envoie la paire (f, x) sur $f(x)$ est croissante.*

Propriété 3.2.14. *Soit $f : Z \times X \to Y$ une fonction croissante. Pour tout élément z de Z, on dénote $f_z : X \to Y$ la fonction qui envoie z dans $f(z, x)$. On dénote $curry_{X,Y,Z}(f)$ la fonction qui envoie z dans f_z. Alors :*

1. *pour tout élément $z \in X$, $f_z : X \to Y$ est croissante,*
2. *$curry_{X,Y,Z}(f) : Z \to [X, Y]$ est croissante.*

Propriété 3.2.15. *Soient $f : Z \times X \to Y$, $g : Z \to X$ et $h : Z \to [X, Y]$ des fonctions croissantes. On a les égalités suivantes.*

$$ev_{X,Y} \circ \langle curry_{X,Y,Z}(f), g\rangle = f \circ \langle id_Z, g\rangle \tag{3.4}$$

$$h = curry_{X,Y,Z}(ev_{X,Y} \circ \langle h \circ \pi_1, \pi_2\rangle) \tag{3.5}$$

3.2.5. Suprema et infima

On va maintenant s'intéresser à l'existence de propriétés supplémentaires dans un ensemble ordonné ou préordonné. En particulier, l'existence de *plus petits majorants* et *plus grands minorants* fournit une variante de la notion usuelle de limite bien adaptée au cadre des ensembles ordonnés.

Définition 3.2.7 (Majorants, minorants)**.** Soit P une partie d'un préordre X. Un *majorant* de P dans X est un élément x de X tel que tout x' dans P est plus petit que x. Un *minorant* de P dans X est un majorant de P dans X°.

On dit aussi que x *majore* P lorsque x est un majorant (resp. minorant) de P.

Définition 3.2.8 (Suprema, infima)**.** Soit X un préordre et P une partie de X. Un *supremum* (pluriel *suprema*) de P dans X est un élément x_0 de X qui est plus petit que tous les majorants de P dans X. Autrement dit, x_0 est tel que pour tout $x' \in X$, on a

$$(\forall x \in P, x \leq x') \quad \Leftrightarrow \quad x_0 \leq x'.$$

Un *infimum* (pluriel *infima*) de P est un supremum de P dans X°.

Propriété 3.2.16. *Si $P \subseteq X$ a un supremum x_0, alors pour tout $x' \in P$, on a $x' \leq x_0$. De même, si $P \subseteq X$ a un infimum x_0, alors pout tout $x' \in P$, on a $x_0 \leq x'$.*

Propriété 3.2.17. *Si X est un ordre, alors une partie P de X a au plus un supremum (respectivement un infimum), que l'on note $\bigvee P$ (respectivement $\bigwedge P$) lorsqu'il existe.*

Soit $f : X \to Y$ une fonction croissante et P une partie de X. Si la partie $\{f(x) \mid x \in P\}$ de Y a un supremum, on désignera celui-ci par $\bigvee_{x \in P} f(x)$, et similairement pour les infimum.

Il est naturel de s'intéresser à des classes d'ensembles ordonnés disposant de tous les suprema et/ou infima pour les parties d'une certaine forme. Parmi les classes les plus populaires, on trouve des variantes de celle des treillis.

Définition 3.2.9. Soit X un ensemble ordonné. On dit que X est :

- un $\vee$-*demi-treillis* s'il dispose de tous les suprema finis non vides,
- un $\wedge$-*demi-treillis* s'il dispose de tous les infima finis non vides,
- un *treillis* s'il dispose de tous les suprema et infima finis non vides,
- un $\vee$-*demi-treillis borné* s'il dispose de tous les suprema finis,
- un $\wedge$-*demi-treillis borné* s'il dispose de tous les infima finis,
- un *treillis borné* s'il dispose de tous les suprema et infima finis,
- un *treillis complet* s'il dispose de tous les suprema.

Dans la définition ci-dessus, "suprema finis" ou "infima finis" signifie les suprema ou infima des *parties* finies. Plus généralement, le terme "suprema bleus" ou "infima bleus", avec "bleu" une propriété quelconque, désigne des suprema des parties dotées de la propriété en question.

Propriété 3.2.18. *Un treillis complet dispose aussi de tous les infima.*

Les suprema et infima étant parfaitement duaux, on aurait également pu définir les treillis complets en utilisant les infima plutôt que les suprema.

Exemple 3.2.1. L'ensemble des entiers naturels dispose de tous les suprema finis, ainsi que tous les infima finis non vides. Autrement dit, c'est un treillis avec un élément minimal.

Exemple 3.2.2. L'ensemble des nombres réels muni de son ordre habituel a tous les suprema et infima bornés, c'est-à-dire tous les supremas des parties qui ont un majorant et les infima des parties qui ont un minorant.

3.2.6. Exercices

* **Ex. 12** — On considère la relation R entre parties d'un ensemble A définie à l'exemple 3.1.2.

1. Donner un ensemble fini A et deux parties distinctes P, Q de A telles que (P, Q) et (Q, P) soient dans R.
2. Démontrer que R est bien une relation de préordre.
3. Donner une définition de sa réciproque R° qui ne fasse pas référence explicite à R ou à l'injectivité.

4. Supposons que A soit un ensemble fini. Pouvez-vous décrire en termes simples le support et la relation d'ordre de l'ensemble ordonné induit par le préordre R sur les parties de A ?

* **Ex. 13** — Soit X un ensemble préordonné, et soit $\equiv_X$ l'endorelation sur $|X|$ définie par $\leq_X \cap \leq_X^\circ$. Démontrer les énoncés suivants.

1. La relation $\equiv_X$ est une relation d'équivalence.
2. Toute relation d'équivalence sur $|X|$ contenue dans $\leq_X$ est contenue dans $\equiv_X$.
3. La relation $\equiv_X$ est l'identité si et seulement si X est ordonné.

3.3. Théorèmes de point fixe

3.3.1. Généralités

Soit $f : X \to X$ une fonction. Un *point fixe* de f est un élément x de X tel que $f(x) = x$. D'un point de vue mathématique, de tels x peuvent être vus comme les solutions de l'équation exprimée par la fonction f. D'un point de vue informatique, ces points fixes correspondent parfois à la sémantique de programmes récursifs.

Dans ces notes, on va se concentrer aux situations où l'ensemble X est muni d'une relation d'ordre. Dans ce cas, on s'intéressera aussi aux *points préfixes*, c'est-à-dire aux éléments x de X tels que $f(x) \leq x$, ainsi qu'aux *points postfixes*, c'est-à-dire aux éléments x de X tels que $x \leq f(x)$.

Il nous arrivera aussi de nous intéresser à des relations de préordre, auquel cas on relâchera la définition de point fixe pour $f(x) \equiv x$, avec $\equiv$ la relation d'équivalence induite par le préordre. Notons que cette définition est plus générale que la précédente puisqu'elle s'y réduit lorsque le préordre est antisymétrique.

Lemme 3.3.1. *Le plus petit point préfixe de f, s'il existe, est un point fixe.*

Démonstration. Supposons qu'on ait x_0 tel que $f(x_0) \leq x_0$, et tel que pour tout x tel que $f(x) \leq x$ on ait $x_0 \leq x$. Puisque f est croissante, on a $f(f(x)) \leq f(x)$ et $f(x)$ est un point préfixe. Donc $x \leq f(x)$, ce qu'il fallait démontrer. □

Lemme 3.3.2. *Le plus grand point postfixe de f, s'il existe, est un point fixe.*

Démonstration. Par application du lemme 3.3.1 dans l'ordre dual X°. □

Tout point fixe étant a fortiori un point préfixe et un point postfixe, les résultats précédents impliquent qu'un plus petit point préfixe est un plus petit point fixe, et qu'un plus grand point postfixe est un plus grand point fixe. Dans le cas où X est un ordre et que f a un (nécessairement unique) plus petit point fixe, on note celui-ci $\mu(f)$ ou bien $\mu x.f(x)$. Le plus grand point fixe, s'il existe, est lui noté $\nu(f)$ ou bien $\nu x.f(x)$.

On peut écrire les propriétés du plus petit et du plus grand point fixe sous la forme des règles d'inférence qui suivent. Celle-ci ne font que reformuler la définition précédentes

avec des notations plus suggestives.

$$\frac{}{f(\mu x.f(x)) \leq \mu x.f(x)} \qquad \frac{f(a) \leq a}{\mu x.f(x) \leq a} \qquad \frac{}{\nu x.f(x) \leq g(\nu x.f(x))} \qquad \frac{a \leq f(a)}{a \leq \nu x.f(x)}$$

La deuxième et quatrième règles peuvent être lues comme des principes d'induction. On aura l'occasion de les utiliser dans ce sens fréquemment.

3.3.2. Transfert de points fixes

Il est parfois utile de transférer un calcul de point préfixe ou postfixe réalisé dans un certain ensemble préordonné à un autre ensemble préordonné. Deux telles situations vont nous intéresser : celle où on transfère le calcul à un ensemble isomorphe, et celle où on le transfère à un ensemble dual, en un sens qu'on va préciser.

Lemme 3.3.3. *Si f est une bijection monotone d'un ordre X dans un ordre Y, et que x_0 est un plus petit (resp. plus grand) point préfixe (resp. postfixe) d'une endofonction monotone g de X, alors $f(x_0)$ est un plus petit (resp. plus grand) point préfixe (resp. postfixe) de $f^\circ ; g ; f$.*

Démonstration. On prouve uniquement le cas du plus petit point préfixe, celui du plus grand point postfixe étant obtenu de façon parfaitement symétrique.

Soit x_0 un plus petit point préfixe de g dans X. On montre que $f(x_0)$ est un point préfixe de $f^\circ ; g ; f$ dans Y, et que c'est un minimum avec cette propriété.

- Puisqu'on a $x_0 \leq_X h(x_0)$, on a $f(x_0) \leq_Y f(h(x_0)) = f(h(f^\circ(f(x_0))))$ et $f(x_0)$ est un point préfixe de $f^\circ ; h ; f$.
- Soit y dans Y tel que $y \leq_Y f(h(f^\circ(y)))$. Mais alors $f^\circ(y) \leq_X f^\circ(f(h(f^\circ(y)))) = h(f^\circ(y))$ et $f^\circ(y)$ est un point préfixe de h. Donc $x_0 \leq_X f^\circ(y)$ et, de façon équivalente, $f(x_0) \leq_Y y$. □

Enfin, dans certains cas, l'ensemble (pré)ordonné considéré est muni d'une *auto-dualité*, c'est-à-dire d'une fonction croissante $i : X \to X^\circ$ involutive, au sens où $i ; i^\circ = id$. Une telle fonction est nécessairement bijective avec i° pour inverse, puisqu'on a automatiquement $i^\circ ; i = id$. Le résultat suivant, corollaire du précédent, est alors utile.

Lemme 3.3.4. *Soient X un ensemble préordonné, i une auto-dualité sur cet ensemble, et f une fonction croissante sur X. Si x_0 est un plus petit point préfixe de h alors $i(x_0)$ est un plus grand point postfixe de $i^\circ ; h ; i$.*

Démonstration. On applique le lemme 3.3.3 et on obtient que x_0 est un plus petit point préfixe de $i^\circ ; h ; i$ dans X°. Mais un plus petit point préfixe dans X° n'est rien d'autre qu'un plus grand point postfixe dans X, et on obtient le résultat escompté. □

Le résultat précédent peut se formuler de façon équivalente pour traiter les plus grands points postfixes de f, puisque si x_0 est un plus grand point postfixe de f, alors il s'agit d'un plus petit point préfixe de f° sur X°. Mais i° est une auto-dualité de X° et donc, par le lemme 3.3.4, on sait que $i^\circ(x_0)$ est un plus grand point postfixe de $i ; f^\circ ; i^\circ$. On conclut que $i^\circ(x_0) = i^\circ(x_0)$ est un plus petit point préfixe de $i^\circ ; f ; i$ sur X.

3.3.3. Points fixes dans un treillis complet

Supposons que X soit un treillis complet. Les résultats suivants, souvent attribués à Knaster et Tarski, sont très utilisés en sémantique des langages de programmation.

Théorème 3.3.1. *Toute fonction croissante $f : X \to X$ a un plus petit point préfixe.*

Démonstration. Pour montrer l'existence d'un plus petit point préfixe, considérons l'ensemble de tous les points préfixes de f, c'est-à-dire la partie P de X définie par

$$P \quad \coloneqq \quad \{x \in X \mid f(x) \leq x\}.$$

On pose $x_0 \in X \coloneqq \bigwedge P$, qui existe puisque X est un treillis complet. On a donc

$$\begin{array}{lll} \forall x \in P, x_0 \leq x & \implies & \forall x \in P, f(x_0) \leq f(x) \qquad (f \text{ croissante}) \\ & \implies & \forall x \in P, f(x_0) \leq x \qquad (\text{déf. de } P) \\ & \iff & f(x_0) \leq x_0 \qquad (\text{propriété universelle de l'infimum}) \end{array}$$

ce qui montre que x_0 est un point préfixe de f. C'est par définition le plus petit. □

Théorème 3.3.2. *Toute fonction croissante $f : X \to X$ a un plus grand point postfixe.*

Démonstration. Un point postfixe de f est un point préfixe de f°. Or, être un treillis complet est une propriété auto-duale, au sens où X° est un treillis complet si et seulement si X l'est. On conclut immédiatement par application du théorème 3.3.1. □

3.3.3.1. Points fixes dans le treillis complet des relations

On a vu que les parties d'un ensemble forme l'exemple prototypique d'un treillis complet, avec le supremum est donné par l'union et l'infimum par l'intersection. C'est vrai en particulier des parties du produit cartésien $A \times B$, et donc de l'ensemble $\mathbf{Rel}(A, B)$ des relations de domaine A et codomaine B, pour tous A et B. De plus, le treillis complet $\mathbf{Rel}(A, B)$ est muni à la fois d'un isomorphisme avec $\mathbf{Rel}(B, A)$ donné par la réciproque et d'une auto-dualité donnée par le complémentaire. Par les lemmes 3.3.3 et 3.3.4, on a donc les égalités suivantes pour toute endofonction monotone F de $\mathbf{Rel}(A, B)$.

$$(\mu X.F(X))^\circ = \mu X.F(X^\circ)^\circ \tag{3.6}$$
$$(\nu X.F(X))^\circ = \nu X.F(X^\circ)^\circ \tag{3.7}$$
$$\overline{\mu X.F(X)} = \nu X.\overline{F(\overline{X})} \tag{3.8}$$
$$\overline{\nu X.F(X)} = \mu X.\overline{F(\overline{X})} \tag{3.9}$$

3.3.4. Exercices

* Exercice 14 Clôtures

1. Définir la clôture transitive et réflexive-transitive d'une relation comme un plus petit point fixe en utilisant le théorème de Knaster-Tarski.

2. Démontrer que les opérations obtenues sont bien équivalentes aux définitions ad-hoc données au chapitre 2.

** Exercice 15 Autour de Knaster-Tarski

Soit X un treillis complet et $f : X \to X$ une fonction croissante. On note $\mathsf{Fix}(f)$ l'ensemble des points fixes de f muni de la relation d'ordre héritée de X. L'objectif de l'exercice est de démontrer que $\mathsf{Fix}(f)$ est un treillis complet.

1. Expliquer en quoi ce résultat généralise les théorèmes 3.3.1 et 3.3.2.
2. On note $\uparrow x$ l'ensemble $\{x' \in X \mid x \leq x'\}$ muni de la relation d'ordre héritée de X. Montrer que $\uparrow x$ est un treillis complet.
3. Soit P une partie de $\mathsf{Fix}(f)$. On pose $x_0 \coloneqq \bigvee P$. Montrer que f se restreint à une fonction croissante de l'ensemble $\uparrow x_0$ dans lui-même.
4. En déduire que x_0 est le plus petit point fixe de f dans $\uparrow x_0$.
5. En déduire que $\mathsf{Fix}\, f$ est un treillis complet.

3.4. Adjonctions

4. Réécriture abstraite

4.1. Définitions de base

La théorie de la réécriture abstraite concerne les *systèmes de réécriture abstraits*. Un tel système est simplement une endorelation $R : A \nrightarrow A$ sur un ensemble A, sans propriété particulière. La théorie de la réécriture abstraite est donc en quelque-sorte un cas particulier de la théorie générale des relations. On va toutefois introduire une terminologie spécifique et imagée.

Si x est un élément de A, un *réduit immédiat* de x par R est un élément y de A tel que (x, y) appartient à R. Un réduit de x par R est un réduit immédiat de x par R^*. Un élément x de A est *normal* pour R lorsque x n'a aucun réduit immédiat par R. On dit que x' est une *forme normale* de x si x' est un réduit de x et que x' est normal. Une *réduction* est une paire dans R^*, une *expansion* est une paire dans $R^{\circ *}$.

4.2. Confluence

On rappelle que $R^=$ est la plus petite relation d'équivalence qui contient R, c'est-à-dire sa clôture symétrique-réflexive-transitive $R^{\leftrightarrow *}$. Deux éléments sont dans cette relation s'ils peuvent être connectés par une suite finie de réductions et d'expansions. En particulier, si deux éléments de A ont un réduit commun, alors ils sont équivalents au sens de $R^=$. On va s'intéresser aux système de réécriture abstraits où la réciproque est vraie, ce qui permettra de résoudre le problème du mot pour $R^=$.

Définition 4.2.1. Le système de réécriture abstrait R a la *propriété de Church et Rosser* lorsque deux éléments équivalents au sens de $R^=$ ont nécessairement un réduit commun. Autrement dit, R vérifie l'inclusion $R^= \subseteq R^* ; R^{*\circ}$.

Définition 4.2.2. Soient R et S des endorelations d'un même ensemble A. On dit que la relation R est *S-confluente* si $R^\circ ; R \subseteq S ; S^\circ$.

Définition 4.2.3. Un système de réécriture abstrait R :

- est *déterministe* si R est *Id*-confluent,
- a la *propriété du diamant* si R est R-confluent,
- est *localement confluent* si R est R^*-confluent,
- est *globalement confluent* ou simplement *confluent* si R^* est R^*-confluente.

Notons que cette définition d'un système déterministe est cohérente avec la définition d'une relation déterministe donnée au chapitre 2.

Théorème 4.2.1. *Un système de réécriture vérifie la propriété de Church et Rosser si et seulement s'il est confluent.*

Démonstration. Soit $R : A \rightarrow\!\!\!\!\!\rightarrow A$ un système de réécriture abstrait qui vérifie la propriété de Church et Rosser. On a $R^{*\circ} ; R^* = R^{\circ *} ; R^* \subseteq R^{=} \subseteq R^* ; R^{*\circ}$ et donc R est confluent.

Supposons maintenant R confluent, et démontrons que R a la propriété de Church et Rosser, c'est-à-dire l'inclusion $R^{=} \subseteq R^* ; R^{*\circ}$. Par définition de la clôture réflexive-symétrique-transitive, il suffit de montrer les inclusions ci-dessous.

$$Id \subseteq R^* ; R^{*\circ} \qquad R \subseteq R^* ; R^{*\circ} \qquad R^\circ \subseteq R^* ; R^{*\circ} \qquad R^* ; R^{*\circ} ; R^* ; R^{*\circ} \subseteq R^* ; R^{*\circ}$$

Les trois premières sont des conséquences immédiates des définitions, ainsi que du fait que la clôture réflexive-transitive commute à la réciproque. On a

$$\begin{aligned} R^* ; R^{*\circ} ; R^* ; R^{*\circ} &\subseteq R^* ; R^* ; R^{*\circ} ; R^{*\circ} && (R \text{ confluente}) \\ &\subseteq R^* ; R^{*\circ} && (\text{transitivité}) \end{aligned}$$

ce qui démontre la dernière inclusion. □

4.2.1. Outils de preuve

Démontrer la confluence d'un système de réécriture peut être difficile. Raisonner sur la confluence locale est plus facile, et étudier la propriété du diamant encore plus. On dispose d'un certain nombre d'outils et techniques pour ce faire.

Lemme 4.2.1. *Si R a la propriété du diamant, R est confluent.*

Démonstration. Supposons que R a la propriété du diamant. On va démontrer que pour tous entiers naturels n, m, on a $R^{\circ n} ; R^m \subseteq R^m ; R^{\circ n}$. Il est clair que ce résultat entraîne la confluence de R.

On raisonne par récurrence forte sur la somme de n et m. L'hypothèse de récurrence est donc que pour tout couple d'entier naturels (p, q) tel que que $p + q < n + m$, on a $R^{\circ p} ; R^q \subseteq R^q ; R^{\circ p}$. On raisonne par cas sur n et m. Les cas $n = 0$ et $m = 0$ sont immédiats. Considérons donc les cas où $n = n' + 1$ et $m = m' + 1$. On a

$$\begin{aligned} R^{\circ(n'+1)} ; R^{m'+1} &= R^{\circ n'} ; R^\circ ; R ; R^{m'} \\ &\subseteq R^{\circ n'} ; R ; R^\circ ; R^{m'} && (R \text{ a la propriété du diamant}) \\ &\subseteq R ; R^{\circ n'} ; R^{m'} ; R^\circ && (\text{hypothèse de récurrence}) \\ &\subseteq R ; R^{m'} ; R^{\circ n'} ; R^\circ && (\text{hypothèse de récurrence}) \\ &= R^{m'+1} ; R^{\circ n'+1}. \end{aligned}$$

On a appliqué l'hypothèse de récurrence avec $(p, q) = (1, m')$, puis $(p, q) = (n', 1)$ et enfin $(p, q) = (m', n')$. On a bien $p + q < n + m = n' + m' + 2$ à chaque fois. □

Lemme 4.2.2 (Encadrement)**.** *Soient R et S des systèmes de réécritures abstraits tels que $R \subseteq S \subseteq R^*$. Si S a la propriété du diamant, R est confluente.*

Démonstration. Par le lemme 4.2.1, on sait que S est confluente ou, de façon équivalente, que S^* a la propriété du diamant. Mais l'inégalité entraîne $R^* \subseteq S^* \subseteq R^{**} = R^*$, donc $R^* = S^*$ a la propriété du diamant, et donc R est confluente. □

4.3. Normalisation, convergence et divergence

La relation de *normalisation* $\widehat{R}\colon A \nrightarrow A$ associée à R est définie comme

$$\widehat{R}\colon A \nrightarrow A \coloneqq R^* \cap (\mathbf{0} \lhd R).$$

En toutes lettres, la paire (x, x') appartient à $\widehat{R}$ lorsque x' est une forme normale de x. En particulier, il est facile de voir que $\widehat{R}\,;R$ est l'ensemble vide. On va s'intéresser à une condition suffisante sur la relation R qui entraîne que chaque élément de A dispose d'une unique forme normale, autrement dit que $\widehat{R}$ soit fonctionnelle.

Théorème 4.3.1. *Si R est confluente alors $\widehat{R}$ est déterministe.*

Démonstration. Soit x un élément de A et y_1, y_2 deux formes normales de x. Puisque R est confluente, y_1 et y_2 ont un réduit commun, disons z. Mais puisque y_1 est normal, son seul réduit est lui-même, et on doit avoir $y_1 = z$. De même pour y_2, donc $y_1 = z = y_2$. □

La réciproque de ce théorème n'est pas vraie. Par exemple, le système de réécriture abstrait $R \coloneqq \{(a, b_1), (b_i, b_{i+1})_{i\in\mathbb{N}}, (a, c_1), (c_i, c_{i+1})_{i\in\mathbb{N}}\}$ n'est pas confluent bien que sa relation de normalisation $\widehat{R}$ soit vide et donc déterministe. On va voir que le problème est lié à l'existence de chemins infinis dans le graphe de réduction.

Considérons le système de réécriture abstrait $R = \{(a_i, a_{i+1})_{i\in\mathbb{N}}, (a_i, b)_{i\in\mathbb{N}})$. Ce système est confluent et, a fortiori, tous ses éléments ont une unique forme normale, à savoir b. On comprend cependant que trouver algorithmiquement la forme normale de a_0, par exemple, n'est pas simple, puisqu'il faut éviter de choisir la séquence infinie $a_0\, R\, a_1\, R\, a_2\, R \ldots$ qui n'atteint jamais b.

Un élément a_0 de A est dit *potentiellement divergent* ou *faiblement divergent* si, intuitivement, il existe une séquence infinie de réécritures partant de a_0. Formellement, l'ensemble des termes potentiellement divergents, noté $R^{\Uparrow}$, est la plus grande partie de A telle que

$$x \in R^{\Uparrow} \quad \Longleftrightarrow \quad \exists x', (x, x') \in R \wedge x' \in R^{\Uparrow}\ .$$

Si on identifie cette partie de A à une relation de A dans $\mathbb{1}$, on voit que $R^{\Uparrow}$ n'est autre que la relation $\nu X.R\,;X$. Le complémentaire de cette partie de A est l'ensemble des termes *nécessairement normalisants*, aussi appelés *fortement normalisants*, noté $R^{\Downarrow}$. Si on identifie cette partie de A à une relation de $\mathbb{1}$ dans A, on peut calculer la définition de $R^{\Downarrow}$ à partir de la réciproque du complémentaire de $R^{\Uparrow}$ comme suit.

$$R^{\Downarrow} = \overline{R^{\Uparrow\circ}} = \overline{(\nu X.R\,;X)^\circ} = \overline{\nu X.(R\,;X^\circ)^\circ} = \overline{\nu X.X\,;R^\circ} = \mu X.\overline{\overline{X}\,;R^\circ} = \mu X.X \lhd R$$

On a donc $R^{\Downarrow} = \mu X.X \lhd R$, ce qui signifie que l'ensemble des termes fortement normalisants est le plus petit ensemble $R^{\Downarrow}$ tel que

$$x \in R^{\Downarrow} \iff \forall x' \in A, (x, x') \in R \implies x' \in R^{\Downarrow} .$$

Par extension, on dit que la relation R est *fortement normalisante* si $R^{\Downarrow}$ est l'ensemble A tout entier. Si R est fortement normalisante, tout élément de A a au moins une forme normale. On peut comprendre un tel système de réécriture abstrait comme la description d'un algorithme potentiellement non-déterministe mais qui s'arrête toujours.

Théorème 4.3.2. *Tout élément dans $R^{\Downarrow}$ a au moins une forme normale. En particulier, si R est fortement normalisant, alors $\widehat{R}$ est entière.*

Démonstration informelle. On raisonne par l'absurde. Soit x_0 un élément de $R^{\Downarrow}$ qui n'a pas de forme normale. Il doit nécessairement avoir un réduit immédiat par R. Ce réduit immédiat, appelons le x_1, appartient à $R^{\Downarrow}$. Il doit donc à son tour avoir un successeur, sans quoi x_1 serait une forme normale de x_0. On peut itérer l'argument pour montrer que x_0 appartient à l'ensemble des éléments faiblement divergents. Donc x_0 appartient à la fois à $R^{\Downarrow}$ et à son complémentaire $R^{\Uparrow}$, ce qui est absurde. □

Corollaire 4.3.1. *Si R est confluente et fortement normalisante alors $\widehat{R}$ est fonctionnelle.*

La normalisation forte, combinée à la confluence, garantit donc l'existence et l'unicité des formes normales. De plus, la normalisation forte donne accès à un principe de raisonnement puissant dont une application est justement de réduire la confluence à la confluence locale, nettement plus facile à démontrer.

Lemme 4.3.1 (Newman)**.** *Soit R une relation fortement normalisante. Alors R est confluente si et seulement si elle est localement confluente.*

Démonstration. La direction droite vers gauche est évidente. Soit $R : A \nrightarrow A$ une relation fortement normalisante et localement confluente, c'est-à-dire que $R^{\circ} ; R \subseteq R^{*} ; R^{*\circ}$. On doit montrer que pour tout x dans A, et tous y_1, y_2 dans A tels que $xR^{*}y_1$ et $xR^{*}y_2$, il existe z tel que $y_1 R^{*} z$ et $y_2 R^{*} z$. Autrement dit, on doit montrer que

$$\varphi \coloneqq \{x \in A \mid \forall y_1, y_2 \in A, (x, y_1) \in R^{*} \wedge (x, y_2) \in R^{*} \implies (y_1, y_2) \in R^{*} ; R^{*\circ}\}$$

est l'ensemble A tout entier. Puisque la relation R est fortement normalisante, cela revient à montrer que $A = R^{\Downarrow} \subseteq \varphi \subseteq A$. Puisque $R^{\Downarrow}$ est un plus petit point préfixe, il suffit de montrer que $\varphi \lhd R \subseteq \varphi$. Soit $x \in \varphi \lhd R$. On peut comprendre cette hypothèse comme notre hypothèse d'induction, qui exprime qu'on sait clore tous les R^{*}-branchements qui partent d'un réduit immédiat de x par R. Soient y_1 et y_2 tels que $(x, y_1) \in R^{*}$ et $(x, y_2) \in R^{*}$. On doit montrer qu'il existe z tel que $(y_1, z) \in R^{*}$ et $(y_2, z) \in R^{*}$. On raisonne par cas sur la longueur de ces deux réductions.

- Si $x = y_i$ pour $i = 1$ ou $i = 2$, on conclut immédiatement.

- Sinon, il existe y'_1 et y'_2 des réduits immédiats de x tels que $(y'_i, y_i) \in R^*$. Par confluence locale de R, il existe z' qui clôt le branchement (y'_1, y'_2) partant de x. On applique l'hypothèse d'induction une première fois pour obtenir z' qui clôt le branchement (y_1, z') partant de y'_1. On applique l'hypothèse d'induction une seconde fois pour obtenir z qui clôt le branchement (z', y_2) partant de y'_2. La jonction obtenue permet de construire un chemin menant de y_1 à z' puis z, ainsi que de y_2 à z. □

Remarque 4.3.1. Le lemme de Newman est le premier à avoir été démontré dans le cadre des systèmes de réécriture abstraits, dans les années 1940.

4.3.1. Outils de preuve

Comment prouver qu'un système est fortement normalisant ? Une possibilité classique est d'utiliser une *mesure*, au sens suivant.

Définition 4.3.1. Soit $R : A \nrightarrow A$ et $S : B \nrightarrow B$ deux systèmes de réécriture abstraits. Un *morphisme de systèmes de réécritures abstraits* de R dans S est une fonction f de A dans B telle que $f^\circ ; R ; f \subseteq S$.

La condition de la définition précédente exprime que pour tout couple (x, x') qui appartient à R le couple $(f(x), f(x'))$ appartient à S.

Lemme 4.3.2 (Mesure)**.** *Un système de réécriture abstrait $R : A \nrightarrow A$ est fortement normalisant si et seulement s'il existe un système de réécriture abstrait $S : B \nrightarrow B$ et un morphisme f de R dans S tels que S soit fortement normalisant.*

Démonstration informelle. Si R est fortement normalisant, il suffit de prendre $S \coloneqq R$ et $f = id$. Si $(x_0, x_1, x_2, \dots)$ est une réduction infinie par R, alors $(f(x_0), f(x_1), f(x_2), \dots)$ est une réduction infinie par S, ce qui est contradictoire avec le fait que S soit fortement normalisant. □

Le même lemme peut être formulé en terme de relation d'ordre dites *bien fondée*, c'est-à-dire de relations d'ordres telles que la réciproque de l'ordre strict associée est fortement normalisante. Un exemple canonique est l'ordre strict $<_\omega$ sur les entiers naturels.

4.3.2. Formulation relationnelle de l'induction bien fondée

On peut reformuler le principe d'induction utilisé dans la preuve du lemme de Newman. Dans ce qui suit, on utilise l'*implication relationnelle*, notée $S \supset T$, définie comme la relation $\overline{S} \cup T$. Il s'agit d'une opération décroissante en S et croissante en T. Elle est caractérisée par le fait que $R \cap S \subseteq T$ si et seulement si $R \subseteq S \supset T$.

Théorème 4.3.3 (Induction bien fondée)**.** *Soient R, S et T des endorelations sur un ensemble A avec R fortement normalisante. Les règles de déduction suivantes sont valides.*

$$\frac{(S \supset T) \lhd R \cap S \subseteq T}{S \subseteq T} \qquad \frac{R^\circ \rhd (S \supset T) \cap S \subseteq T}{S \subseteq T}$$

Démonstration. On remarque tout d'abord que $S \subseteq T$ est équivalente à l'inclusion

$$\mathbf{1}_{A,\mathbb{1}} ; \mathbf{1}_{\mathbb{1},A} \subseteq S \supset T.$$

On a

$$\begin{aligned}
\mathbf{1}_{A,\mathbb{1}} ; \mathbf{1}_{\mathbb{1},A} \subseteq S \supset T &\iff \mathbf{1}_{\mathbb{1},A} \subseteq \mathbf{1}_{A,\mathbb{1}} \rhd (S \supset T) \\
&\iff R^{\Downarrow} \subseteq \mathbf{1}_{A,\mathbb{1}} \rhd (S \supset T) \\
&\impliedby \mathbf{1}_{A,\mathbb{1}} \rhd (S \supset T) \lhd R \subseteq \mathbf{1}_{A,\mathbb{1}} \rhd S \supset T \\
&\impliedby (S \supset T) \lhd R \subseteq S \supset T \\
&\iff (S \supset T) \lhd R \cap S \subseteq T.
\end{aligned}$$

On obtient la deuxième règle à partir de la première comme suit.

$$\begin{aligned}
S \subseteq T \iff S^\circ \subseteq T^\circ &\impliedby (S^\circ \supset T^\circ) \lhd R \cap S^\circ \subseteq T^\circ \\
&\iff ((S^\circ \supset T^\circ) \lhd R \cap S^\circ)^\circ \subseteq T \\
&\iff R^\circ \rhd (S \supset T) \cap S \subseteq T.
\end{aligned}$$

□

4.4. Stratégies

Une système de réécriture peut être vu comme un algorithme non-déterministe, mais aussi comme une spécification qui décrit des réécritures possibles. Ce point de vue mène naturellement à s'intéresser aux implémentations de cette spécification, ce qu'on identifie aux restrictions du système de réécriture.

Définition 4.4.1. Soient R et S deux systèmes de réécriture abstraits sur le même ensemble A. On dit que S est une *stratégie* pour R si S est incluse dans R et que l'image inverse de A par S est égale à celle de A par R.

Donc S est une stratégie pour R dès lors que $S \subseteq R$ et que $R ; \mathbf{1} \subseteq S ; \mathbf{1}$. Ces deux inclusions entraînent automatiquement l'égalité $R ; \mathbf{1} = S ; \mathbf{1}$. Autrement dit, pour tout x dans A, il existe x_1 tel que $(x, x_2) \in R$ si et seulement s'il existe x_2 tel que $(x, x_2) \in S$.

Si S est une stratégie pour R, alors toute forme normale pour S est une forme normale pour R, et $\widehat{S}$ est incluse dans $\widehat{R}$. On dit que S est *complète* lorsque $\widehat{S}$ est égale à $\widehat{R}$. Une stratégie complète pour un système de réécriture s'approche d'un algorithme capable de calculer une forme normale de n'importe quel élément. C'est encore plus exact lorsque la stratégie est déterministe.

Propriété 4.4.1. *Soient R et S deux systèmes de réécritures abstraits sur le même ensemble A tels que S est inclus dans R. Si R est fortement normalisant alors S est fortement normalisant.*

Théorème 4.4.1. *Si R est un système de réécriture abstrait confluent et fortement normalisant, alors toute stratégie pour R est complète.*

Démonstration. On sait déjà que $S^{\Downarrow}$ est incluse dans $R^{\Downarrow}$, on démontre l'inclusion dans le sens contraire, à savoir que toute forme normale pour R est une forme normale pour S.

Soit x' une forme normale de x par R. On sait par la propriété 5.1.6 que S est fortement normalisante, donc x a au moins une forme normale y pour S. Mais y est aussi une forme normale de x pour R. Puisque R est confluente, les formes normales pour R sont uniques, donc $x = y$ et x est une forme normale pour S. □

Dans le cas idéal d'un système de réécriture abstrait confluent et fortement normalisant, toutes les stratégies se valent donc du point de vue de la normalisation. On peut donc implémenter un tel système en choisissant arbitrairement une stratégie déterministe, qu'on appliquera autant que possible pour calculer une forme normale. Par normalisation forte, on sait que cet algorithme termine toujours.

4.4.1. Exercices

* **Ex. 16** — Donner un exemple de relation de réécriture non vide sur un ensemble fini qui n'a pas de formes normales.

* **Ex. 17** — Définir des systèmes de réécriture $R_1, \ldots, R_9$ tels que :

- R_1 est fortement normalisant mais pas fini,
- R_2 est fini mais pas normalisant,
- R_3 est fortement normalisant et confluent,
- R_4 est fortement normalisant mais pas confluent,
- R_5 n'est ni confluent ni normalisant,
- R_6 est confluent mais pas normalisant,
- R_7 est confluent et normalisant mais pas fortement normalisant,
- R_8 est normalisant mais ni confluent ni fortement normalisant,
- R_9 est localement confluent mais pas confluent.

* **Ex. 18** — Démontrer que pour toute relation $R : A \rightarrow\!\!\!\!\!\rightarrow A$, on a $R^{*\circ} = R^{\circ *}$.

* **Ex. 19** — Soit une relation $R : A \rightarrow\!\!\!\!\!\rightarrow A$. Démontrer que les trois relations R_1, R_2 et R_3 définies ci-dessous sont égales à la fermeture réflexive-transitive de R.

$$R_1 \coloneqq \mu S.Id \cup R \cup S\,;S \qquad R_2 \coloneqq \mu S.Id \cup R \cup S\,;S \qquad R_3 \coloneqq \mu S.Id \cup S\,;R$$

* **Ex. 20** — Démontrer que pour toute relation $R : A \rightarrow\!\!\!\!\!\rightarrow A$, on a $R^*\,;R^{*\circ} \subseteq R^{=}$.

* **Ex. 21** — Soit $A = \{a, b, c, d\}$ et R la relation $\{(a, b), (a, c), (b, d), (c, d)\}$ sur A. Lister toutes les stratégies sur R.

* **Ex. 22** — Soit $R : A \rightarrow\!\!\!\!\!\rightarrow A$ un système de réécriture abstrait. Démontrer par manipulation formelle des opérations relationnelles les deux énoncés suivants.

1. Tous les éléments fortement normalisants pour la clôture transitive de R sont fortement normalisants pour R. Autrement dit, $R^{+\Downarrow} \subseteq R^{\Downarrow}$.
2. Tous les éléments fortement normalisants pour R sont fortement normalisants pour la clôture transitive de R. Autrement dit, $R^{\Downarrow} \subseteq R^{+\Downarrow}$.

* **Ex. 23** — Démontrer par manipulation formelle des opérations relationnelles que si S est une stratégie pour R, alors R et S ont le même ensemble de formes normales.

Deuxième partie

Algèbre universelle

5. Théories algébriques du premier ordre

Ce chapitre commence par une introduction aux idées générales par le biais d'un exemple, avant d'aborder dans les détails techniques.

5.1. Le cas particulier des monoïdes

On appelle *structure algébrique* un ensemble muni d'opérations connectées par des équations universellement quantifiées. On va voir que toutes les structures algébriques classiques se prêtent à un format de description uniforme, et que cette question est connectée à l'informatique. Pour ce faire, on va étudier des exemples de structures algébriques parmi les plus simples qui soient.

Définition 5.1.1. Un *magma pointé* M est un triplet $(|M|, \cdot_M, \varepsilon_M)$ où $|M|$ est un ensemble appelé *support* de M, $(\cdot_M) : |M| \times |M| \to |M|$ est une fonction binaire sur $|M|$ appelée *loi* du magma pointé M, et ε_M est un élément de $|M|$.

Comme à notre habitude, on omet les indices lorsque la structure algébrique à laquelle appartient les opérations peut être déduite du contexte. Il en sera de même pour toutes les structures algébriques considérées.

Remarque 5.1.1. L'élement ε_M de $|M|$ peut être vu comme une opération nullaire, c'est-à-dire d'arité (nombre d'arguments) zéro.

Définition 5.1.2. Un *monoïde* M est un magma pointé qui vérifie les équations

$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \tag{5.1}$$

$$\varepsilon \cdot x = x \tag{5.2}$$

$$x \cdot \varepsilon = x \tag{5.3}$$

pour tous $x, y, z \in |M|$.

De façon général, un *homomorphisme* entre structures algébriques est une fonction qui préserve la structure en question. Par exemple, si M et N sont des magmas pointés, un *homomorphisme de magmas pointés* est une fonction f de domaine $|M|$ et codomaine $|N|$ telle que $f(x \cdot y) = f(x) \cdot f(y)$ et $f(\varepsilon) = \varepsilon$. Tout monoïde est un magma pointé, et un *homomorphisme de monoïdes* est un homomorphisme entre les magmas pointés sous-jacents à deux monoïdes.

La question qui va nous occuper est la suivante : étant donnée une classe de structures algébriques, par exemple celle des magma pointés ou des monoïdes, comment transformer n'importe quel ensemble en une instance d'une telle structure, et ce de façon générique ? Dans le cas des magma pointés, la réponse devrait être familière aux informaticiennes et informaticiens : il s'agit de la notion de *syntaxe*.

5.1.1. Une syntaxe pour les magma pointés

5.1.1.1. Le magma pointé libre

L'objectif de cette section est de décrire et manipuler le *magma pointé libre* sur un ensemble Γ quelconque. Intuitivement, le magma pointé libre est la solution minimale au problème qui consiste à équiper Γ d'une structure de magma pointé. Ici « minimale » ne doit pas être compris au sens de la cardinalité de l'ensemble sous-jacent, mais au sens de la propriété universelle suivante.

Définition 5.1.3. Un *magma pointé libre* sur un ensemble Γ est un couple (Γ', r), où Γ' est un magma pointé et r une fonction de Γ dans Γ', tel que pour tout magma pointé M et toute fonction γ de Γ dans M, il existe un unique homomorphisme de magma pointé $\gamma^\sharp$ de Γ' dans M tel que $\gamma = r\,;\gamma^\sharp$.

Il existe une construction canonique du magma pointé libre sur un ensemble Γ. On définit l'ensemble $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ des arbres finis dont les feuilles sont étiquettées par des éléments de Γ et les nœuds sont étiquettés soit par $\cdot$, soit par ε. Les nœuds ont autant de sous-termes que l'opérateur correspondant a d'arguments. Plus formellement, l'ensemble $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ est le plus petit ensemble fermé par les règles suivantes.

$$\frac{x \in \Gamma}{\mathtt{var}(x) \in \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)} \qquad \frac{t \in \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma) \qquad u \in \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)}{\mathtt{op}[\cdot](t,u) \in \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)} \qquad \frac{}{\mathtt{op}[\varepsilon]() \in \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)}$$

Il est clair que cet ensemble est doté d'une structure de monoïde dont la loi est donnée par l'opération $\mathtt{op}[\cdot]$ et dont l'élément neutre est $\mathtt{op}[\varepsilon]()$. Qui plus est, le fait qu'il s'agisse du plus petit ensemble fermé par ces règles induit la propriété universelle suivante.

Théorème 5.1.1. *Le couple* $(\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma), \mathtt{var}(-))$ *est un magma pointé libre sur* Γ.

Remarque 5.1.2. On ne se soucie pas de démontrer l'existence de l'ensemble $\mathcal{T}[\Sigma_{\mathsf{MaP}}]$ pour l'instant, ni la validité du théorème 5.1.1. On justifiera la notation $\mathcal{T}[\Sigma_{\mathsf{MaP}}]$ ultérieurement.

Dans le cas de la structure de magma pointé libre donnée par la construction $\mathcal{T}[\Sigma_{\mathsf{MaP}}]$, on notera $\mathsf{fold}(\gamma)$ plutôt que $f^\sharp$ l'homomorphisme de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ dans M. La propriété universelle implique que ce morphisme vérifie les équations suivantes.

$$\mathsf{fold}(\gamma)(\mathtt{var}(x)) = \gamma(x) \tag{5.4}$$
$$\mathsf{fold}(\gamma)(\mathtt{op}[\cdot](t,u)) = \mathsf{fold}(\gamma)(t) \cdot_M \mathsf{fold}(\gamma)(u) \tag{5.5}$$
$$\mathsf{fold}(\gamma)(\mathtt{op}[\varepsilon]()) = \varepsilon_M \tag{5.6}$$

La première est donnée par l'équation de la propriété, les deux suivantes expriment le fait que $\mathsf{fold}(\gamma)$ est un homomorphisme. Ces équations ont un contenu calculatoire : on peut les lire comme la définition d'une fonction récursive. Cela suggère d'adapter la structure précédente sous forme de code OCaml comme à la figure 5.1. Une valeur de type `t pointed_magma_structure` équipe le type `t` d'une structure de magma pointé en

```
type 's pointed_magma_structure = { mul : 's -> 's -> 's; eps : 's; }

type 'gamma pointed_magma_term =
| Var of 'gamma
| Mul of 'gamma pointed_magma_term * 'gamma pointed_magma_term
| Eps

let free_pointed_magma_structure =
  { mul = fun t u -> Mul (t, u); eps = Eps; }

let rec fold m gamma = function
| Var x -> gamma x
| Mul (t, u) -> m.mul (fold m gamma t) (fold m gamma u)
| Eps -> m.eps
```

FIG. 5.1. : une représentation des magmas pointés en OCaml

spécifiant la loi et l'élément neutre. Le type du magma pointé libre est défini par un type algébrique, qu'on équipe de la structure de magma pointé évidente et déjà citée. La fonction récursive `fold` est parfois appelé *récurseur primitif* de ce type, et permet d'envoyer une valeur du type dans n'importe quel magma pointé comme exprimé par le théorème 5.1.1.

Remarque 5.1.3. D'un point de vue informatique, le récurseur primitif traverse un terme, c'est-à-dire un arbre de syntaxe, pour lui associer un élément de son codomaine obtenue en appliquant des opérations. De façon imagée, on peut le décrire comme un *interprète générique* pour le langage des magmas pointés. Ou encore, pour adopter un vocabulaire théoricien, on peut le décrire comme une *sémantique*, c'est-à-dire une interprétation de la syntaxe dans un *modèle* du langage.

5.1.1.2. Induction

Le code OCaml a le grand avantage d'être exécutable, contrairement aux mathématiques usuelles. En revanche, la formulation mathématique s'en distingue en stipulant l'unicité de l'homomorphisme $\mathsf{fold}(\gamma)$, dont on peut déduire un principe de raisonnement crucial, le *raisonnement par induction*.

Propriété 5.1.1. *Soit P une partie de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ qui satisfait les conditions suivantes.*

1. *Pour tout x dans Γ, le terme $\mathtt{var}(x)$ appartient à P.*
2. *Pour tous t et u dans P, le terme $\mathtt{op}[\cdot](t, u)$ appartient à P.*
3. *Le terme $\mathtt{op}[\varepsilon]()$ appartient à P.*

Alors P est l'ensemble $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ tout entier.

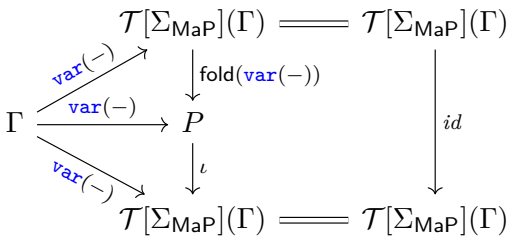


FIG. 5.2. : construction d'un principe d'induction

Démonstration. Soit P une partie de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ qui satisfait les conditions de l'énoncé de la propriété. Les conditions 2 et 3 permettent d'équiper P d'une structure de magma pointé héritée de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$. On écrit ι pour l'injection de P dans $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ induite par l'inclusion. On voit immédiatement qu'il s'agit d'un homomorphisme de monoïde.

La condition 1 implique que la fonction $\mathtt{var}(-)$ qui envoie x dans $\mathtt{var}(x)$ se restreint à une fonction de Γ dans P. Le théorème 5.1.1 implique donc l'existence d'un homomorphisme $\mathsf{fold}(\mathtt{var}(-))$ de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ dans P. Donc $\mathsf{fold}(\mathtt{var}(-)) ; \iota$ est un homomorphisme de $\mathcal{T}[\Gamma_{\mathsf{MaP}}]$ dans lui-même tel que $\mathtt{var}(-) ; (\mathsf{fold}(\mathtt{var}(-)) ; \iota) = \mathtt{var}(-)$. Or, la fonction identité est un autre homomorphisme tel que $\mathtt{var}(-) ; id = \mathtt{var}(-)$. On a donc nécessairement $\mathsf{fold}(\mathtt{var}(-)) ; \iota = id$ par la propriété universelle ; cette situation est résumée à la figure 5.2. Tout ceci implique que tout élément $t \in \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ est égal à son image $\mathsf{fold}(\mathtt{var}(-))(t)$ dans P et donc, en particulier, appartient à P. Autrement dit, $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ est une partie de P. □

Remarque 5.1.4. Il est aussi possible de déduire le théorème 5.1.1 de la propriété 5.1.1, mais la construction est plus difficile. Il s'agit de l'approche qui a été suivie historiquement en mathématiques, puisqu'elle est plus naturelle pour qui adopte la théorie des ensembles comme langage primitif.

5.1.1.3. Renommage

Soit ρ une fonction d'un ensemble Γ dans un ensemble Δ. Si l'on pense à Γ et à Δ comme à des ensembles de noms de variables, alors on doit penser à ρ comme à un *renommage* qui associe à chaque variable x dans Γ son nouveau nom $\rho(x)$ dans Δ. Il doit être possible d'appliquer un tel renommage aux variables d'un élément de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ pour obtenir un élément de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Delta)$. On peut définir cette opération en applicant la propriété universelle une nouvelle fois, en exploitant le fait que $\mathcal{T}[\Sigma]_{\mathsf{MaP}}(\Delta)$ est muni d'une structure de monoïde.

Définition 5.1.4. Si $\rho : \Gamma \to \Delta$ est une fonction, on appelle *application du renommage* ρ l'homomorphisme défini par $\mathsf{fold}(\rho ; \mathtt{var}(-)) : \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma) \to \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Delta)$.

Cet homomorphisme est noté $(-)[\rho] : \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma) \to \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Delta)$. Il est caractérisé

par les équations suivantes.

$$\begin{aligned}\mathtt{var}(x)[\rho] &= \mathtt{var}(\rho(x))\\ \mathtt{op}[\cdot](t,u)[\rho] &= \mathtt{op}[\cdot](t[\rho],u[\rho])\\ \mathtt{op}[\varepsilon]()[\rho] &= \mathtt{op}[\varepsilon]()\end{aligned}$$

De plus, l'application d'un renommage vérifie les propriétés de fonctorialité intuitives.

$$t[id] = t \tag{5.7}$$
$$t[\rho\,;\varphi] = t[\rho][\varphi] \tag{5.8}$$

Remarque 5.1.5. On utilise parfois le terme « renommage » pour désigner à la fois ρ et l'homomorphisme $(-)[\rho]$. Une telle terminologie est ambiguë et devrait être évitée.

5.1.1.4. Substitution

Enfin, on peut définir une dernière opération essentielle, la *substitution*. Celle-ci généralise le renommage pour autoriser le remplacement d'une variable par un nouvel arbre de syntaxe. Une substitution est une fonction $\sigma : \Gamma \to \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Delta)$. Il faut y penser comme à une façon d'associer à chaque nom de variable x dans Γ un terme $\sigma(x)$ dans $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Delta)$. Étant donné un terme t de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$, l'application de la substitution σ à t doit mener à un nouvel arbre dans $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Delta)$ où tout sous-arbre de la forme $\mathtt{var}(x)$ a été remplacé par $\sigma(x)$.

Définition 5.1.5. Si $\sigma : \Gamma \to \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Delta)$ est une fonction, on appelle *application de la substitution* σ l'homomorphisme de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ dans $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Delta)$ défini comme $\mathsf{fold}(\sigma)$.

De façon abusive, la tradition est de voir le renommage comme un cas particulier de substitution, et donc de noter $(-)[\sigma]$ pour l'application de la substitution σ. Pourtant, il ne s'agit pas stricto-sensu de la même opération que de l'application de σ vu comme un simple renommage ! Toutefois, les hypothèses de typage assurent qu'il ne peut pas y avoir confusion entre les deux opérations, leurs codomaines étant distincts.

On voudrait pouvoir écrire les équations de fonctorialité pour l'application de substitution analogues aux résultats obtenus pour l'application d'un renommage. Toutefois, les équations de fonctorialité qui suivent sont mal typées.

$$t[id] = t \qquad \text{(mal typé)}$$
$$t[\sigma\,;\sigma'] = t[\sigma][\sigma'] \qquad \text{(mal typé)}$$

5.1.2. Les monoïdes

On souhaite maintenant jouer le même jeu que précédemment avec les monoïdes, et obtenir une description du monoïde libre doté de la propriété universelle attendue.

Définition 5.1.6. Un *monoïde libre* sur un ensemble Γ est un couple (Γ', r), où Γ' est un monoïde et r une fonction de Γ dans Γ', tel que pour tout monoïde M et toute fonction γ de Γ dans M, il existe un unique homomorphisme de monoïde $\gamma^\sharp$ de Γ' dans M tel que $\gamma = r\,;\gamma^\sharp$.

Un monoïde étant un cas particulier de magma pointé, il est naturel de chercher à obtenir un monoïde libre sur un ensemble Γ à partir du magma pointé libre $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$. Cela mène naturellement à se poser la question du monoïde libre *sur un magma pointé* plutôt que sur un simple ensemble.

5.1.2.1. Monoïde libre sur un magma pointé

Définition 5.1.7. Un *monoïde libre* sur un magma pointé M est un couple (M', r), où M' est un monoïde et r un homomorphisme de magma pointés de M dans M', tel que pour tout monoïde N et tout homomorphisme de magmas pointés f de M dans N, il existe un unique homomorphisme de monoïde $f^\sharp$ de M' dans N tel que $f = r\,;f^\sharp$.

Un monoïde étant un magma pointé auquel on a imposé les équations (5.1) à (5.3), il est naturel de cherche à construire le monoïde libre sur un magma pointé donné par un certain quotient. Toutefois, on doit prendre garde à ce que le quotient soit encore un magma pointé, ce qui mène aux définitions suivantes.

Soit M un magma pointé et $R : M \nrightarrow M$ une endorelation de M.

Définition 5.1.8. L'*extension compatible* de R, notée $\Sigma_{\mathsf{MaP}}(R)$, est la relation définie par

$$\Sigma_{\mathsf{MaP}}(R) : M \nrightarrow M \quad \coloneqq \quad \{(\varepsilon, \varepsilon)\} \cup \{(x \cdot y, x' \cdot y') \mid (x, x') \in R, (y, y') \in R\}.$$

La *clôture compatible* de R, notée $\Sigma^\bullet_{\mathsf{MaP}}(R)$, est la relation définie par

$$\Sigma^\bullet_{\mathsf{MaP}}(R) : M \nrightarrow M \quad \coloneqq \quad \mu X.R \cup \Sigma_{\mathsf{MaP}}(X).$$

Remarque 5.1.6. Le terme « compatible » signifie ici « compatible avec la structure de magma pointé ». D'autres structures algébriques donnent lieu à d'autres notions de compatibilité. Le sens exact de ce terme dépend donc de son contexte d'utilisation.

Définition 5.1.9. Une relation est *compatible* lorsqu'elle contient son extension compatible. C'est une *congruence* lorsqu'elle est compatible et une relation d'équivalence.

Remarquons qu'une relation d'équivalence contient toujours $(\varepsilon, \varepsilon)$ par réflexivité. Vérifier qu'une relation d'équivalence est une congruence se réduit donc à vérifier sa compatibilité avec la multiplication du monoïde.

La clôture symétrique-réflexive-transitive et la clôture compatible ne sont pas des opérations indépendantes, au sens où elles ne commutent pas. Pour l'illustrer, considérons le magma pointé libre sur trois générateurs $\{a, b, c\}$, c'est-à-dire $M \coloneqq \mathcal{T}[\Sigma_{\mathsf{MaP}}](\{a, b, c\})$. Dans ce qui suit, on omet les injections $\mathtt{var}(-)$ pour alléger les notations.

- Considérons la relation compatible S_1 sur M définie par

  $$S_1 \quad \coloneqq \quad \Sigma^\bullet_{\mathsf{MaP}}(\{(a, b)\}).$$

  On remarque que si la paire (t, u) appartient à S_1, ni t ni u ne peuvent contenir de feuille de la forme c. Les paires (c, c) et (a, b) appartiennent à $R^=$, mais pas $(c{\cdot}a, c{\cdot}b)$. Donc $S_1^=$ n'est pas compatible.

- Considérons la relation d'équivalence S_2 sur M définie par

$$S_2 \quad \coloneqq \quad \{(a,b),(b\cdot b,c)\}^{=}.$$

  Les paires $(a\cdot a, b\cdot b)$ et $(b\cdot b, c)$ appartiennent à la clôture compatible de S_2, mais pas la paire $(a\cdot a, c)$. Donc $\Sigma^{\bullet}_{\mathsf{MaP}}(S_2)$ n'est pas transitive, et a fortiori n'est pas une relation d'équivalence.

Ces deux exemples illustrent que la clôture symétrique-réflexive-transitive d'une relation compatible n'est pas nécessairement compatible, et que la clôture compatible d'une relation d'équivalence n'est pas nécessairement une relation d'équivalence. On a donc besoin d'une construction ad-hoc.

Définition 5.1.10. La *clôture congruentielle* de R, notée $\Sigma^{=}_{\mathsf{MaP}}(R)$, est la relation définie par

$$\Sigma^{=}_{\mathsf{MaP}}(R) : M \nrightarrow M \quad \coloneqq \quad \mu X.R \cup X^{=} \cup \Sigma_{\mathsf{MaP}}(X).$$

Soit M un magma pointé et R une relation d'équivalence sur M. On souhaite équiper M/R d'une structure de magma pointé héritée de celle de M, c'est-à-dire définir une fonction f de $M/R \times M/R$ dans M/R telle que, pour tous x et y dans M, on ait $[x\cdot y]_R = f([x]_R,[y]_R)$. Il suffit d'utiliser la propriété universelle de l'ensemble quotient (théorème 2.5.1) avec la fonction $g(x,y) = [x\cdot y]_R$. Il est facile de vérifier que g respecte R si et seulement si R est compatible, par définition !

On peut donc transformer tout magma pointé M en monoïde. Pour cela, on définit l'endorelation $\equiv_{\mathsf{Mon}}$: $M \nrightarrow M$ comme la clôture congruentielle de la relation $R : M \nrightarrow M$ définie par

$$R_{\mathsf{Mon}} \coloneqq \{((t\cdot u)\cdot v, t\cdot(u\cdot v)) \mid t,u,v \in M\} \cup \{(\varepsilon\cdot t, t) \mid t \in M\} \cup \{(t\cdot\varepsilon, t) \mid t \in M\}.$$

On a vu que le quotient $M/{\equiv_{\mathsf{Mon}}}$ hérite de la structure de magma pointé de M. Qui plus est, c'est un monoïde en vertu du quotient imposé. De plus, il s'agit d'un monoïde libre sur M.

Théorème 5.1.2. *Le couple $(M/{\equiv_{\mathsf{Mon}}}, [-]_{\equiv_{\mathsf{Mon}}})$ est un monoïde libre sur le magma pointé M.*

Pour le démontrer, on va avoir besoin des résultats intermédiaires suivants.

Propriété 5.1.2. *Soient M et M' deux magmas pointés et f un homomorphisme de M dans M'. Soit R une endorelation de M. Si f respecte R alors f respecte $\Sigma_{\mathsf{MaP}}(R)$.*

Corollaire 5.1.1. *Soient M et M' deux magmas pointés et f un homomorphisme de M dans M'. Soit R une endorelation de M. Si f respecte R alors f respecte la clôture congruentielle de R.*

Démonstration. Supposons $f^\circ ; R ; f \subseteq Id$. On doit montrer $f^\circ ; \Sigma^{\bullet}_{\mathsf{MaP}}(R) ; f \subseteq Id$. On a

$$\begin{aligned} f^\circ ; \Sigma^{\bullet}_{\mathsf{MaP}}(R) ; f \subseteq Id &\iff \Sigma^{\bullet}_{\mathsf{MaP}}(R) \subseteq f^\circ \rhd Id \lhd f \\ &\impliedby R \cup (f^\circ \rhd Id \lhd f)^{=} \cup \Sigma_{\mathsf{MaP}}(f^\circ \rhd Id \lhd f) \subseteq f^\circ \rhd Id \lhd f. \end{aligned}$$

On doit donc démontrer les inclusions suivantes.

$$f^\circ ; R ; f \subseteq Id \tag{5.9}$$
$$f^\circ ; (f^\circ \rhd Id \lhd f)^= ; f \subseteq Id \tag{5.10}$$
$$f^\circ ; \Sigma_{\mathsf{MaP}}(f^\circ \rhd Id \lhd f) ; f \subseteq Id \tag{5.11}$$

La première est vérifiée par hypothèse. La seconde est conséquence de la propriété 2.5.3 et la troisième de la propriété 5.1.2. □

Propriété 5.1.3. *Si M est un magma pointé, M' un monoïde, et $f : M \to M'$ un homomorphisme de magma pointé, alors f respecte la relation R_{Mon}.*

Démonstration. On montre que f respecte la relation R_{Mon} en raisonnant par cas.

- Considérons $t, u, v \in M$. On a

$$\begin{aligned} f((t \cdot u) \cdot v) &= (f(t) \cdot f(u)) \cdot f(v) && (f \text{ hom. magma pointé}) \\ &= f(t) \cdot (f(u) \cdot f(v)) && (M' \text{ monoïde}) \\ &= f(t \cdot (u \cdot v)). && (f \text{ hom. magma pointé}) \end{aligned}$$

- Considérons $t \in M$. On a

$$\begin{aligned} f(\varepsilon \cdot t) &= \varepsilon \cdot f(t) && (f \text{ hom. magma pointé}) \\ &= f(t). && (M' \text{ monoïde}) \end{aligned}$$

- Le cas de la neutralité de ε à droite est similaire au précédent. □

On peut maintenant démontrer le théorème 5.1.2.

Démonstration. Soit N un monoïde et f un homomorphisme de magma pointé de M dans N. Par la propriété 5.1.3, f respecte R_{Mon}. Par le corollaire 5.1.1, f respecte $\equiv_{\mathsf{Mon}}$. Donc, par la propriété universelle de l'ensemble quotient, il existe une unique fonction $f^\sharp$ de $M/\equiv_{\mathsf{Mon}}$ dans N telle que $[-]_{\equiv_{\mathsf{Mon}}} ; f^\sharp$. Il reste à montrer que cette fonction est un homomorphisme de monoïde. On déduit facilement les égalités

$$f^\sharp([x]_{\equiv_{\mathsf{Mon}}} \cdot [y]_{\equiv_{\mathsf{Mon}}}) = f^\sharp([x \cdot y]_{\equiv_{\mathsf{Mon}}}) = f(x \cdot y) = f(x) \cdot f(y) = f^\sharp([x]_{\equiv_{\mathsf{Mon}}}) \cdot f^\sharp([y]_{\equiv_{\mathsf{Mon}}})$$
$$f^\sharp(\varepsilon) = f^\sharp([\varepsilon]_{\equiv_{\mathsf{Mon}}}) = f(\varepsilon) = \varepsilon$$

des propriétés énoncées précédemment. □

5.1.2.2. Monoïde libre sur un ensemble

On voit donc comment bâtir le monoïde libre sur un ensemble Γ. Il suffit de quotienter l'ensemble $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ par la congruence $\equiv_{\mathsf{Mon}}$.

Propriété 5.1.4. *Soit M un magma pointé, N un monoïde, et f un homomorphisme de magmas pointés de M dans N. Alors f respecte $\equiv_{\mathsf{Mon}}$.*

Théorème 5.1.3. *Pour tout ensemble* Γ*, le couple* $(\mathcal{F}[\mathsf{Mon}](\Gamma), \iota)$*, où*

$$\mathcal{F}[\mathsf{Mon}](\Gamma) \coloneqq \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)/\equiv_{\mathsf{Mon}}$$

et ι *est la composée* $\mathtt{var}(-)\,;[-]_{\equiv_{\mathsf{Mon}}}$*, est un monoïde libre sur* Γ*.*

Démonstration. Soit M un monoïde et $\gamma : \Gamma \to M$ une fonction. Puisque $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ est un magma pointé libre sur Γ, il existe un unique homomorphisme de magma pointé $\mathsf{fold}(\gamma) : \mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma) \to M$ tel que $\gamma = \mathtt{var}(-)\,;\mathsf{fold}(\gamma)$. Puisque $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)/\equiv_{\mathsf{Mon}}$ est un monoïde libre sur $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$, il existe un unique homomorphisme $\mathsf{fold}(\gamma)^\sharp$ tel que $\mathsf{fold}(\gamma) = [-]_{\equiv_{\mathsf{Mon}}}\,;\mathsf{fold}(\gamma)^\sharp$. On a donc en particulier $\gamma = \mathtt{var}(-)\,;[-]_{\equiv_{\mathsf{Mon}}}\,;\mathsf{fold}(\gamma)^\sharp$. □

5.1.3. Contextes et compatibilité

Dans ce qui suit, si $R : A_1 \nrightarrow A_2$ et $S : B_1 \nrightarrow B_2$ sont des relations, on note $R \times S$ pour la relation de $A_1 \times B_1$ dans $A_2 \times B_2$ qui contient exactement les paires de paires $((x_1, y_1), (x_2, y_2))$ telles que (x_1, x_2) est dans R et (y_1, y_2) est dans S.

Définition 5.1.11. Soit M un magma pointé et R une endorelation de M. L'*extension contextuelle* de R, notée $\Sigma^{\square}_{\mathsf{MaP}}(R)$, est l'endorelation de M définie par

$$\begin{aligned}\Sigma^{\square}_{\mathsf{MaP}}(R) &\coloneqq \cdot^\circ\,;((R \times Id) \cup (Id \times R))\,;\cdot \\ &= \{(x \cdot y, x' \cdot y) \mid (x, x') \in R, y \in M\} \cup \{(x \cdot y, x \cdot y') \mid x \in M, (y, y') \in R\}.\end{aligned}$$

Une relation est *contextuelle* lorsqu'elle contient son extension contextuelle. La *clôture contextuelle* de R, notée $\mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R)$, est l'endorelation de M définie par

$$\mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R) \coloneqq \mu X.R \cup \Sigma^{\square}_{\mathsf{MaP}}(X).$$

Propriété 5.1.5. *On a* $\Sigma_{\mathsf{MaP}}(R) \subseteq \Sigma^{\square}_{\mathsf{MaP}}(R)^{\leq 2}$*.*

Démonstration. On a

$$\begin{aligned}\cdot^\circ\,;R \times R\,;\cdot &\subseteq \cdot^\circ\,;(R^2 \times R^0 \cup R^1 \times R^1 \cup R^0 \times R^2)\,;\cdot \\ &= \cdot^\circ\,;((R \times Id) \cup (Id \times R))\,;((R \times Id) \cup (Id \times R))\,;\cdot \\ &\subseteq \cdot^\circ\,;((R \times Id) \cup (Id \times R))\,;\cdot\,;\cdot^\circ\,;((R \times Id) \cup (Id \times R))\,;\cdot \\ &\subseteq \Sigma^{\square}_{\mathsf{MaP}}(R)^2\end{aligned}$$

et on en déduit $\Sigma_{\mathsf{MaP}}(R) = \cdot^\circ\,;R \times R\,;\cdot \cup \{(\varepsilon, \varepsilon)\} \subseteq \Sigma^{\square}_{\mathsf{MaP}}(R)^2 \cup Id \subseteq \Sigma^{\square}_{\mathsf{MaP}}(R)^{\leq 2}$. □

Lemme 5.1.1. *La relation* $\Sigma_{\mathsf{MaP}}(\mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R))$ *est incluse dans* $\mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R)^{=}$*.*

Démonstration. On a

$$\begin{aligned}\Sigma_{\mathsf{MaP}}(\mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R)) &\subseteq \Sigma^{\square}_{\mathsf{MaP}}(\mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R))^{\leq 2} && \text{(propriété 5.1.5)} \\ &\subseteq \mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R)^{\leq 2} && \text{(définition de la clôture compatible)} \\ &\subseteq \mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R)^{=}.\end{aligned}$$

□

Lemme 5.1.2. *La relation* $\mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R)^{=}$ *est compatible.*

Démonstration. Soit S la clôture contextuelle de R, c'est-à-dire $\mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R)$. On a

$$\begin{aligned}\Sigma_{\mathsf{MaP}}(S^{=}) \subseteq S^{=} &\iff {\cdot^{\circ}}\,;S^{=} \times S^{=}\,;\cdot \subseteq S^{=}\\ &\impliedby (S \times S)^{=} \subseteq {\cdot^{\circ}} \rhd S^{=} \lhd \cdot\\ &\impliedby \mathit{Id} \cup S \times S \cup {\cdot^{\circ}} \rhd S^{=} \lhd \cdot\,; {\cdot^{\circ}} \rhd S^{=} \lhd \cdot \subseteq {\cdot^{\circ}} \rhd S^{=} \lhd \cdot\end{aligned}$$

On doit donc montrer trois inclusions. La multiplication du magma pointé est une fonction, on a donc les inclusions ${\cdot^{\circ}}\,;\cdot \subseteq \mathit{Id}$ et $\mathit{Id} \subseteq \cdot\,;{\cdot^{\circ}}$, desquelles découlent la première et la troisième inclusion. La deuxième inclusion correspond au lemme 5.1.1. □

Lemme 5.1.3. *La clôture congruencielle d'une relation est contextuelle.*

Démonstration. On doit montrer $\Sigma^{\square}_{\mathsf{MaP}}(\Sigma^{=}_{\mathsf{MaP}}(R)) \subseteq \Sigma^{=}_{\mathsf{MaP}}(R)$. On a

$$\begin{aligned}\Sigma^{\square}_{\mathsf{MaP}}(\Sigma^{=}_{\mathsf{MaP}}(R)) &= {\cdot^{\circ}}\,;(\Sigma^{=}_{\mathsf{MaP}}(R) \times \mathit{Id} \cup \mathit{Id} \times \Sigma^{=}_{\mathsf{MaP}}(R))\,;\cdot\\ &\subseteq {\cdot^{\circ}}\,;(\Sigma^{=}_{\mathsf{MaP}}(R) \times \Sigma^{=}_{\mathsf{MaP}}(R) \cup \Sigma^{=}_{\mathsf{MaP}}(R) \times \Sigma^{=}_{\mathsf{MaP}}(R))\,;\cdot\\ &\subseteq \Sigma_{\mathsf{MaP}}(\Sigma^{=}_{\mathsf{MaP}}(R))\\ &\subseteq \Sigma^{=}_{\mathsf{MaP}}(R).\end{aligned}$$

□

Corollaire 5.1.2. *Les relations* $\mathsf{Cont}[\Sigma_{\mathsf{MaP}}](R)^{=}$ *et* $\Sigma^{=}_{\mathsf{MaP}}(R)$ *sont égales.*

Démonstration. Par le lemme 5.1.2, la clôture symétrique-réflexive-transitive de la clôture compatible de R est compatible, et c'est une relation d'équivalence. Elle comprend donc la clôture congruencielle de R. Par le lemme 5.1.3, la clôture congruencielle de R est contextuelle. C'est une relation d'équivalence. Donc elle comprend la clôture symétrique-réflexive-transitive de la clôture contextuelle de R. □

5.1.4. Présentation par un système de réécriture

La construction de $\mathcal{F}[\mathsf{Mon}](\Gamma)$ que nous venons de donner constitue une *présentation par générateurs et relations* de ce monoïde. Le terme *présentation* signifie qu'il s'agit d'une façon assez concrète de décrire ce monoïde, parmi d'autres. Les *générateurs* sont les éléments de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$. Les *relations* sont les couples de R_{Mon}, complétés pour obtenir une congruence par laquelle on va quotienter l'ensemble des générateurs.

D'un point de vue informatique, disposer d'une présentation par générateurs et relations est agréable puisque l'ensemble des générateurs est constitué d'objets syntaxiques faciles à manipuler par des programmes. Cependant, la manipulation informatique des relations est plus délicate. On aimerait en particulier savoir décider la question suivante : étant donnés n'importe quels termes t, u de $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$, a-t-on $t \equiv_{\mathsf{Mon}} u$? Autrement dit, dispose-t-on d'un algorithme qui décide l'égalité dans $\mathcal{F}[\mathsf{Mon}](\Gamma)$? Cette question est souvent appelée *problème du mot* pour notre présentation du monoïde libre.

Pour résoudre ce problème, on va se tourner vers une approche algorithmique générale. L'idée est d'*orienter* la relation d'équivalence $\equiv_{\mathsf{Mon}}$, de sorte à obtenir ce qu'on appelle un *système de réécriture* sur $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$. Ce système sera doté de bonnes propriétés qui permettent de décider le problème du mot, et d'implémenter simplement la multiplication du monoïde libre.

Considérons la relation R_{Mon} définie plus haut. Celle-ci n'est visiblement pas symétrique. L'intuition est que si (t, t') appartient à R_{Mon}, on peut penser à t' comme à t simplifié. On note $\to_{\mathsf{Mon}}$ la clôture contextuelle de R_{Mon}. Explicitement, il s'agit de la plus petite relation close par les règles suivantes.

$$\frac{}{(t \cdot u) \cdot v \to_{\mathsf{Mon}} t \cdot (u \cdot v)} \qquad \frac{}{\varepsilon \cdot t \to_{\mathsf{Mon}} t} \qquad \frac{}{t \cdot \varepsilon \to_{\mathsf{Mon}} t}$$

$$\frac{t \to_{\mathsf{Mon}} t'}{t \cdot u \to_{\mathsf{Mon}} t' \cdot u} \qquad \frac{u \to_{\mathsf{Mon}} u'}{t \cdot u \to_{\mathsf{Mon}} t \cdot u'}$$

Intuitivement, cette relation est la plus petite relation qui permet l'application de $\to_{\mathsf{Mon}}$ à un unique sous-terme arbitraire.

Propriété 5.1.6. *La relation $\equiv_{\mathsf{Mon}}$ est la plus petite relation d'équivalence qui contient $\to_{\mathsf{Mon}}$.*

Démonstration. Immédiat par le corollaire 5.1.2. □

On va utiliser les résultats du chapitre 4 pour démontrer que cette relation est fortement normalisante et confluente, en commençant par la normalisation forte. Pour cela, on va utiliser le lemme 4.3.2.

On définit pour chaque terme t un entiers naturel, sa *mesure*, notée $\mathsf{mes}(t)$.

$$\begin{aligned} \mathsf{mes}(\mathtt{op}[\cdot]()) &= 1 \\ \mathsf{mes}(\mathtt{var}(x)) &= 1 \\ \mathsf{mes}(\mathtt{op}[\cdot](t, u)) &= 2\mathsf{mes}(t) + \mathsf{mes}(u) \end{aligned}$$

Il suffit de vérifier que cette mesure décroit strictement à chaque étape de réduction, puisqu'il n'existe pas de suite infinie d'entiers naturels qui soit strictement décroissante.

Lemme 5.1.4. *Si (t, t') appartient à R_{Mon} alors $\mathsf{mes}(t) < \mathsf{mes}(t')$.*

Démonstration. On commence par remarquer qu'une induction évidente montre que la mesure d'un terme est toujours strictement positive. Soient t et t' tels que $(t, t') \in R_{\mathsf{Mon}}$. On raisonne par cas.

- Si $t = t' \cdot \varepsilon$ ou $t = \varepsilon \cdot t'$, alors on a $\mathsf{mes}(t) = 2\mathsf{mes}(t) + 1 > \mathsf{mes}(t')$.
- Si $t = ((u_1 \cdot u_2) \cdot u_3)$ et $t' = \mathsf{deg}(u_1 \cdot (u_2 \cdot u_3))$, on a

$$\begin{aligned} \mathsf{mes}(t) = \mathsf{mes}((u_1 \cdot u_2) \cdot u_3) &= 4\mathsf{mes}(u_1) + 2\mathsf{mes}(u_2) + \mathsf{mes}(u_3) \\ &> 2\mathsf{mes}(u_1) + 2\mathsf{mes}(u_2) + \mathsf{mes}(u_3) \quad (MEAS(u_1) > 0) \\ &= \mathsf{mes}(u_1 \cdot (u_2 \cdot u_3)) = \mathsf{mes}(t'). \end{aligned}$$

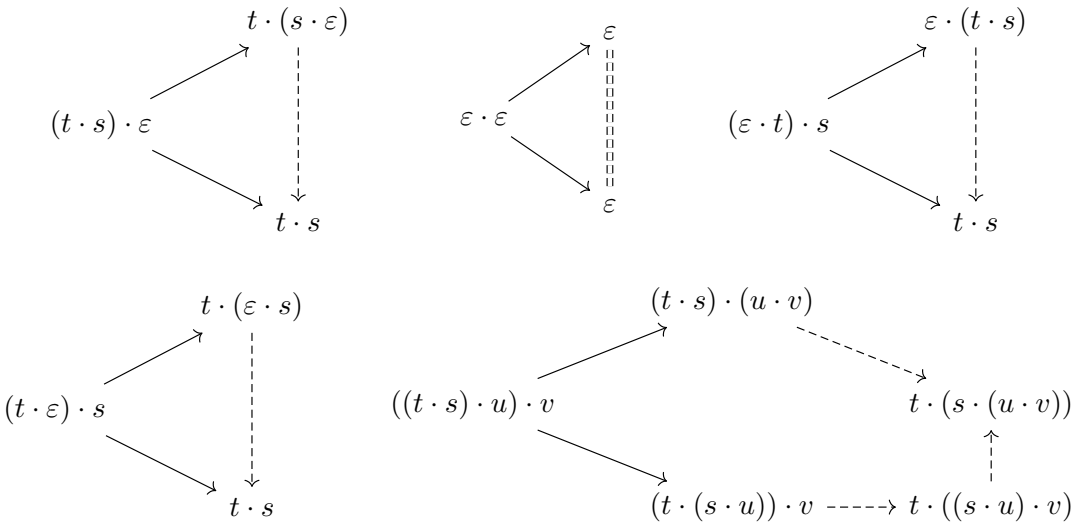


FIG. 5.3. : cas principaux de la confluence locale de $\rightarrow_{\mathsf{Mon}}$

□

Propriété 5.1.7. *Si* $\mathsf{mes}(t) > \mathsf{mes}(t')$ *pour tous* (t, t') *appartenant à* R*, alors il en va de même pour tous* (u, u') *appartenant à la clôture contextuelle de* R*.*

Démonstration. Par une induction de routine, en utilisant le fait que $\mathsf{mes}(t) > 0$ pour tout t. □

Lemme 5.1.5. *Si* $t \rightarrow_{\mathsf{Mon}} t'$ *alors* $\mathsf{mes}(t) > \mathsf{mes}(t')$*.*

Démonstration. On raisonne par induction sur $t \rightarrow_{\mathsf{Mon}} t'$. Les cas de base correspondent au 5.1.5, le cas de l'induction de la propriété 5.1.7. □

Corollaire 5.1.3. *La relation* $\rightarrow_{\mathsf{Mon}}$ *est fortement normalisante.*

On peut maintenant déduire la confluence du système de réécriture de sa confluence locale en utilisant le lemme de Newman.

Lemme 5.1.6. *La relation* $\rightarrow_{\mathsf{Mon}}$ *est localement confluente.*

Ébauche de démonstration. Pour démontrer la confluence locale, il suffit d'étudier les *paires critiques* du système, c'est-à-dire les paire (u, v) dans $\rightarrow^{\circ}_{\mathsf{Mon}}$; $\rightarrow_{\mathsf{Mon}}$ obtenues en réduisant des parties du terme d'origine qui se recouvrent. Dans le cas d'un système de réécriture fortement normalisant, on peut montrer que si toutes les paires critiques peuvent être jointes, alors le système est localement confluent (et donc confluent par le lemme de Newman). Les branchements correspondant aux paires critiques de $\rightarrow_{\mathsf{Mon}}$ sont représentés par des flèches pleines à la figure 5.3. On peut joindre chacun de ces branchements via les flèches en pointillés, le système est donc localement confluent. □

Corollaire 5.1.4. *La relation* $\to_{\mathsf{Mon}}$ *est confluente.*

Grâce à ces deux propriétés, on sait que tout élément du magma pointé libre à une forme normale par $\to_{\mathsf{Mon}}$, notée $\widehat{t}$, qu'on peut calculer par application maximale du système de réécriture. Par la propriété de Church et Rosser, on sait aussi que $t \equiv_{\mathsf{Mon}} u$ si et seulement si $\widehat{t}$ est égal à $\widehat{u}$. On a résolu le problème du mot.

On peut donc implémenter le monoïde libre sur Γ comme l'ensemble des arbres de syntaxe $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ modulo le quotient par la relation d'équivalence $\equiv_{\mathsf{Mon}}$ dont on vient de voir qu'elle est décidable. La multiplication du monoïde est donnée par l'opération syntaxique $\mathtt{op}[\cdot]$ de la structure de magma pointé sous-jacente, puis par le quotient.

Le système de réécriture suggère une autre implémentation du monoïde libre que celle qui consiste à prendre comme éléments tout l'ensemble $\mathcal{T}[\Sigma_{\mathsf{MaP}}](\Gamma)$ puis à effectuer un quotient. Pour éviter ce quotient, on peut adopter directement les formes normales n, m de $\to_{\mathsf{Mon}}$ comme éléments du monoïde libre. Ces formes normales obéissent à la grammaire suivante.

$$\begin{aligned} n, m &::= \mathtt{op}[\varepsilon]() \mid \underline{p} && \text{(listes potentiellement vides)} \\ p, q &::= \mathtt{var}(x) \mid \mathtt{op}[\cdot](\mathtt{var}(x), p) && \text{(listes non vides)} \end{aligned}$$

Pour munir cet ensemble de la multiplication, on peut simplement définir $m \cdot n$ comme la forme normale de $\mathtt{op}[\cdot](m, n)$. Mais on peut aussi constater que cette définition se réduit entièrement aux équations suivantes. On y distingue la multiplication des listes potentiellement vides et celle des listes vides, notée $\cdot'$.

$$\begin{aligned} \mathtt{op}[\varepsilon]() \cdot n &= n & \mathtt{var}(x) \cdot' q &= \mathtt{op}[\cdot](\mathtt{var}(x), q) \\ \underline{p} \cdot \mathtt{op}[\varepsilon]() &= p & \mathtt{op}[\cdot](\mathtt{var}(x), p) \cdot' q &= \mathtt{op}[\cdot](\mathtt{var}(x), p \cdot' q) \\ \underline{p} \cdot \underline{q} &= p \cdot' q \end{aligned}$$

On obtient ainsi une définition du monoïde libre sans quotient, et où le test d'égalité des éléments se réduit à l'égalité syntaxique. En contrepartie, la multiplication est devenue une opération avec un contenu calculatoire non-trivial, puisqu'elle doit maintenir le caractère normal des termes.

Remarque 5.1.7. Les formes normales de $\to_{\mathsf{Mon}}$ ressemblent au type des listes imbriquées à droite qu'on trouve des langages fonctionnels, et la multiplication à la concaténation. Cependant, on a ici un cas particulier pour la liste de longueur unitaire $\mathtt{var}(x)$.

Remarque 5.1.8. On a vu que le monoïde libre sur un ensemble de générateurs correspond, informatiquement parlant, au type de données abstrait des listes. C'est également le cas des monoïdes commutatifs libres, c'est-à-dire des monoïdes qui satisfont librement l'équation supplémentaire $t \cdot s = s \cdot t$. Le type de données abstrait correspondant est celui des ensembles finis avec répétitions, ou *multiensembles finis*. On en trouve une implémentation dans la bibliothèque de conteneurs du langage C++, par exemple, et on peut la simuler en OCaml avec le module `Map` ; à noter que cela suppose que l'ensemble des générateurs est muni d'une relation d'ordre totale décidable. Si on ajoute à la commutativité l'idempotence, c'est-à-dire l'équation $t \cdot t = t$, on obtient le type de données abstrait des ensembles finis, également bien connu.

5.1.5. Exercices

* **Ex. 24** — Démontrer les équations 5.7 et 5.8 en utilisant uniquement la propriété universelle de $\mathcal{T}[\Sigma_{\mathsf{MaP}}]$.

* **Ex. 25** — On aimerait exprimer que l'application d'une substitution vérifie les mêmes propriétés de fonctorialité que l'application d'un renommage. Toutefois, il est impossible de reproduire l'équation équation (5.8) telle quelle. Pourquoi ? Proposer une formulation alternative et la démontrer.

* **Ex. 26** — Démontrer les propriétés 2.5.3 et 5.1.2.

5.2. Le cas général

Le but de cette section est de montrer en quoi les magmas pointés et les monoïdes tels qu'étudiés à la section précédente ne sont qu'un cas particulier de la théorie générale des théories algébriques du premier ordre. Dans ce qui suit, on utilise intensivement les produits et coproduits ensemblistes ; voire la section 2.4 pour les définitions attenantes.

Définition 5.2.1. Une *signature du premier ordre* Σ est la donnée d'un ensemble $|\Sigma|$ appelé *opérateurs de* Σ et d'une fonction $\mathsf{ar}_\Sigma : |\Sigma| \to \mathbb{N}$ appelée *arité de* Σ.

Toute signature du premier ordre Σ agit sur un ensemble A pour lui associer un nouvel ensemble que l'on note $\Sigma(A)$. Cet ensemble est défini par l'équation

$$\Sigma(A) \quad \coloneqq \quad \coprod_{f\in|\Sigma|} \prod_{i=1}^{\mathsf{ar}(f)} A.$$

Cette action s'étend aux relations, au sens où si R est une relation de A dans B, on définit une relation $\Sigma(R)$ de $\Sigma(A)$ dans $\Sigma(B)$ par

$$\Sigma(R) \quad \coloneqq \quad \coprod_{f\in|\Sigma|} \prod_{i=1}^{\mathsf{ar}(f)} R \quad = \quad \bigcup_{f\in|\Sigma|} \{(f, ((x_1, \dots, x_{\mathsf{ar}(f)}), (y_1, \dots, y_{\mathsf{ar}(f)}))) \mid (x_i, y_i) \in R\}.$$

On vérifie facilement que $\Sigma(-)$ préserve la fonctionalité des relations.

Exemple 5.2.1. La signature des magmas pointés Σ_{MaP} est définie par $|\Sigma_{\mathsf{MaP}}| = \{\cdot, \varepsilon\}$ ainsi que $\mathsf{ar}(\cdot) = 2$ et $\mathsf{ar}(\varepsilon) = 0$. Pour tout ensemble A, on a $\Sigma_{\mathsf{MaP}}(A) = A^{\mathsf{ar}(\varepsilon)} + A^{\mathsf{ar}(\cdot)} = \mathbb{1} + A^2$. Pour toute relation $R : A \nrightarrow B$, on a

$$\begin{aligned}\Sigma_{\mathsf{MaP}}(R) &= \mathbb{1} + R^2 \\ &= \{((\varepsilon, ()), (\varepsilon, ()))\} \cup \{((\cdot, (x_1, x_2)), (\cdot, (y_1, y_2))) \mid (x_1, y_1), (x_2, y_2) \in R\}.\end{aligned}$$

Définition 5.2.2. Une Σ-*algèbre* X est la donnée d'un ensemble $|X|$ appelé *support* de X et d'une fonction $\alpha_X : \Sigma\,|X| \to |X|$ appelée *action* de l'algèbre.

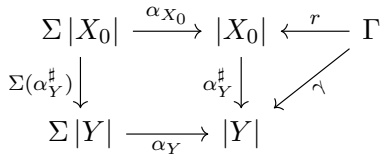


FIG. 5.4. : propriété universelle de la Σ-algèbre libre sur Γ.

Exemple 5.2.2. Une Σ_{MaP}-algèbre M est la donnée d'un ensemble $|M|$ et d'une fonction $\alpha_M : \mathbb{1} + M^2 \to M$. Mais pour tous ensembles A, B, C, l'ensemble $A + B \to C$ des fonctions de $A + B$ dans C est en bijection avec l'ensemble $A \to C \times B \to C$. En particulier, α_X donne lieu à une fonction de $\mathbb{1}$ dans $|M|$, c'est-à-dire un élément de $|M|$, et une fonction de $|M|^2$ dans $|M|$. Une Σ_{MaP}-algèbre est donc la même chose qu'un magma pointé.

Définition 5.2.3. Soient X et Y deux Σ-algèbres. Un *homomorphisme* de X dans Y est une fonction $f : |X| \to |Y|$ telle que $\alpha_X \,;\, f = \Sigma(f) \,;\, \alpha_Y$.

Exemple 5.2.3. Si M et N sont des Σ_{MaP}-algèbre, un homomorphisme de M dans N est une fonction f de $|M|$ dans $|N|$ telle que $\alpha_M \,;\, f = (\mathbb{1} + f^2) \,;\, \alpha_N$. Cette équation s'exprime en deux équations comme $\cdot_M \,;\, f = f^2 \,;\, \cdot_N$ et $\varepsilon_M \,;\, f = \varepsilon_N$. Si on écrit ces équations en termes d'éléments, on retrouve les conditions $f(x \cdot_M y) = f(x) \cdot_N f(y)$ et $f(\varepsilon_M) = \varepsilon_N$ qu'on a utilisé pour définir les homomorphismes de magma pointés en début de chapitre.

Définition 5.2.4. Une Σ-*algèbre libre* sur un ensemble Γ dits *de générateurs* est la donnée d'une Σ-algèbre X_0 et d'une fonction $r : \Gamma \to |X_0|$ telle que pour toute Σ-algèbre Y et fonction $\gamma : |X_0| \to |Y|$ il existe un unique homomorphisme $\gamma^\sharp : X_0 \to Y$ tel que $r \,;\, \gamma^\sharp = \gamma$ (voir figure 5.4).

Exemple 5.2.4. Une Σ_{MaP}-algèbre libre sur Γ est exactement un magma pointé libre sur Γ au sens de la définition 5.1.3.

Définition 5.2.5. Soit Σ une signature du premier ordre Σ et Γ un ensemble. Si A est un ensemble, on note $F_\Gamma(A)$ l'ensemble $\Gamma + \Sigma(A)$. L'ensemble des Σ-*termes* sur Γ, noté $\mathcal{T}[\Sigma](\Gamma)$, est défini par

$$\mathcal{T}[\Sigma](\Gamma) \quad := \quad \bigcup_{n \geq 0} F_\Gamma(\emptyset)^n.$$

Il est utile d'expliciter les premières étapes de la construction de l'ensemble des termes

donnée ci-dessus pour une signature concrète, par exemple celle des magma pointés.

$$\begin{aligned}
F_\Gamma(\emptyset)^0 &= \emptyset \\
F_\Gamma(\emptyset)^1 &= \{(1,x) \mid x \in \Gamma\} \cup \{(2,(f,(x_1,\dots,x_{\mathsf{ar}(f)}))) \mid f \in |\Sigma_{\mathsf{MaP}}|, x_i \in F_\Gamma(\emptyset)^0\} \\
&= \{(1,x) \mid x \in \Gamma\} \cup \{(2,(\varepsilon,()))\} \\
F_\Gamma(\emptyset)^2 &= \{(1,x) \mid x \in \Gamma\} \cup \{(2,(f,(x_1,\dots,x_{\mathsf{ar}(f)}))) \mid f \in |\Sigma_{\mathsf{MaP}}|, x \in F_\Gamma(\emptyset)^1\} \\
&= \{(1,x) \mid x \in \Gamma\} \cup \{(2,(\varepsilon,()))\} \cup \{(2,(\cdot,(x_1,x_2))) \mid x_1,x_2 \in F_\Gamma(\emptyset)^1\} \\
F_\Gamma(\emptyset)^{n+1} &= \{(1,x) \mid x \in \Gamma\} \cup \{(2,(f,(x_1,\dots,x_{\mathsf{ar}(f)}))) \mid f \in |\Sigma_{\mathsf{MaP}}|, x \in F_\Gamma(\emptyset)^n\} \\
&= \{(1,x) \mid x \in \Gamma\} \cup \{(2,(\varepsilon,()))\} \cup \{(2,(\cdot,(x_1,x_2))) \mid x_1,x_2 \in F_\Gamma(\emptyset)^n\}
\end{aligned}$$

On voit donc que l'ensemble $F_\Gamma(\emptyset)^{n+1}$ est celui des arbres de syntaxe de hauteur n dont les nœuds sont ceux décrits par la signature Σ. L'ensemble $\mathcal{T}[\Sigma](\Gamma)$ contient donc les arbres de hauteur finie arbitraire dont les nœuds sont décrits par la signautre Σ, y compris leur facteur de branchement.

Dans ce qui suit, et en cohérence avec la section précédente, on note $\mathtt{var}(x)$ pour $(1,x)$ et $\mathtt{op}[f](x_1,\dots,x_{\mathsf{ar}(f)})$ pour $(2,(f,(x_1,\dots,x_{\mathsf{ar}(f)})))$. On note $\mathtt{var}(-)$ la fonction de Γ dans $\mathcal{T}[\Sigma](\Gamma)$ qui envoie x dans $\mathtt{var}(x)$. La fonction

$$\begin{aligned}
\Sigma(\mathcal{T}[\Sigma](\Gamma)) &\to \mathcal{T}[\Sigma](\Gamma) \\
(f,(x_1,\dots,x_{\mathsf{ar}(f)})) &\mapsto \mathtt{op}[f](x_1,\dots,x_{\mathsf{ar}(f)})
\end{aligned}$$

munit $\mathcal{T}[\Sigma](\Gamma)$ d'une structure de Σ-algèbre. On la note $\mathtt{op}[-]$.

Théorème 5.2.1. *La paire $(\mathcal{T}[\Sigma](\Gamma), \mathtt{var}(-))$ est une Σ-algèbre libre sur Γ.*

Ébauche de démonstration. La propriété universelle est essentiellement une conséquence du fait que l'opération $\Sigma(-)$ préserve les unions croissantes d'ensembles. Cela assure que $\mathcal{T}[\Sigma](\Gamma)$ est un point fixe, en un sens ensembliste, de F_Γ. Ce point fixe est minimal pour l'inclusion, ce dont découle le théorème. □

En utilisant la propriété universelle de $\mathcal{T}[\Sigma]$, on peut dériver les mêmes mécanismes que dans le cas particulier des magmas pointés. L'utilisation d'une mécanique générale simplifie souvent les notations ; comparer la propriété ci-dessous à la propriété 5.1.1.

Propriété 5.2.1. *Soit P une partie de $\mathcal{T}[\Sigma](\Gamma)$ qui satisfait les conditions suivantes.*

1. *Pour tout x dans Γ, le terme $\mathtt{var}(x)$ appartient à P.*

$$\Gamma\,;\mathtt{var}(-) \subseteq P : \mathbb{1} \dashrightarrow \mathcal{T}[\Sigma](\Gamma)$$

2. *Pour tout élément a de $\Sigma(P)$, le terme $\mathtt{op}[-](a)$ de $\mathcal{T}[\Sigma](\Gamma)$ appartient à P.*

$$\Sigma(P)\,;\mathtt{op}[-] \subseteq P : \mathbb{1} \dashrightarrow \mathcal{T}[\Sigma](\Gamma)$$

Alors P est l'ensemble $\mathcal{T}[\Sigma](\Gamma)$ tout entier.

$t, s ::=$	**termes du premier ordre**
$\mid \mathtt{var}(x)$	variable
$\mid \mathtt{op}[f](t_1, \ldots, t_n)$	application d'opérateur

(a) grammaire des termes du premier ordre

$$\boxed{\Sigma \mid \Gamma \vdash t : \mathbb{T}} \qquad \frac{\Gamma \ni \Gamma}{\Sigma \mid \Gamma \vdash \mathtt{var}(x) : \mathbb{T}} \qquad \frac{\mathsf{ar}(f) = n \qquad (\Sigma \mid \Gamma \vdash t_i : \mathbb{T})_{1 \leq i \leq n}}{\Sigma \mid \Gamma \vdash \mathtt{op}[f](t_1, \ldots, t_n) : \mathbb{T}}$$

(b) bonne formation des termes du premier ordre

$$\begin{aligned} \mathtt{var}(x)[\sigma] &= \sigma(x) \\ \mathtt{op}[f](t_1, \ldots, t_n)[\sigma] &= \mathtt{op}[f](t_1[\sigma], \ldots, t_n[\sigma]) \end{aligned}$$

(c) substitution

FIG. 5.5. : Termes du premier ordre génériques

On conclut par un résumé de la structure des termes du premier ordre sur la signature Σ avec des notations agréables à la figure 5.5. En particulier, on note $\Sigma \mid \Gamma \vdash t : \mathbb{T}$ pour l'appartenance $t \in \mathcal{T}[\Sigma](\Gamma)$. La définition inductive donnée à la figure 5.5b est le pendant de la construction ensembliste donnée précédemment. On rappelle également les équations définissant le résultat de l'application d'une substitution σ à un terme t, notée $t[\sigma]$, à la figure 5.5c. Cette opération est obtenue par application de la propriété universelle des Σ-termes sur Γ, ce qui donne automatiquement le lemme suivant.

Lemme 5.2.1. *Si σ est une fonction de Γ dans $\mathcal{T}[\Sigma](\Delta)$ et que t appartient à $\mathcal{T}[\Sigma](\Gamma)$ alors $t[\sigma]$ appartient à $\mathcal{T}[\Sigma](\Delta)$.*

6. Syntaxe abstraite avec lieurs

Dans ce chapitre on s'intéresse à une généralisation des termes du premier ordre capable de traiter la syntaxe des langages de programmation. Il s'agit de traiter les opérateurs qui introduisent de nouvelles variables locales à un sous-termes. Ces opérations sont appelées des *lieurs* et sont essentielles en programmation, par exemple pour décrire la possibilité de définir des fonctions. Ce chapitre décrit en détail le traitement général des lieurs.

6.1. Introduction

Pour illustrer ce chapitre, on revient sur l'exemple vu au chapitre 5.

Considérons le monoïde libre muni du système de réécriture étudié à la section 5.1.4. On a vu que ce monoïde est isomorphe à celui des listes finies sur les générateurs, et que le système de réécriture implémente essentiellement la concaténation des listes. Tout élément de ce monoïde, obtenu par application répétée de opérations, représente donc un calcul dont la forme normale est le résultat.

On vient de décrire un langage calculatoire. Toutefois, on n'oserait parler de langage de *programmation* étant donné la pauvreté des opérations disponibles. La première opération à ajouter pour se rapprocher d'un langage est celle permettant la réutilisation d'un calcul, c'est-à-dire la définition d'une variable locale. En d'autres termes, on veut étudier le langage décrit par la grammaire suivante. Par souci de lisibilité, on y omet les injections $\mathtt{var}(-)$ et $\mathtt{op}[-]$ et note la multiplication de façon infixe.

$$n, m ::= x \mid \varepsilon \mid n \cdot m \mid \mathtt{let}(n, x.m)$$

Le terme $\mathtt{let}(n, x.m)$ désigne le terme m où toute instance de la variable x désigne le terme n. On peut y penser comme à l'expression OCaml notée `let x = n in m`, à ceci près que la notation $x.m$ rapproche x de m plutôt que de n, pour mettre l'accent sur la *portée* de x, restreinte au sous-terme m. En accord avec cette description, on souhaite munir cette syntaxe du système de réécriture engendré par la clôture contextuelle de la relation primitive $\to_\beta$ définie ci-dessous.

$$\varepsilon \cdot m \to_\beta m \tag{6.1}$$
$$n \cdot \varepsilon \to_\beta n \tag{6.2}$$
$$(n \cdot m) \cdot p \to_\beta n \cdot (m \cdot p) \tag{6.3}$$
$$\mathtt{let}(n, x.m) \to_\beta m[x \backslash n] \tag{6.4}$$

Remarque 6.1.1. La règle de réécriture 6.4 montre que la définition locale de variable *internalise* la substitution, au sens où elle reflète dans la syntaxe d'un langage objet une

opération d'abord définie à l'extérieur de celui-ci, dans le langage sujet, c'est-à-dire ici dans les mathématiques usuelles.

À ce stade, il semblerait naturel de discuter de la normalisation forte et de la confluence locale et globale de ce système de réécriture. Ce serait toutefois sauter les étapes : ayant enrichi la syntaxe des opérateurs, il nous faut réexaminer l'opération clef qu'est la substitution. C'est ce qu'on va faire à la section suivante. On va régler définitivement ce problème en traitant de façon générique tout *lieur*, c'est-à-dire qui introduit une variable locale à un sous-terme.

6.2. Termes du second ordre

6.2.1. Signatures et prétermes

La figure 6.1 décrit de manière synthétique les Σ-*termes* où Σ est une signature du second ordre. Cette définition généralise celle de terme du premier ordre vue au chapitre 5. On commence par introduire la notion de *sorte*. Une sorte est une formule très simple qui décrit la forme d'un terme du second ordre. La sorte $\mathbb{T}$ désigne un terme au sens usuel. La sorte $\mathbb{V} \to \mathfrak{S}$ est celle d'un terme de sorte $\mathfrak{S}$ ayant accès à une variable libre supplémentaire.

Remarque 6.2.1. D'après la grammaire présentée à la figure 6.1a, une sorte est une formule de la forme $\mathbb{V} \to \ldots \to \mathbb{V} \to \mathbb{T}$. Elle se résume donc entièrement à son nombre de flèches. Pour cette raison, on la trouve parfois présentée comme un entier naturel dans la littérature. La notation sous forme de formule est plus lourde mais plus lisible, et s'étend à des situations plus riches.

Dans ce qui suit, on note A^* l'ensemble des listes finies potentiellement vides dont les éléments appartiennent à l'ensemble A. Bien que cette notation entre en conflit avec celle utilisée pour la clôture réflexive-transitive d'une relation, on ne l'emploiera jamais dans des contextes où cela peut prêter à confusion.

Définition 6.2.1. Une *signature du second ordre* Σ est la donnée d'un ensemble $|\Sigma|$ appelé *opérateurs de* Σ et d'une fonction $ar_\Sigma : |\Sigma| \to \mathsf{Sort}^*$ appelée *arité de* Σ.

On notera $\Sigma \ni f : \mathfrak{S}_1 \to \ldots \to \mathfrak{S}_n \to \mathbb{T}$ pour signifier que la signautre Σ contient un opérateur f dont l'arité $\mathsf{ar}(f)$ est la liste $\mathfrak{S}_1, \ldots, \mathfrak{S}_\mathfrak{n}$. Cette notation est suggestive : un opérateur f est un symbole formel qui s'applique à n arguments dont le $i$ème doit être de sorte $\mathfrak{S}_i$. Le cas d'une signature est du premier ordre est précisément celui où la sorte $\mathfrak{S}_i$ est égale à $\mathbb{T}$ pour tout i.

Exemple 6.2.1. La signature Σ_{MaP} des magmas pointés du chapitre 5 est formée des opérateurs $\varepsilon : \mathbb{T}$ et $\cdot : \mathbb{T} \to \mathbb{T} \to \mathbb{T}$. La définition de variable locale décrite en introduction correspond à l'opérateur $\texttt{let} : \mathbb{T} \to (\mathbb{V} \to \mathbb{T}) \to \mathbb{T}$. On note Σ_{MaPL} la signature Σ_{MaP} étendue avec cet opérateur.

Exemple 6.2.2. Le λ-*calcul pur* est un tout petit langage de programmation théorique qui a été très étudié par les logiciens et informaticiens théoriciens. Sa syntaxe est engendrée par la signature Σ_λ formée des opérateurs $\texttt{app} : \mathbb{T} \to \mathbb{T} \to \mathbb{T}$ et $\texttt{fun} : (\mathbb{V} \to \mathbb{T}) \to \mathbb{T}$.

$\mathsf{Sort} \ni \mathfrak{S}, \mathfrak{T} ::=$		**Sortes**
	$\mid \mathbb{T}$	sorte des termes
	$\mid \mathbb{V} \to \mathfrak{S}$	liaison générique de variable
$\mathsf{PreTerm} \ni t, s ::=$		**Prétermes du second ordre**
	$\mid \mathtt{var}(x)$	variable
	$\mid \mathtt{op}[f](t_1, \ldots, t_n)$	application d'opérateur
	$\mid x.t$	lieur générique
$\mathsf{PreSub} \ni \sigma, \varphi ::=$		**Présubstitutions**
	$\mid \mathtt{id}$	Substitution identité
	$\mid \sigma, x\backslash t$	Substitution étendue avec définition

(a) grammaire des termes du second ordre

$$\boxed{\Sigma \mid \Gamma \vdash t : \mathfrak{S}} \qquad \frac{\Gamma \ni x}{\Sigma \mid \Gamma \vdash \mathtt{var}(x) : \mathbb{T}} \qquad \frac{\Sigma \mid \Gamma, x \vdash t : \mathfrak{S}}{\Sigma \mid \Gamma \vdash x.t : \mathbb{V} \to \mathfrak{S}}$$

$$\frac{\Sigma \ni f : \mathfrak{S}_1 \to \ldots \to \mathfrak{S}_n \to \mathbb{T} \qquad (\Sigma \mid \Gamma \vdash t_i : \mathfrak{S}_i)_{1 \le i \le n}}{\Sigma \mid \Gamma \vdash \mathtt{op}[f](t_1, \ldots, t_n) : \mathbb{T}}$$

(b) jugement de bonne formation des prétermes du second ordre

$$\begin{aligned}
\mathtt{id}(x) &= \mathtt{var}(x) \\
(\sigma, x\backslash t)(x) &= t \\
(\sigma, y\backslash t)(x) &= \sigma(x) && (x \neq y)
\end{aligned}$$

$$\begin{aligned}
\mathtt{var}(x)[\sigma] &= \sigma(x) \\
\mathtt{op}[f](t_1, \ldots, t_n)[\sigma] &= \mathtt{op}[f](t_1[\sigma], \ldots, t_n[\sigma]) \\
(x.t)[\sigma] &= y.t[\sigma, x\backslash y] && (y = \nu(t, \sigma))
\end{aligned}$$

$$\begin{aligned}
\mathtt{id}[\sigma] &= \mathtt{id} \\
(\varphi, x\backslash t)[\sigma] &= \varphi[\sigma], x\backslash t[\sigma]
\end{aligned}$$

(c) substitution hygiénique

FIG. 6.1. : Termes du second ordre génériques

La définition des termes du second ordre sur une signature procède en deux étapes. On va d'abord décrire les *prétermes* sur cette signature, avant de raffiner cette notion par un quotient qui va prendre son sens dans ce qui suit.

On commence par se donner un ensemble infini dénombrable de noms, noté $\mathbb{A}$. Ses éléments sont notés x, y, z, x', x_1, x_2, etc. On se donne également une fonction ν qui à toute partie finie A de $\mathbb{A}$ associe un élément $\nu(A)$ de $\mathbb{A}$ tel que $\nu(A) \notin A$. Le choix de ν n'influe pas sur les définitions finales. Toutefois, le lecteur ou la lectrice à la recherche d'objets concrets peut par définir $\mathbb{A}$ comme l'ensemble des entiers naturels et par exemple poser $\nu(A) = 1 + \max A$.

Définition 6.2.2. Soit Σ une signature du second ordre et Γ un sous-ensemble de $\mathbb{A}$. L'ensemble des Σ-*prétermes* sur Γ est défini comme l'ensemble des termes de la grammaire de la figure 6.1a qui satisfont le prédicat $\Sigma \mid \Gamma \vdash t : \mathbb{T}$ dérivable en appliquant les règles du jugement défini inductivement à la figure 6.1b.

Dans le cas de la gestion d'un lieur présentée à la figure 6.1b, on suppose implicitement que x n'appartient pas à Γ. Si x appartient à Γ, on s'interdit d'appliquer la règle. On va voir tout de suite que cette restriction n'en est pas une en pratique, puisqu'on va toujours raisonner modulo renommage des variables liées. Pour donner un sens précis à cette assertion, il faut étudier la notion de substitution.

6.2.2. Substitution

6.2.2.1. Substitution naïve et hygiénique

On va adopter une représentation concrète des substitutions, contrairement au chapitre 5 où elles étaient modélisées par des fonctions. De même que pour les termes, on va commencer par définir les *pré*substitutions. La grammaire donnée à la figure 6.1a considère qu'une présubstitution est essentiellement une liste de définitions. On abrège en $x_1 \backslash t_1, \ldots, x_n \backslash t_n$ la présubstitution $\mathtt{id}, x_1 \backslash t_1, \ldots, x_n \backslash t_n$. On note $\sigma(x)$ pour le terme t associé à x le plus à droite dans σ, ou bien $\mathtt{var}(x)$ s'il n'existe pas de tel temre. La définition formelle est donnée à la figure 6.1c.

Pour définir l'application d'une présubstitution en présence de lieurs, on peut essayer généraliser naïvement le cas du premier ordre. On va simplement traiter le cas du lieur générique $x.t$ en évitant de renommer les occurrences de x dans t. Il suffit pour cela d'appliquer la substitution $\sigma, x \backslash x$ sur t, au vu de notre traitement des variables.

$$\begin{aligned} \mathtt{var}(x)[\sigma]_{\mathsf{n}} &= \sigma(x) \\ \mathtt{op}[f](t_1, \ldots, t_n)[\sigma]_{\mathsf{n}} &= \mathtt{op}[f](t_1[\sigma]_{\mathsf{n}}, \ldots, t_n[\sigma]_{\mathsf{n}}) \\ (x.t)[\sigma]_{\mathsf{n}} &= y.t[\sigma, x \backslash x]_{\mathsf{n}} \end{aligned}$$

Quelles sont les propriétés de l'application naïve ? Tout d'abord, on peut montrer que la substitution identité $\mathtt{id}$ mérite son nom : l'appliquer laisse le terme inchangé.

Propriété 6.2.1. $x[x_1 \backslash x_1, \ldots, x_k \backslash x_k]_{\mathsf{n}} = x$.

Démonstration. On procède par induction sur l'entier k.

- Cas $k = 0$: on est dans le cas $x[\mathtt{id}]_{\mathsf{n}}$ qui est égal à x par définition.
- Cas $k = k' + 1$: alors soit $x = x_k$, et on conclut immédiatement car

$$x[x_1\backslash x_1, \ldots, x_k\backslash x_k]_{\mathsf{n}} = x_k = x,$$

soit $x \neq x_k$, auquel cas on a

$$x[x_1\backslash x_1, \ldots, x_{k'}\backslash x_{k'}, x_k\backslash x_k]_{\mathsf{n}} = x[\mathtt{id}, x_1\backslash x_1, \ldots, x_{k'}\backslash x_{k'}]_{\mathsf{n}} = x$$

où la deuxième égalité est l'hypothèse d'induction. □

Propriété 6.2.2. $t[x_1\backslash x_1, \ldots, x_k\backslash x_k]_{\mathsf{n}} = t.$

Démonstration. Par induction sur le terme t. Tous les cas sont des applications immédiates de la définition de l'application naïve de présubstitution et de l'hypothèse d'induction, à l'exception du cas des variables qui fait appel à la propriété 6.2.1. □

Corollaire 6.2.1. $t[\mathtt{id}]_{\mathsf{n}} = M.$

Le corollaire 6.2.1 exprime que la substitution identité agit sur les termes comme la fonction identité. Il est donc naturel de se demander s'il existe une opération sur les présubstitutions qui corresponde à la composition de leurs applications. Autrement dit, les présubstitutions σ_1 et σ_2 étant données, existe-t-il une présubstitution φ telle que pour tout préterme t, on ait $t[\varphi]_{\mathsf{n}} = t[\sigma_1]_{\mathsf{n}}[\sigma_2]_{\mathsf{n}}$? Le raisonnement ci-dessous, dû à Krivine [6], montre que la réponse est négative.

Soient x, y, z trois noms distincts, et posons $\sigma_1 = x\backslash y$ et $\sigma_2 = y\backslash x$. On a $y[x\backslash y]_{\mathsf{n}}[y\backslash x]_{\mathsf{n}} = x$ et $z[x\backslash y]_{\mathsf{n}}[y\backslash x]_{\mathsf{n}} = z$. Donc, si φ existe, elle doit être $y\backslash x$. Pourtant, on a

$$\begin{aligned}(y.x)[x\backslash y]_{\mathsf{n}}[y\backslash x]_{\mathsf{n}} &= (y.y)[y\backslash x]_{\mathsf{n}} = y.y \\ &\neq (y.x)[y\backslash x]_{\mathsf{n}} = y.x\end{aligned}$$

et donc φ ne peut pas jouer le rôle de la composition de σ_1 et σ_2.

Dans le raisonnement ci-dessus, le problème survient à cause de la *capture* de y dans $(y.x)[x\backslash y]$. Ce mot désigne le changement de statut de la variable y, qui apparaît *libre* dans la substitution $x\backslash y$ mais *liée* dans $(y.x)[x\backslash y]_{\mathsf{n}} = \mathtt{fun}(y.y)$. Informellement, une variable x apparaît libre dans un terme si elle y apparaît hors de la portée du lieur générique $x.(-)$.

Définition 6.2.3 (Variables libres et liées)**.** On définit l'ensemble $\mathsf{fv}(t)$ des variables libres d'un terme par récurrence sur t comme suit.

$$\mathsf{fv}(\mathtt{var}(x)) = \{x\} \qquad \mathsf{fv}(\mathtt{op}[f](t_1, \ldots, t_n) = \bigcup_i \mathsf{fv}(t_i) \qquad \mathsf{fv}(x.t) = \mathsf{fv}(t) \setminus \{x\}$$

L'ensemble $\mathsf{fv}(\sigma)$ (respectivement $\mathsf{dv}(\sigma)$) des variables *libres* (respectivement *définies*) d'une substitution σ est défini par récurrence comme suit.

$$\begin{aligned}\mathsf{fv}(\mathtt{id}) &= \emptyset & \mathsf{dv}(\mathtt{id}) &= \emptyset \\ \mathsf{fv}(\sigma, x\backslash t) &= \mathsf{fv}(\sigma) \cup \mathsf{fv}(t) & \mathsf{dv}(\sigma, x\backslash t) &= \mathsf{dv}(\sigma) \cup \{x\}\end{aligned}$$

Pour alléger les notations, on écrira $\nu(a_1, \ldots, a_n)$ pour désigner $\nu\left(\bigcup_i \mathsf{fv}(a_i)\right)$, où a_i désigne soit un préterme, soit une présubstitution. On écrira $x_1, \ldots, x_m = \nu(a_1, \ldots, a_n)$ pour désigner le choix de m variables libres successives distinctes et n'apparaissant dans aucun des a_i. Formellement, on a donc $x_j = \nu(a_1, \ldots, a_n, x_1, \ldots, x_{j-1})$ pour j compris entre 1 et m. On écrira $x_1, \ldots, x_m \# a_1, \ldots, a_n$ pour exprimer que $x_1, \ldots, x_m \notin \bigcup_i \mathsf{fv}(a_i)$ avec $x_1, \ldots, x_m$ des noms distincts. Plus généralement, on écrira $\sigma \# a_1, \ldots, a_n$ pour $\mathsf{dv}(\sigma) \# a_1, \ldots, a_n$.

Remarque 6.2.2. On ne parlera **jamais** des variables liées *au sein d'un terme*, pour éviter de distinguer des termes qui ne se distinguent que par celles-ci.

On peut maintenant définir la substitution de sorte à éviter la capture, ce qu'on appelle parfois *hygiène* dans la terminologie des langages de programmation. Pour ce faire, on renomme les variables liées en variables *fraîches* à la volée. Une variable fraîche est une variable n'apparaissant pas libre dans le terme et la substitution concernés. La définition de cette *substitution hygiènique* est donnée à la figure 6.1c. On définit aussi l'action d'une substitution σ sur une substitution φ, notée $\varphi[\sigma]$, qui consiste à faire agir σ sur chaque terme lié dans φ indépendamment.

Propriété 6.2.3. $\mathsf{fv}(t[\sigma]) \cup \mathsf{fv}(\sigma) = (\mathsf{fv}(t) - \mathsf{dv}(\sigma)) \cup \mathsf{fv}(\sigma)$.

Corollaire 6.2.2. $\mathsf{fv}(t[\sigma]) \subseteq (\mathsf{fv}(t) - \mathsf{dv}(\sigma)) \cup \mathsf{fv}(\sigma)$.

La propriété ci-dessus exprime l'effet qu'a l'application d'une substitution sur les variables libres d'un terme. En particulier, toutes les variables libres du terme après action de la substitution sont soit libres dans la substitution, soit libres dans le terme initial et non liées par la substitution, soit les deux.

6.2.2.2. Composition et produit de substitutions

On définit l'opération de composition de substitution conjecturée plus haut. Informellement, la composition $\sigma \circ \varphi$ contient les liaisons $x\backslash t$ de φ suivies de celles de σ où l'on a appliqué φ à chaque terme.

Définition 6.2.4 (Composition de substitution)**.** L'opération de *composition* associe à deux substitutions σ et φ leur *composée*, notée $\sigma \circ \varphi$ et définie par récurrence sur σ.

$$\mathtt{id} \circ \varphi = \varphi$$
$$(\sigma, x\backslash t) \circ \varphi = (\sigma \circ \varphi), x\backslash t[\varphi]$$

Propriété 6.2.4. $\mathsf{fv}(\sigma \circ \varphi) = (\mathsf{fv}(\sigma) - \mathsf{dv}(\varphi)) \cup \mathsf{fv}(\varphi)$ *et* $\mathsf{dv}(\sigma \circ \varphi) = \mathsf{dv}(\sigma) \cup \mathsf{dv}(\varphi)$.

Pour démontrer des propriétés plus intéressantes de la composée, on définit une autre opération sur les substitutions. Celle-ci, très simple, est une forme de concaténation.

Définition 6.2.5 (Produit de substitution)**.** L'opération de *multiplication* associe à deux substitutions σ et φ leur *produit*, noté σ, φ, défini par récurrence sur φ comme suit.

$$\sigma, \mathtt{id} = \sigma$$
$$\sigma, (\varphi, x\backslash t) = (\sigma, \varphi), x\backslash t$$

Il est clair que l'ensemble des variables libres (resp. définies) d'un produit de substitutions σ, φ est l'union des variables libres (resp. définies) des substitutions σ et φ. Le produit de substitutions est associatif et admet la substitution identique comme élément neutre à gauche et à droite. Il interagit avec la composition comme exprimé par les équations suivantes, qu'on utilisera souvent.

$$\sigma \circ \psi = \psi, \sigma[\psi] \tag{6.5}$$
$$(\sigma, \varphi)[\psi] = \sigma[\psi], \varphi[\psi] \tag{6.6}$$

6.2.3. Équivalence au renommage des variables liées près

On voudrait maintenant montrer que cette définition de la substitution remédie aux défauts de la précédente. Ce n'est pas le cas, du moins pas littéralement. Pire, on semble même avoir régressé : l'analogue du corollaire 6.2.1 échoue. Ainsi, en appliquant $(x.x)[\mathtt{id}]$ on obtient $\nu(\emptyset).\nu(\emptyset)$, mais le nom $\nu(\emptyset)$ n'a aucune raison d'être le même que x.

Pour remédier à ce défaut, il est naturel d'identifier les prétermes qui ne diffèrent que par l'identité de leurs variables liées. Le cas intéressant est celui des lieurs, où l'on doit exprimer que les variables liées n'importent pas. Par exemple, deux termes $x.t$ et $y.s$ doivent être considérés équivalents si $t[x\backslash z]$ et $s[y\backslash z]$ sont équivalents pour tous les noms z libres ni dans t ni dans s. Par exemple, sur la signature du λ-calcul, on ne veut pas identifier $\mathtt{fun}(x.\mathtt{app}(x, y))$ et $\mathtt{fun}(y.\mathtt{app}(y, x))$.

Pour rendre précis ces idées, on va être amenés à introduire des outils techniques, à savoir une classe de substitution particulière, ainsi que la notion de sous-ensemble cofini. On pourra ensuite donner une définition inductive de l'égalité modulo renommage des occurences liées des variables, appelée équivalence α.

6.2.3.1. Renommages

Un *renommage* est une substitution ρ telle que $\rho(x)$ soit de la forme $\mathtt{var}(-)$ pour tout x. Les renommages, qui seront notés $\rho, \rho_1, \rho_2, \rho'$ et ainsi de suite, constituent un ensemble de substitution qui contient l'identité et close par substitution. Il s'agit de substitutions particulièrement simples qui vérifient une propriété essentielle pour les démonstrations : faire agir un renommage sur un terme laisse sa taille inchangée.

Définition 6.2.6 (Taille d'un préterme)**.** La *taille* d'un préterme t, notée $\mathsf{sz}(t)$, est l'entier défini par récurrence sur t comme suit.

$$\mathsf{sz}(\mathtt{var}(x)) = 1 \qquad \mathsf{sz}(\mathtt{op}[f](t_1, \ldots, t_n)) = 1 + \sum_{i=1}^{n} \mathsf{sz}(t_i) \qquad \mathsf{sz}(x.t) = 1 + \mathsf{sz}(t)$$

La définition exacte du cas de base de sz n'est pas importante pour l'usage que l'on va en faire dans les arguments d'induction. L'essentiel est que la taille d'un terme soit strictement supérieure à celle de ses sous-termes. Ce point, souvent utilisé de pair avec le lemme suivant, permet de raisonner par par induction sur la taille.

Lemme 6.2.1. *Pour tout terme t et tout renommage ρ, on a $\mathsf{sz}(t[\rho]) = \mathsf{sz}(t)$.*

Démonstration. Soit t un terme. On va démontrer l'énoncé par induction sur la taille de t. L'hypothèse d'induction est donc que pour tout terme t' tel que $\mathsf{sz}(t') < \mathsf{sz}(t)$, pour tout renommage ρ, on a $\mathsf{sz}(t'[\rho]) = \mathsf{sz}(t')$. On raisonne par cas sur t.

- Cas $t = \mathtt{var}(x)$: puisque ρ est un renommage, $\rho(x)$ est une variable et donc de la même taille que t.
- Cas $t = \mathtt{op}[f](t'_1, \ldots, t'_n)$: on remarque tout d'abord que $\mathsf{sz}(t'_i) < \mathsf{sz}(t)$ pour tout i, et donc que l'hypothèse d'induction s'applique aux termes t'_i. On a

$$\mathsf{sz}(t[\rho]) = \mathsf{sz}(\mathtt{op}[f](t'_1[\rho], \ldots, t'_n[\rho])) = 1 + \sum_{i=1}^{n} \mathsf{sz}(t'_i[\rho]) = 1 + \sum_{i=1}^{n} \mathsf{sz}(t'_i) = \mathsf{sz}(t).$$

- Cas $t = x.t'$: on remarque tout d'abord que $\mathsf{sz}(t') < \mathsf{sz}(t)$, et donc que l'hypothèse d'induction s'applique au terme t'. Soit $y = \nu(t', \rho)$. On a

$$\mathsf{sz}(t[\rho]) = \mathsf{sz}(x.t'[\rho]) = \mathsf{sz}(y.t'[\rho, x\backslash y]) = 1 + \mathsf{sz}(t'[\rho, x\backslash y]) = 1 + \mathsf{sz}(t') = \mathsf{sz}(t)$$

  ce qui conclut la preuve. □

En plus de ce résultat sur la substitution, les équations suivantes, presque évidentes, permettent de manipuler efficacement.

Lemme 6.2.2. *Pour tous renommages ρ, ρ_1, ρ_2 et toute substitution σ, les équations*

$$\begin{aligned} \rho_1 \circ (\rho_2 \circ \sigma) &= (\rho_1 \circ \rho_2) \circ \sigma \\ \rho \circ \mathtt{id} &= \rho \\ \rho_1[\rho_2][\sigma] &= \rho_1[\rho_2 \circ \sigma] \\ \rho[\sigma] &= \rho \qquad\qquad \textit{si } \mathsf{fv}(\rho) \cap \mathsf{dv}(\sigma) = \emptyset \end{aligned}$$

sont validées.

6.2.3.2. Ensembles cofinis

Définition 6.2.7. Un sous-ensemble A d'un ensemble U est *cofini* si son complémentaire $U \setminus A$ est fini.

On note $A \subseteq_{\mathsf{cof}} U$ lorsque A est un sous-ensemble cofini de U. Par abus de langage, on dira qu'un *ensemble* A est cofini lorsque l'ensemble U dont A est une partie peut être déduit du contexte. Par exemple, un ensemble cofini d'entiers naturels est, formellement, un sous-ensemble cofini de $\mathbb{N}$. Dans ces circonstances, on notera $\overline{A}$ le complémentaire de A, c'est-à-dire $U \setminus A$. Tout ensemble A cofini peut par définition être écrit sous la forme $\overline{X}$ pour X un sous-ensemble fini de l'ensemble U auquel A appartient.

Exemple 6.2.3. Pour un entier naturel n fixé quelconque, l'ensemble des entiers supérieurs ou égaux à n est cofini. L'ensemble $\mathbb{N}$ tout entier est cofini, puisque c'est le complémentaire de l'ensemble vide d'entiers. En revanche, l'ensemble des entiers pairs n'est ni fini ni cofini, tout comme l'ensemble des entiers impairs.

Propriété 6.2.5. *Soient A et B deux parties d'un même ensemble U. Si A est cofinie, alors $A \cup B$ est cofini. Si A et B sont cofinies, alors $A \cap B$ est cofini. Si A est cofinie et B finie, alors $A \setminus B$ est cofinie.*

6.2.3.3. La relation d'équivalence

Définition 6.2.8. La relation d'*équivalence* α entre prétermes, notée $t \equiv_\alpha s$, est définie inductivement par les trois règles données ci-dessous.

$$\frac{}{x \equiv_\alpha x} \qquad \frac{t_1 \equiv_\alpha s_1 \quad \dots \quad t_n \equiv_\alpha s_n}{\mathtt{op}[f](t_1, \dots, t_n) \equiv_\alpha \mathtt{op}[f](s_1, \dots, s_n)} \qquad \frac{\exists F \subseteq_{\mathsf{cof}} \mathbb{A}, \forall z \in F, t[x\backslash z] \equiv_\alpha s[y\backslash z]}{x.t \equiv_\alpha y.s}$$

On écrira que t et u sont *α-équivalents* lorsque $t \equiv_\alpha u$. Deux termes sont α-équivalents s'ils sont identiques modulo un renommage de leurs variables liées. Intuitivement, la quantification cofinie utilisée pour traiter le cas du lieur générique exprime que $x.t$ et $y.s$ sont α-équivalents lorsque t et s sont identiques si on renomme x et y par *presque tous* les noms z, c'est-à-dire pour tous les noms z sauf un nombre fini d'entre eux.

Remarque 6.2.3. Il n'est pas évident que l'équivalence α, définie comme nous l'avons fait, soit une relation décidable. C'est bien le cas ; trouver une formulation algorithmique est le sujet de l'exercice 30.

L'équivalence α s'étend aux substitutions de façon naturelle. Deux substitutions σ et φ sont α-équivalentes lorsqu'elles définissent le même ensemble de variables et que, pour tout x appartenant à cet ensemble, on a $\sigma(x) \equiv_\alpha \varphi(x)$. Notons que deux renommages sont α-équivalents si et seulement s'ils sont égaux.

Théorème 6.2.1. *L'équivalence α est une relation d'équivalence.*

Démonstration. La symétrie de la relation est évidente par inspection des règles. On va démontrer la réflexivité et la transitivité en détaillant uniquement le cas du lieur générique, les autres étant immédiats.

On démontre d'abord la réflexivité, c'est-à-dire que pour tout terme t, on a $t \equiv_\alpha t$. On raisonne par induction sur la taille de t, puis par cas sur t. Le cas où t est une variable est immédiat. Celui où t est une application d'un opérateur de la signature est obtenu par application immédiate de l'hypothèse d'induction. Il reste celui où t est de la forme $x.t'$. On choisit l'ensemble $\mathbb{A}$ cofini tout entier pour F. On doit donc montrer que pour tout z, on a $t'[x\backslash z] \equiv_\alpha t'[x\backslash z]$. Par le lemme 6.2.1, on a que $t'[x\backslash z] = \mathsf{sz}(t') < \mathsf{sz}(t)$, et on conclut immédiatement par hypothèse d'induction.

On démontre maintenant la transitivité. On doit montrer que pour tous termes u, t, v, si $t \equiv_\alpha u$ et $u \equiv_\beta v$ alors $t \equiv_\alpha v$. On raisonne par induction sur la taille de u puis par cas. Encore une fois, les cas des variables et de l'application d'un opérateur de la signature sont routiniers. Considérons le cas $u = z.u'$. Par inversion des hypothèses, on doit avoir $t = x.t'$ et $v = y.v'$ ainsi qu'un ensemble cofini de noms F_1 tel que pour tout k_1 dans F_1 on ait $t'[x\backslash k_1] \equiv_\alpha u'[z\backslash k_1]$, et un ensemble cofini de noms F_2 tel que pour tout k_2 dans F_2 on ait $v'[y\backslash k_1] \equiv_\alpha u'[z\backslash k_2]$. On doit montrer que $x.t' \equiv_\alpha y.v'$, c'est-à-dire choisir un ensemble cofini F tel que pour tout k dans F on ait $t'[x\backslash k] \equiv_\alpha u'[y\backslash k]$. L'intersection

de deux ensembles cofinis est cofinie, et on choisit $F_1 \cap F_2$ pour F. Tout k dans F est à la fois dans F_1 et dans F_2, et on a donc $t'[x\backslash k] \equiv_\alpha u'[z\backslash k]$ et $u'[z\backslash k] \equiv_\alpha t'[y\backslash k]$. Par le lemme 6.2.1, on a $\mathsf{sz}(u'[z\backslash k]) = \mathsf{sz}(u') < \mathsf{sz}(u)$, et on conclut immédiatement par application de l'hypothèse d'induction. □

Le théorème 6.2.1 assure l'existence de l'ensemble des *termes*, défini comme quotient des prétermes par l'équivalence α.

Définition 6.2.9. L'ensemble $\mathcal{T}[\Sigma]$ des *termes* sur la signature du second ordre Σ est défini comme le quotient des prétermes par l'équivalence α, de même pour l'ensemble $\mathcal{S}[\Sigma]$ des *substitutions*.

Pour que la notion de terme et de substitution soit utile, il faut que toutes les opérations définies précédemment sur les prétermes passent au quotient. On montre par une induction de routine que c'est le cas de la taille.

Propriété 6.2.6. *Deux prétermes α-équivalents ont la même taille.*

On va utiliser cette propriété pour montrer que l'application d'un renommage passe au quotient. La preuve est plus laborieuse, et nécessite la démonstration simultanée d'un certain nombre de propriétés qui lient renommage et α-équivalence.

Lemme 6.2.3 (Gestion des renommages)**.** *Soit t un préterme.*

1. *Pour tout préterme u qui est α-équivalent à t :*
   a) *pour tout renommage ρ, les prétermes $t[\rho]$ et $u[\rho]$ sont α-équivalents ;*
   b) *pour toute variable x, le préterme $x.t$ est α-équivalent au préterme $x.u$.*
2. *Les prétermes t et $t[\mathtt{id}]$ sont α-équivalents.*
3. *Pour tous renommages ρ_1 et ρ_2, les prétermes $t[\rho_1][\rho_2]$ et $t[\rho_1 \circ \rho_2]$ sont α-équivalents.*
4. *Pour tous renommages ρ_1, ρ_2 et ρ_3 tels qu'aucune variable définie par ρ_2 n'est libre dans t, les prétermes $t[\rho_1, \rho_2, \rho_3]$ et $t[\rho_1, \rho_3]$ sont α-équivalents.*

Démonstration. On démontre simultanément par induction sur la taille de t tous les énoncés qui forment le lemme, puisque la preuve de chacun fait intervenir les autres. La preuve de chaque énoncé va utiliser la propriété de *changement de variable* qui suit, essentielle pour manipuler efficacement les lieurs génériques.

Soit t' un préterme de taille strictement inférieure à celle de t et ρ un renommage. Pour tout nom x et tout nom y tel que y n'est ni libre dans t, ni libre ni défini dans ρ, les prétermes $(x.t')[\rho]$ et $y.t'[x\backslash y][\rho]$ sont α-équivalents.

On doit donc montrer que les prétermes $x'.t'[\rho, x\backslash x']$ et $y.t'[x\backslash y][\rho]$ sont α-équivalents, où x' désigne $\nu(t', \rho)$. Pour tout z libre ni dans t', ni libre ni défini dans ρ, on a

$$\begin{aligned} t'[\rho, x\backslash x'][x'\backslash z] &\equiv_\alpha t'[(\rho, x\backslash x') \circ (x'\backslash z)] && \text{(hyp. d'induction, énoncé 3)} \\ &= t'[x'\backslash z, \rho[x'\backslash z], x\backslash z] \\ &\equiv_\alpha t'[\rho, x\backslash z] && \text{(hyp. d'induction, énoncé 4, } x' \# t', \rho) \end{aligned}$$

et

$$\begin{aligned}
t'[x\backslash y][\rho][y\backslash z] &\equiv_\alpha t'[(x\backslash y)\circ\rho\circ(y\backslash z)] && \text{(hyp. d'induction, énoncé 3)}\\
&= t'[(\rho, x\backslash y)\circ(y\backslash z)] && (y\notin \mathsf{dv}(\rho))\\
&= t'[y\backslash z, \rho[y\backslash z], x\backslash z]\\
&\equiv_\alpha t'[\rho, x\backslash z] && \text{(hyp. d'induction, énoncé 4)}
\end{aligned}$$

ce qui conclut la preuve de la propriété.

On prouve maintenant les énoncés 1 à 4. Chaque preuve consiste en un raisonnement par cas sur la forme de t, puis en l'utilisation judicieuse du résultat de changement de variable prouvé ci-dessus et des hypothèses d'induction. Celles-ci sont appliquées pour raisonner sur des prétermes plus petits que t, ce qui sera la conséquence du lemme 6.2.1.

1. Soit u un préterme α-équivalent à t.

   a) On raisonne par cas sur t, puis par inversion sur $t \equiv_\alpha u$. Le cas $t = u = \mathtt{var}(x)$ est immédiat. Le cas $t = \mathtt{op}[f](t'_1, \ldots, t'_n)$ est traité par un usage routinier de l'hypothèse d'induction. Reste le cas $t = x.t'$ et $u = y.u'$. On dispose d'un certain ensemble cofini F tel que pour tout nom k dans F les prétermes $t'[x\backslash k]$ et $u'[y\backslash k]$ sont α-équivalents. De plus, on sait que t' et u' ont la même taille par la propriété 6.2.6, les hypothèses d'induction s'appliquent donc à u'. Soit z un nom appartenant à F qui ne soit libre ni dans t', ni dans u', ni libre ni défini dans ρ. On a

   $$\begin{aligned}
   (x.t')[\rho] &\equiv_\alpha z.t'[x\backslash z][\rho] && \text{(changement de variable)}\\
   &\equiv_\alpha z.u'[y\backslash z][\rho] && \text{(hyp. d'induction, énoncés 1a et 1b)}\\
   &\equiv_\alpha (y.u')[\rho] && \text{(changement de variable)}
   \end{aligned}$$

   ce qui conclut la preuve de cet énoncé.

   b) Pour montrer $x.t \equiv_\alpha x.u$, il suffit de montrer $t[x\backslash z]$ et $u[x\backslash z]$ pour tout nom z, ce qui est une conséquence immédiate de la propriété précédente.

2. On raisonne par cas sur t. Les cas d'une variable et de l'application d'un opérateur sont routiniers. Reste le cas $t = x.t'$. Soit z un nom non libre dans t'. On a

   $$\begin{aligned}
   (x.t')[\mathtt{id}] &\equiv_\alpha z.t'[x\backslash z][\mathtt{id}] && \text{(changement de variable)}\\
   &\equiv_\alpha z.t'[x\backslash z] && \text{(hyp. d'induction, énoncé 2)}
   \end{aligned}$$

   qui est clairement α-équivalent à $x.t'$.

3. Soit ρ_1, ρ_2 des renommages. On raisonne par cas sur t. Le cas $t = \mathtt{var}(x)$ est traité par une induction de routine sur ρ_1. Le cas $t = \mathtt{op}[f](t'_1, \ldots, t'_n)$ est traité par application immédiate des hypothèses d'induction. Reste le cas $t = x.t'$. Soit z un

nom qui n'est pas libre dans t', ni libre ni défini dans ρ_1 et ρ_2. On a

$$\begin{aligned}
(x.t')[\rho_1][\rho_2] &\equiv_\alpha (z.t'[x\backslash z][\rho_1])[\rho_2] && \text{(changement de variable)}\\
&\equiv_\alpha z.t'[x\backslash z][\rho_1][z\backslash z][\rho_2] && \text{(changement de variable)}\\
&\equiv_\alpha z.t'[(x\backslash z)\circ\rho_1\circ(z\backslash z)\circ\rho_2] && \text{(hyp. d'induction, énoncé 3)}\\
&= z.t'[\rho_2, z\backslash z, \rho_1[z\backslash z][\rho_2], x\backslash z] && (z\notin \mathsf{dv}(\rho_1,\rho_2))\\
&\equiv_\alpha z.t'[\rho_2, \rho_1[\rho_2], x\backslash z] && \text{(hyp. d'induction, énoncé 4)}\\
&= z.t'[(x\backslash z)\circ\rho_1\circ\rho_2] && (z\notin \mathsf{dv}(\rho_1,\rho_2))\\
&= z.t'[x\backslash z][\rho_1\circ\rho_2] && \text{(hyp. d'induction, énoncé 3)}\\
&\equiv_\alpha (x.t')[\rho_1\circ\rho_2]. && \text{(changement de variable)}
\end{aligned}$$

ce qui conclut la preuve de cet énoncé.

4. Soient ρ_1, ρ_2, ρ_3 des renommages tels qu'aucune variable définie par ρ_2 n'apparaisse libre dans t. On raisonne par cas sur t. Le cas $t = \mathtt{var}(x)$ est traité par une induction de routine sur ρ_2 puis sur ρ_3. Encore une fois, le cas $t = \mathtt{op}[f](t'_1,\dots,t'_n)$ est traité par application immédiate des hypothèses d'induction. Reste le cas $t = x.t'$. Soit z qui ne soit ni libre dans t', ni libre ni défini dans ρ_1, ρ_2 ou ρ_3. On a

$$\begin{aligned}
(x.t')[\rho_1,\rho_2,\rho_3] &= z.t'[x\backslash z][\rho_1,\rho_2,\rho_3] && \text{(changement de variable)}\\
&= z.t'[x\backslash z][\rho_1,\rho_3] && \text{(hyp. d'induction, énoncé 4 avec } \rho_2 \# t,z)\\
&= (x.t')[\rho_1,\rho_3] && \text{(changement de variable)}
\end{aligned}$$

ce qui conclut la preuve de cet énoncé. □

Il nous faudrait maintenant revisiter les résultats précédents, en particulier le lemme 6.2.3, pour généraliser des renommages aux substitutions arbitraires. C'est un exercice fastidieux qui est laissé au lecteur ; on se contente d'énoncer les propriétés attendues.

Lemme 6.2.4 (Gestion des substitutions)**.** *Soit t un préterme.*

1. *Pour tout préterme u qui est α-équivalent à t, pour toute substitution σ, les prétermes $t[\sigma]$ et $u[\sigma]$ sont α-équivalents.*

2. *Pour toute substitution σ et tous noms x, y tels que y n'est ni libre dans t ni libre ni défini dans*

3. *Pour toutes substitutions σ et φ, les prétermes $t[\sigma][\varphi]$ et $t[\sigma\circ\varphi]$ sont α-équivalents.*

Démonstration. □

6.2.4. La gestion des variables liées en pratique

Les résultats précédents permettent de se rendre compte de la difficulté qu'il peut y avoir à traiter de façon à la fois rigoureuse et élémentaire la gestion des variables libres

et liées. Ce problème est fréquemment passé sous silence dans les cours de mathématiques qui introduisent la notion de variable liée. Pourtant, des solutions satisfaisantes sont disponibles, et elles sont satisfaisantes quoique pas exemptes de défauts. On peut par exemple adopter une syntaxe moins lisible mais plus disciplinée (voir par exemple l'exercice 36), ou bien se placer dans un cadre mathématique où la notion de variable fraîche reçoit un statut à part entière, comme les ensembles nominaux [7].

Dans les faits, lorsqu'on travaille sur papier plutôt que sur machine, on adopte la convention dite *de Barendregt*, du nom de l'auteur de l'ouvrage de référence sur le λ-calcul [1]. L'idée est la suivante.

> Au moment de manipuler un préterme t dans une définition ou une preuve, on s'assure par un renommage approprié de choisir un représentant de la classe d'équivalence de t dont toutes les variables liées sont distinctes des variables libres utilisées à ce point du texte.

Cette convention est difficile à rendre mathématiquement précise puisque la notion de variables libres "utilisées à ce point du texte" est informelle, sauf à introduire (beaucoup) d'outillage mathématique. Cependant, elle permet dans les faits de ne plus jamais avoir à se soucier de problèmes de capture, et en particulier de *prétendre que la substitution naïve coïncide avec la substitution hygiénique*. L'usage a montré qu'elle n'était pas source d'erreur, du moins tant que l'on traite de langages où la structure de la liaison est suffisamment simple. Ce sera le cas de tous les langages traités dans ces notes, et pour cette raison on adopte à partir de ce point la convention de Barendregt sans rechigner.

6.2.5. Exercices

* **Ex. 27** — Démontrer la propriété 6.2.5.

* **Ex. 28** — Dans cet exercice, on considère les prétermes sur la signature Σ_λ du λ-calcul.

1. Démontrer que la propriété $\mathsf{fv}(t[\sigma]) = (\mathsf{fv}(t) - \mathsf{dv}(\sigma)) \cup \mathsf{fv}(\sigma)$ est fausse.
2. Démontrer que la propriété $\mathsf{sz}(t[\sigma]) = \mathsf{sz}(t)$ est fausse.

* **Ex. 29** — On considère la signature $\Sigma \coloneqq \Sigma_\lambda \cup \{\mathtt{zero} : \mathbb{T}, \mathtt{suc} : \mathbb{T} \to \mathbb{T}\}$. Grouper les Σ-prétermes suivants par classe d'α-équivalence.

$\mathtt{fun}(x.x)$ $\quad$ $\mathtt{fun}(x.\mathtt{suc}(\mathtt{app}(x,y)))$ $\quad$ $\mathtt{fun}(z.\mathtt{suc}(\mathtt{app}(z,y)))$ $\quad$ $\mathtt{fun}(y.y)$ $\quad$ $\mathtt{fun}(z.x)$

$\mathtt{fun}(y.\mathtt{zero})$ $\quad$ $\mathtt{fun}(y.\mathtt{suc}(\mathtt{app}(y,y)))$ $\quad$ $\mathtt{fun}(x.\mathtt{zero})$

*** **Ex. 30** — Soit Σ une signature du second ordre. L'*équivalence* ν est la plus petite relation sur les Σ-prétermes close par les règles suivantes.

$$\frac{}{x \equiv_\nu x} \qquad \frac{t_1 \equiv_\nu s_1 \quad \dots \quad t_n \equiv_\nu s_n}{\mathtt{op}[f](t_1,\dots,t_n) \equiv_\nu \mathtt{op}[f](s_1,\dots,s_n)} \qquad \frac{z = \nu(t,s) \qquad t[x\backslash z] \equiv_\nu s[y\backslash z]}{x.t \equiv_\nu y.s}$$

1. Montrer que l'équivalence ν est décidable en implémentant une fonction

   ```
   eq : soterm -> soterm -> bool
   ```

   telle que `eq s t` s'évalue vers `true` si et seulement si `s` et `t` sont ν-équivalents. Le type `soterm` désigne les prétermes du second ordre sur une signature quelconque, et est défini comme suit.

   ```
   type op = string and var = string
   type soterm = Var of var | Op of op * soterm list | Bind of var * soterm
   ```

2. Montrer que l'équivalence ν et l'équivalence α coïncident.

6.3. Manipulations relationnelles

Dans ce qui suit, on fixe une signature du second ordre Σ et une endorelation R de $\mathcal{T}[\Sigma]$. On définit rapidement les analogues de second ordre des relations vues au chapitre 5.

Définition 6.3.1. L'*extension* Σ-*compatible* de R est la relation

$$\begin{aligned}\Sigma(R) \coloneqq &\{(\mathtt{var}(x), \mathtt{var}(x)) \mid x \in \mathbb{A}\} \\ &\cup \bigcup_{f \in |\Sigma|} \{(\mathtt{op}[f](t_1, \ldots, t_{\mathsf{ar}(f)}), \mathtt{op}[f](t'_1, \ldots, t'_{\mathsf{ar}(f)})) \mid \forall i, (t_i, t'_i) \in R\} \\ &\cup \{(x.t, x.t') \mid x \in \mathbb{A}, (t, t') \in R\}.\end{aligned}$$

Une relation est Σ-*compatible* lorsqu'elle contient son extension Σ-compatible. Une Σ-*congruence* est une relation d'équivalence Σ-compatible.

Propriété 6.3.1. *Si R est une Σ-congruence et que la paire (t, s) appartient à R alors la paire $(u[x\backslash t], u[x\backslash s])$ appartient à R pour tout u.*

Définition 6.3.2. L'*extension* Σ-*contextuelle* de R est la relation

$$\begin{aligned}\Sigma^{\square}(R) \coloneqq &\bigcup_{f \in |\Sigma|} \bigcup_{1 \le i \le \mathsf{ar}(f)} \left\{ \begin{array}{l} (\mathtt{op}[f](t_1, \ldots, t_i, \ldots, t_{\mathsf{ar}(f)}), \mathtt{op}[f](t_1, \ldots, t'_i, \ldots, t_{\mathsf{ar}(f)}) \\ \mid (t_i, t'_i) \in R \end{array} \right\} \\ &\cup \{(x.t, x.t') \mid x \in \mathbb{A}, (t, t') \in R\}.\end{aligned}$$

Une relation est Σ-*contextuelle* lorsqu'elle contient son extension Σ-contextuelle.

Propriété 6.3.2. *Une relation réflexive et Σ-compatible est Σ-contextuelle.*

Troisième partie

Langages simplement typés

7. Le système T

Ce chapitre discute du système T, formalisme calculatoire qui peut être vu comme un langage de programmation rudimentaire où l'on dispose de fonctions et de la récursion primitive sur les entiers naturels. Ce système a été proposé par Kurt Gödel [4] dans le cadre de travaux sur les fondations des mathématiques. On en donne une présentation modernisée due à William Tait [9].

7.1. Syntaxe pure

La syntaxe pure du système T est donnée sur la signature Σ_{T} définie par

$$\begin{aligned}\Sigma_{\mathrm{T}} \coloneqq \{&\mathtt{app} : \mathbb{T} \times \mathbb{T} \to \mathbb{T}, \mathtt{fun} : (\mathbb{V} \to \mathbb{T}) \to \mathbb{T}, \mathtt{zero} : \mathbb{T},\\ &\mathtt{suc} : \mathbb{T} \to \mathbb{T}, \mathtt{fold}_{\mathtt{Nat}} : \mathbb{T} \to \mathbb{T} \to (\mathbb{V} \to \mathbb{V} \to \mathbb{T}) \to \mathbb{T}\}.\end{aligned}$$

Par la technologie développée au chapitre 6, cette syntaxe hérite automatiquement de l'opération de substitutions. On ne considère que les termes sur cette signature, c'est-à-dire modulo équivalence α. On suit la convention de Barendregt, comme annoncé.

Il est utile de donner une définition explicite des termes du second-ordre sur cette signature. Celle-ci correspond à la grammaire définie à la figure 7.1. De plus, on adopte quelques raccourcis de notation pour faciliter la lecture. Tout d'abord, on écrira généralement x plutôt que $\mathtt{var}(x)$ lorsqu'aucune confusion ne peut en résulter. Ensuite, on écrira $t\, s_1\, s_2 \ldots s_k$ pour $\mathtt{app}(\ldots\ (\mathtt{app}(\mathtt{app}(t, s_1), s_2), \ldots), s_k)$. Enfin, on abrègera en $\underline{n}$ le préterme $\mathtt{suc}^n(\mathtt{zero})$ contenant n successeurs imbriqués.

$\mathcal{T}[\Sigma_{\mathrm{T}}] \ni t, u, v ::=$	**Termes de** T
$\mid \mathtt{var}(x)$	Variable
$\mid \mathtt{fun}(x.t)$	Abstraction
$\mid \mathtt{app}(t, u)$	Application
$\mid \mathtt{zero}$	Zéro
$\mid \mathtt{suc}(t)$	Successeur
$\mid \mathtt{fold}_{\mathtt{Nat}}(t, u, x.y.v)$	Récursion primitive

FIG. 7.1. : Syntaxe de T

7.2. Types et calcul

7.2.1. Équivalence β

La relation d'équivalence α décrite précédemment exprime une forme très simple d'égalité de programmes : deux programmes qui ne diffèrent que par le nom de leurs variables liées sont essentiellement les mêmes. On va maintenant introduire une notion d'égalité plus dynamique, au sens où elle caractérise les programmes qui « calculent le même résultat ».

Définition 7.2.1. La *β-réduction atomique* est l'endorelation de $\mathcal{T}[\Sigma_{\mathrm{T}}]$ notée $t \rightsquigarrow_\beta t'$ et définie par les trois clauses listées ci-dessous.

$$\mathtt{app}(\mathtt{fun}(x.t), u) \rightsquigarrow_\beta t[x \backslash u] \tag{7.1}$$

$$\mathtt{fold}_{\mathtt{Nat}}(\mathtt{zero}, u, x.y.v) \rightsquigarrow_\beta v \tag{7.2}$$

$$\mathtt{fold}_{\mathtt{Nat}}(\mathtt{suc}(t), u, x.y.v) \rightsquigarrow_\beta v[x \backslash t, y \backslash \mathtt{fold}_{\mathtt{Nat}}(t, u, x.y.v)] \tag{7.3}$$

Définition 7.2.2 (β-équivalence)**.** La relation de *β-équivalence*, notée $t \equiv_\beta u$, est la plus petite T-congruence contenant la β-réduction atomique.

L'équivalence β est un objet d'étude fondamental de la théorie des langages de programmation. Pour expliquer ces relations, il est utile de distinguer dans la syntaxe de T deux familles d'opérations distinctes.

- Les *formes d'introduction* permettent de créer de nouvelles données. Elles comprennent l'abstraction, le zéro et le successeur.
- Les *formes d'élimination* permettent d'utiliser des données existantes. Elles comprennent l'application et la récursion primitive.

Les clauses 7.1 à 7.3 expriment qu'une forme d'introduction imbriquée sous une forme d'élimination est équivalente à un terme plus simple. Par exemple, l'application d'une abstraction à un argument est équivalente au corps de l'abstraction où le paramètre a été substitué par l'argument, comme exprimé par l'équation (7.1).

Remarque 7.2.1. La variable n'est ni une forme d'introduction, ni une forme d'élimination. La variable réalise une opération qu'on pourrait qualifier de *logistique*, au sens où elle n'agit pas par elle même mais fait fonctionner le reste du langage en interagissant avec la substitution.

7.2.2. Types

La syntaxe de T telle que nous l'avons définie n'impose que peu de contrainte sur l'imbrication des opérations. En particulier, une forme d'élimination peut rencontrer une forme d'introduction incohérente, au sens où aucune des équations (7.1) à (7.3) ne s'applique. C'est le cas par exemple d'un terme de la forme $\mathtt{app}(\mathtt{zero}, u)$ ou encore $\mathtt{fold}_{\mathtt{Nat}}(\mathtt{fun}(z.M), u, x.y.v)$. Il est raisonnable de considérer de tels termes comme dépourvus de sens.

Pour ne pas avoir à se soucier de tels termes, on va raffiner la syntaxe de T en classifiant les termes en fonction de la nature de leur résultat. Le lecteur ou la lectrice aura sans doute reconnu la notion de *système de types*, essentielle à de nombreux langages de programmation. L'idée est d'associer aux termes t des *types* A, auxquels on peut penser comme à des formules logiques qui classifient le résultat construit par t. Les types de T sont très simples : on ne peut y construire que des entiers, de type $\mathtt{Nat}$, et des fonctions, de type $A \to B$, où A et B sont des types. On écrira que t *habite le type* A, et on notera $M : A$, lorsque A est un type possible pour t.

La relation $t : A$ est définie inductivement par un ensemble de règles. Essayons d'imaginer à quoi celles-ci peuvent ressembler. Les règles traitant du zéro, du successeur et de l'application semblent s'écrire d'elles-même.

$$\frac{}{\mathtt{zero} : \mathtt{Nat}} \qquad \frac{t : \mathtt{Nat}}{\mathtt{suc}(t) : \mathtt{Nat}} \qquad \frac{t : A \to B \qquad s : A}{\mathtt{app}(t, s) : B}$$

Les cas de l'abstraction et de la récursion primitive sont nettement moins évidents.

$$\frac{t : B \ ?!}{\mathtt{fun}(x.t) : A \to B} \qquad \frac{t : \mathtt{Nat} \qquad s : A \qquad u : A \ ?!}{\mathtt{fold}_{\mathtt{Nat}}(t, s, x.y.u)}$$

Considérons le cas de l'abstraction, celui de la récursion primitive étant similaire. Une abstraction $\mathtt{fun}(x.t)$ est une fonction et doit donc nécessairement habiter un type de la forme $A \to B$. Pour montrer que c'est le cas, il faut pouvoir montrer que son corps t habite le type B. Mais ce n'est pas tout : les occurrences à x dans t doivent être considérées comme habitant le type A. Or, notre jugement $t : A$ est incapable d'exprimer cette hypothèse. Symétriquement, on ne voit pas non plus comment traiter les variables.

$$\frac{}{x : ?!}$$

Pour régler ce problème, on est naturellement amenés à considérer un jugement de typage dit *hypothétique* : un terme t habite le type A sous l'hypothèse que ses variables libres $x_1, \ldots, x_k$ habitent les types $B_1, \ldots, B_k$, ce qu'on écrit $x_1 : B_1, \ldots, x_k : B_k$. On appelle cette liste d'hypothèses un *contexte de typage*, et on écrira

$$\Gamma \vdash t : A$$

pour exprimer que t habite le type A *sous le contexte* Γ, contexte qui prescrit les types habités par les variables libres de t. Les contextes sont soumis à une restriction importante : une même variable apparaît **au plus** une fois dans un contexte de typage donné. Ainsi, $x : \mathtt{Nat}, y : \mathtt{Nat} \to \mathtt{Nat}, x : \mathtt{Nat}$ n'est pas un contexte valide.

Définition 7.2.3 (Types de T)**.** La syntaxe des types et des contextes de T est définie à la figure 7.2.

Définition 7.2.4 (Affaiblissement)**.** On dit qu'un contexte Δ peut être *affaibli* en un contexte Γ, ce qu'on note $\Delta \supseteq \Gamma$, lorsqu'on peut obtenir Γ en effaçant un nombre fini de liaisons dans Δ. Formellement, il s'agit du jugement inductif défini à la figure 7.3.

$\mathsf{Type}_{\mathrm{T}} \ni A, B, C ::=$		**Types**
	$\mid \mathtt{Nat}$	Type des entiers naturels
	$\mid A \to B$	Type fonctionnel
$\mathsf{Con}_{\mathrm{T}} \ni \Gamma, \Delta ::=$		**Contextes**
	$\mid \cdot$	Contexte vide
	$\mid \Gamma, x : A$	Déclaration de variable

FIG. 7.2. : Types de T

$$\boxed{\Delta \supseteq \Gamma} \qquad \frac{}{\Delta \supseteq \cdot} \qquad \frac{\Delta \supseteq \Gamma}{\Delta, x : A \supseteq \Gamma} \qquad \frac{\Delta \supseteq \Gamma}{\Delta, x : A \supseteq \Gamma, x : A}$$

FIG. 7.3. : Affaiblissement des contextes de T

Propriété 7.2.1. *Le jugement d'affaiblissement est réflexif et transitif.*

Démonstration. Par des inductions de routine. □

Définition 7.2.5 (Typage des termes de T)**.** Le typage des termes est défini par les jugements inductifs présentés à la figure 7.4. On écrit $\mathcal{T}[\Sigma_{\mathrm{T}}](\Gamma; A)$ pour l'ensemble des termes habitant le type A sous le contexte Γ, et on abrège $\cdot \vdash t : A$ en $t : A$.

La règle traitant du cas de la variable utilise le jugement auxiliaire dit d'*extension*, noté $\Delta \supseteq \Gamma$, qui exprime que le contexte Γ peut être obtenu à partir de Δ en oubliant certaines variables.

Il faut insister encore sur le fait que toute variable apparaît au plus une fois dans un contexte. C'est une supposition implicite des deux règles mettant en jeu la liaison,

$$\boxed{\Gamma \vdash M : A} \qquad \frac{\Gamma \supseteq x : A}{\Gamma \vdash \mathtt{var}(x) : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \mathtt{fun}(x.t) : A \to B}$$

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash \mathtt{app}(t, s) : B} \qquad \frac{}{\Gamma \vdash \mathtt{zero} : \mathtt{Nat}} \qquad \frac{\Gamma \vdash t : \mathtt{Nat}}{\Gamma \vdash \mathtt{suc}(t) : \mathtt{Nat}}$$

$$\frac{\Gamma \vdash t : \mathtt{Nat} \qquad \Gamma \vdash s : A \qquad \Gamma, x : \mathtt{Nat}, y : A \vdash u : A}{\Gamma \vdash \mathtt{fold}_{\mathtt{Nat}}(t, s, x.y.u) : A}$$

FIG. 7.4. : Typage des termes de T

$$\boxed{\Delta \vdash \sigma : \Gamma} \qquad \frac{\Delta \supseteq \Gamma}{\Delta \vdash \mathtt{id} : \Gamma} \qquad \frac{\Delta \vdash \sigma : \Gamma \qquad \Delta \vdash t : A}{\Delta \vdash \sigma, x\backslash t : \Gamma, x : A}$$

FIG. 7.5. : Typage des substitutions de T

c'est-à-dire l'abstraction et la récursion primitive. Ainsi, la prémisse de la règle

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \mathtt{fun}(x.t) : A \to B}$$

présuppose que x n'apparaît pas dans Γ. Cependant, puisque $\mathtt{fun}(x.t)$ est un terme et donc considéré modulo équivalence α, on peut toujours renommer x en une variable qui n'apparaît pas dans Γ.

En plus de typer les termes, il est également commode de disposer d'un jugement de typage classifiant les substitutions. Une substitution σ est formée de plusieurs termes, nommés qui plus est, il est donc naturel de la classifier par un contexte de typage.

Définition 7.2.6 (Typages des substitutions de T)**.** Le jugement de typage des substitutions est défini à la figure 7.5. On écrit $\mathcal{T}[\Sigma_{\mathrm{T}}](\Delta;\Gamma)$ pour l'ensemble des substitutions habitant le contexte Γ sous le contexte Δ.

On démontre maintenant un lemme fondamental, caractéristique des systèmes de types, qui exprime la compatibilité entre le typage et la substitution.

Lemme 7.2.1 (Affaiblissement)**.** *Les jugements de typage des termes et des substitutions sont stables par extension. Autrement dit, pour toute extension Δ d'un contexte Γ, si $\Gamma \vdash t : A$ alors $\Delta \vdash t : A$, et si $\Gamma \vdash \sigma : \Theta$ alors $\Delta \vdash \sigma : \Theta$.*

Théorème 7.2.1 (Substitution)**.** *Si $\Gamma \vdash t : A$ et $\Delta \vdash \sigma : \Gamma$ alors $\Delta \vdash t[\sigma] : A$.*

Démonstration. Par induction sur la dérivation de typage $\Gamma \vdash t : A$. On ne détaille que les trois premiers cas.

- Cas $\dfrac{\Gamma \supseteq x : A}{\Gamma \vdash x : A}$: par une nouvelle induction sur le jugement d'extension.
- Cas $\dfrac{\Gamma \vdash t_1 : A \to B \qquad \Gamma \vdash t_2 : A}{\Gamma \vdash \mathtt{app}(t_1, t_2) : B}$: on a $\mathtt{app}(t_1, t_2)[\sigma] = \mathtt{app}(t_1[\sigma], t_2[\sigma])$ et donc

$$\frac{\Delta \vdash t_1[\sigma] : A \to B \qquad \Delta \vdash t_2[\sigma] : A}{\Delta \vdash \mathtt{app}(t_1[\sigma], t_2[\sigma]) : B}$$

  où les deux prémisses sont les hypothèses d'induction.

- Cas $\dfrac{\Gamma, x : A \vdash t_1 : B}{\Gamma \vdash \texttt{fun}(x.t') : A \to B}$: on a $\texttt{fun}(x.t')[\sigma] = \texttt{fun}(x.t'[\sigma, x\backslash x])$ et donc

$$\frac{\Delta, x : A \vdash t'[\sigma, x\backslash x] : B}{\Delta \vdash \texttt{fun}(x.t'[\sigma, x\backslash x]) : B}$$

où la prémisse est obtenue par l'hypothèse d'induction, appliquée à

$$\frac{\Delta, x : A \vdash \sigma : \Gamma \qquad \Delta, x : A \vdash x : A}{\Delta, x : A \vdash (\sigma, x\backslash x) : \Gamma, x : A}$$

où la prémisse de gauche est obtenue par le lemme 7.2.1. □

Remarque 7.2.2. Le cas de l'abstraction dans la preuve du théorème 7.2.1 exploite la convention de Barendregt pour traiter la substitution de façon naïve.

7.2.3. Réduction β

L'équivalence β définit une forme d'égalité entre les programmes à partir de la réduction β atomique $\rightsquigarrow_\beta$. Il est aussi très naturel de considérer le système de réécriture induit par la β-réduction atomique en s'autorisant à réécrire un sous-terme arbitraire.

Définition 7.2.7 (β-réduction)**.** La relation de *β-réduction*, notée $t \to_\beta t'$, est la plus petite relation T-contextuelle contenant la β-réduction atomique.

7.2.3.1. Réduction du sujet

On va prouver quelques propriétés élémentaires de la réduction β. La première est que les jugements de typage sont stables par réduction. Cette propriété, appelée *réduction du sujet*, est caractéristique des systèmes de types.

Théorème 7.2.2 (Réduction du sujet)**.** *Si* $\Gamma \vdash t : A$ *et* $t \to_\beta t'$ *alors* $\Gamma \vdash t' : A$.

Démonstration. Par induction sur la dérivation de $t \to_\beta t'$. On démontre les cas correspondant à la réduction β atomique en utilisant le théorème 7.2.1, les autres suivent par induction et clôture contextuelle. □

Remarque 7.2.3. Le théorème 7.2.2 exprime que le typage est préservé par la réduction. En revanche, ce système ne vérifie pas l'*expansion du sujet*, propriété dualle qui exprime que si $t \to_\beta t'$ et $\Gamma \vdash t' : A$ alors $\Gamma \vdash t : A$. Il est instructif de chercher des contre-exemples.

7.2.3.2. Confluence

Une deuxième propriété essentielle est la confluence. On va prouver celle-ci en utilisant la méthode dite *de Tait et Martin-Löf*, du nom de ses inventeurs William Tait et Per Martin-Löf. Il s'agit d'exploiter le lemme 4.2.2 en définissant une relation comprise entre la réduction β et sa clôture réflexive-transitive et qui de plus vérifie la propriété du diamant. Il existe un choix canonique en la matière.

$$\boxed{t \rightrightarrows_\beta t'}$$

$$\frac{}{\mathtt{var}(x) \rightrightarrows_\beta \mathtt{var}(x)} \qquad \frac{t \rightrightarrows_\beta t'}{\mathtt{fun}(x.t) \rightrightarrows_\beta \mathtt{fun}(x.t')} \qquad \frac{t \rightrightarrows_\beta t' \qquad u \rightrightarrows_\beta u'}{\mathtt{app}(t,u) \rightrightarrows_\beta \mathtt{app}(t',u')}$$

$$\frac{t \rightrightarrows_\beta t' \qquad u \rightrightarrows_\beta u'}{\mathtt{app}(\mathtt{fun}(x.t),u) \rightrightarrows_\beta t'[x\backslash u']} \qquad \frac{}{\mathtt{zero} \rightrightarrows_\beta \mathtt{zero}} \qquad \frac{t \rightrightarrows_\beta t'}{\mathtt{suc}(t) \rightrightarrows_\beta \mathtt{suc}(t')}$$

$$\frac{t \rightrightarrows_\beta t' \qquad u \rightrightarrows_\beta u' \qquad v \rightrightarrows_\beta v'}{\mathtt{fold}_{\mathtt{Nat}}(t,u,x.y.v) \rightrightarrows_\beta \mathtt{fold}_{\mathtt{Nat}}(t',u',x.y.v')} \qquad \frac{u \rightrightarrows_\beta u'}{\mathtt{fold}_{\mathtt{Nat}}(\mathtt{zero},u,x.y.v) \rightrightarrows_\beta u'}$$

$$\frac{t \rightrightarrows_\beta t' \qquad u \rightrightarrows_\beta u' \qquad v \rightrightarrows_\beta v'}{\mathtt{fold}_{\mathtt{Nat}}(\mathtt{suc}(t),u,x.y.v) \rightrightarrows_\beta v'[x\backslash t', y\backslash \mathtt{fold}_{\mathtt{Nat}}(t',u',x.y.v')]}$$

$$\boxed{\sigma \rightrightarrows_\beta \sigma'} \qquad \frac{}{\mathtt{id} \rightrightarrows_\beta \mathtt{id}} \qquad \frac{\sigma \rightrightarrows_\beta \sigma' \qquad t \rightrightarrows_\beta t'}{\sigma, x\backslash t \rightrightarrows_\beta \sigma', x\backslash t'}$$

FIG. 7.6. : Réduction parallèle de T

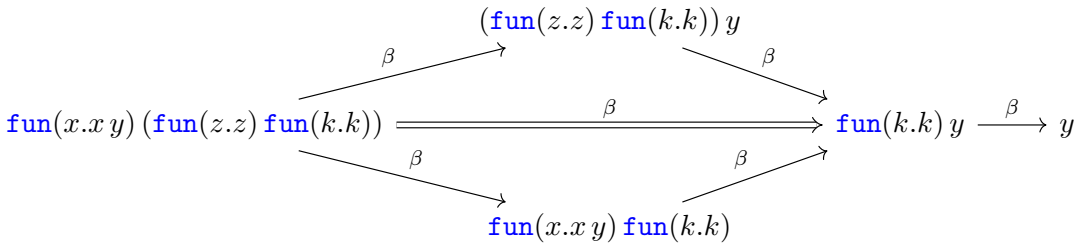


FIG. 7.7. : Exemple de graphe de réduction

Définition 7.2.8 (Réduction β parallèle)**.** La relation de *réduction β parallèle* de T, notée $\rightrightarrows_\beta$, est la relation définie inductivement à la figure 7.6.

Intuitivement, une étape de réduction β parallèle $t \rightrightarrows_\beta t'$ applique plusieurs étapes de réductions β atomiques $t_0 \rightsquigarrow_\beta t'_0$ telles que t_0 est un sous-terme de t. En revanche, elle ne permet pas d'appliquer des réductions β atomiques qui n'apparaissaient pas dans le terme de départ. Le graphe de réduction présenté à la figure 7.7, où les flèches simples représentent des réductions et les flèches doubles des réductions parallèles, illustre ce phénomène. Depuis $\mathtt{fun}(x.x\,y)\,(\mathtt{fun}(z.z)\,\mathtt{fun}(k.k))$ on peut atteindre $\mathtt{fun}(k.k)\,y$ en une seule étape de réduction parallèle, mais pas y.

Remarque 7.2.4. On peut aussi définir la réduction β-parallèle comme la *clôture* T-*parallèle* de la β-réduction atomique. Si Σ est une signature du second ordre, la clôture Σ-parallèle d'une relation R est la relation $\mu X.\Sigma(X)\,;R^?$.

On démontre maintenant les hypothèses du lemme 4.2.2.

Propriété 7.2.2. *La réduction β parallèle est réflexive,* T*-compatible et* T*-contextuelle.*

Propriété 7.2.3. *Si $t \to_\beta t'$ alors $t \rightrightarrows_\beta t'$.*

Démonstration. Puisque la réduction β parallèle est T-contextuelle, il suffit de traiter les équations (7.1) à (7.3). Elles suivent par réflexivité de la réduction parallèle. □

Propriété 7.2.4. *Si $t \rightrightarrows_\beta t'$ alors $t \to_\beta^* t'$.*

Démonstration. Par induction sur $t \rightrightarrows_\beta t'$ en utilisant le fait que $\to_\beta^*$ est T-compatible et contient la réduction β atomique. □

Propriété 7.2.5. *Si $t \rightrightarrows_\beta t'$ et $\sigma \rightrightarrows_\beta \sigma'$ alors $t[\sigma] \rightrightarrows_\beta t'[\sigma']$.*

Démonstration. Par une induction de routine sur t. □

On veut maintenant démontrer que $\rightrightarrows_\beta$ a la propriété du diamant. Pour ce faire, on emploie une approche simple proposée par Takahashi [10].

Définition 7.2.9. Le *contracté parallèle maximal* d'un terme t, noté $c(t)$, est un terme défini par récurrence sur t comme suit.

$$
\begin{aligned}
c(x) &= x \\
c(\mathtt{fun}(x.t_1)) &= \mathtt{fun}(x.c(t_1)) \\
c(\mathtt{app}(\mathtt{fun}(x.t_1'), t_2)) &= c(t_1')[x \backslash c(t_2)] \\
c(\mathtt{app}(t_1, t_2)) &= \mathtt{app}(c(t_1), c(t_2)) \quad (t_1 \neq \mathtt{fun}(\_.\_)) \\
c(\mathtt{zero}) &= \mathtt{zero} \\
c(\mathtt{suc}(t_1)) &= \mathtt{suc}(c(t_1)) \\
c(\mathtt{fold}_{\mathtt{Nat}}(\mathtt{zero}, t_2, x.y.t_3)) &= c(t_2) \\
c(\mathtt{fold}_{\mathtt{Nat}}(\mathtt{suc}(t_1'), t_2, x.y.t_3)) &= c(t_3)[x \backslash c(t_1'), y \backslash \mathtt{fold}_{\mathtt{Nat}}(c(t_1'), c(t_2), x.y.c(t_3))] \\
c(\mathtt{fold}_{\mathtt{Nat}}(t_1, t_2, x.y.t_3)) &= \mathtt{fold}_{\mathtt{Nat}}(c(t_1), c(t_2), x.y.c(t_3)) \quad (t_1 \neq \mathtt{zero}, \mathtt{suc}(\_))
\end{aligned}
$$

Notons que les clauses qui définissent la fonction c peuvent être lues comme la définition d'une fonction par filtrage de motif en OCaml.

Propriété 7.2.6. *Si $t \rightrightarrows_\beta t'$ alors $t' \rightrightarrows_\beta c(t)$.*

Démonstration. Par induction sur $t \rightrightarrows_\beta t'$. Comme précédemment, les cas correspondant aux règles de compatibilité sont immédiats. Les cas correspondants aux règles qui implémentent la réduction β atomique utilisent la propriété 7.2.5. Détaillons le cas $\mathtt{app}(\mathtt{fun}(x.t_1), t_2) \rightrightarrows_\beta t_1'[x \backslash t_2']$ avec $t_1 \rightrightarrows_\beta t_1'$ et $t_2 \rightrightarrows_\beta t_2'$. On doit montrer

$$t_1'[x \backslash t_2'] \rightrightarrows_\beta c(\mathtt{app}(\mathtt{fun}(x.t_1), t_2)) = c(t_1)[x \backslash c(t_2)].$$

Les hypothèses d'induction donnent $t_1' \rightrightarrows_\beta c(t_1)$ et $t_2' \rightrightarrows_\beta c(t_2)$. On obtient la réduction parallèle attendue par la propriété 7.2.5. □

$$\boxed{\mathsf{Nf}_\mathrm{T}} \qquad \frac{v \in \mathsf{Nf}_\mathrm{T}}{\mathtt{fun}(x.v) \in \mathsf{Nf}_\mathrm{T}} \qquad \frac{}{\mathtt{zero} \in \mathsf{Nf}_\mathrm{T}} \qquad \frac{v \in \mathsf{Nf}_\mathrm{T}}{\mathtt{suc}(v) \in \mathsf{Nf}_\mathrm{T}} \qquad \frac{e \in \mathsf{Ne}_\mathrm{T}}{e \in \mathsf{Nf}_\mathrm{T}}$$

$$\boxed{\mathsf{Ne}_\mathrm{T}} \qquad \frac{}{\mathtt{var}(x) \in \mathsf{Ne}_\mathrm{T}} \qquad \frac{e \in \mathsf{Ne}_\mathrm{T} \qquad v \in \mathsf{Nf}_\mathrm{T}}{\mathtt{app}(e, v) \in \mathsf{Ne}_\mathrm{T}} \qquad \frac{e \in \mathsf{Ne}_\mathrm{T} \qquad v \in \mathsf{Nf}_\mathrm{T} \qquad w \in \mathsf{Nf}_\mathrm{T}}{\mathtt{fold}_{\mathtt{Nat}}(e, v, x.y.w) \in \mathsf{Ne}_\mathrm{T}}$$

FIG. 7.8. : Termes normaux et neutres de T

Corollaire 7.2.1. *La réduction β parallèle a la propriété du diamant.*

Démonstration. Si $t \rightrightarrows_\beta t_1$ et $t \rightrightarrows_\beta t_2$ alors $t_1 \rightrightarrows_\beta c(t)$ et $t_2 \rightrightarrows_\beta c(t)$. □

Théorème 7.2.3. *La réduction β est confluente.*

Démonstration. On a $\to_\beta \subseteq \rightrightarrows_\beta \subseteq \to_\beta^*$ par les propriétés 7.2.3 et 7.2.4, et $\rightrightarrows_\beta$ a la propriété du diamant par le corollaire 7.2.1. On conclut par le lemme 4.2.2. □

Puisque la réduction β est confluente, on sait qu'elle vérifie la propriété de Church et Rosser (théorème 4.2.1), et donc que deux termes sont β-équivalents si et seulement s'ils ont un réduit commun. La confluence de la réduction β assure aussi l'unicité des formes $\to_\beta$-normales. On va discuter de la structure de ces formes normales.

7.2.3.3. Structure des formes normales bien typées

Soit t un terme bien typé de T qui est $\to_\beta$-normal, c'est-à-dire tel qu'il n'existe aucun u tel que $t \to_\beta u$. Quelle peut être la forme de t ? Il est clair que t peut être le terme $\mathtt{zero}$, ou bien une variable. Si t est de la forme $\mathtt{suc}(t')$, alors t' doit être normal. De même si t est de la forme $\mathtt{fun}(x.t')$. Si t est de la forme $\mathtt{app}(t_1, t_2)$, la situation est un peu plus complexe. Il faut que t_1 et t_2 soient normaux, puisque tout sous-terme d'un terme normal est normal, mais également que t_1 ne soit pas une abstraction. De plus, t_1 ne peut pas être le terme zéro ni un terme successeur puisque t est bien typé. En résumé, le terme $\mathtt{app}(t_1, t_2)$ est normal si et seulement si t_2 est normal et t_1 est normal et n'est pas une forme d'introduction. Le même raisonnement appliqué au cas où t est de la forme $\mathtt{fold}_{\mathtt{Nat}}(t_1, t_2, x.y.t_3)$ permet de conclure que ce terme est $\to_\beta$-normal si et seulement si t_2 et t_3 sont β-normaux et que t_1 est $\to_\beta$-normal et n'est pas une forme d'introduction.

Pour résumer les observations précédentes, on va caractériser inductivement l'ensemble des formes $\to_\beta$-normales bien typées en même temps que l'ensemble des formes *β-neutres*, c'est-à-dire des formes $\to_\beta$-normales bien typées qui ne sont pas des formes d'introduction.

Propriété 7.2.7 (Termes normaux et neutres)**.** *Un terme bien typé de* T *est $\to_\beta$-normal si et seulement s'il appartient à l'ensemble* Nf_T *défini inductivement à la figure 7.8.*

$$\mathsf{HKon}_{\mathrm{T}} \ni H ::= W \mid \mathtt{suc}(H) \mid \mathtt{fun}(x.H) \qquad \textbf{Contextes de tête}$$
$$\mathsf{WHKon}_{\mathrm{T}} \ni W ::= \diamond \mid \mathtt{app}(W,t) \mid \mathtt{fold}_{\mathtt{Nat}}(W,t,x.y.t') \qquad \textbf{Contextes de tête faibles}$$

FIG. 7.9. : Contextes de tête et de tête faible de T

Informellement, un terme normal est une succession de formes d'introduction éventuellement suivie d'un terme neutre. L'ensemble des termes neutres est l'ensemble Ne_{T} défini à la figure 7.8. Un terme neutre est une succession de formes d'élimination ultimement « bloquée » par la présence d'une variable. Comme attendu, un terme normal clos de type $\mathtt{Nat}$ est nécessairement de la forme $\underline{k}$ pour $k \in \mathbb{N}$.

7.3. Autour de la réduction

7.3.1. Réduction de tête

Les *contextes de tête* et les *contextes de tête faibles* sont les entités syntaxiques décrits par la grammaire présentée à la figure 7.9. Un contexte de tête est donc un terme de T où le symbole formel $\diamond$, appelé *trou*, apparaît exactement une fois. Le trou doit de plus apparaître à gauche d'une application ou d'une récursion primitive. On écrit $H\{t\}$ pour le terme obtenu en remplaçant naïvement le symbole $\diamond$ par t dans H. Ici, « naïvement » signifie que cette forme de substitution autorise la capture. Par exemple, le terme $\mathtt{fun}(x.\diamond)\{x\}$ est bien $\mathtt{fun}(x.x)$ plutôt que $\mathtt{fun}(y.x)$ avec y une variable fraîche quelconque. Ainsi, le trou n'est pas une variable au sens usuel.

Soit R une endorelation des termes de T. L'*extension contextuelle de tête* et l'*extension contextuelle de tête faible* de R, notées respectivement $\mathsf{Cont}_{\mathsf{h}}[\Sigma_{\mathrm{T}}](R)$ et $\mathsf{Cont}_{\mathsf{wh}}[\Sigma_{\mathrm{T}}](R)$, sont les endorelations des termes de T définies comme suit.

$$\mathsf{Cont}_{\mathsf{h}}[\Sigma_{\mathrm{T}}](R) \coloneqq \{(H\{t\}, H\{t'\}) \mid (t,t') \in R, H \in \mathsf{HKon}_{\mathrm{T}}\}$$
$$\mathsf{Cont}_{\mathsf{wh}}[\Sigma_{\mathrm{T}}](R) \coloneqq \{(W\{t\}, W\{t'\}) \mid (t,t') \in R, W \in \mathsf{WHKon}_{\mathrm{T}}\}$$

Une relation R est *contextuelle de tête* si elle contient $\mathsf{Cont}_{\mathsf{h}}[\Sigma_{\mathrm{T}}](R)$ et *contextuelle de tête faible* si elle contient $\mathsf{Cont}_{\mathsf{wh}}[\Sigma_{\mathrm{T}}](R)$.

Définition 7.3.1. La *β-réduction de tête*, notée $\to_{\mathsf{h}}$, est la plus petite relation contextuelle de tête qui contient la β-réduction atomique. La *β-réduction de tête faible*, notée $\to_{\mathsf{wh}}$, est la plus petite relation contextuelle de tête faible qui contient la β-réduction atomique.

Il est facile de vérifier que toute relation contextuelle de tête faible est contextuelle de tête, et que toute relation contextuelle de tête est contextuelle, ce qui entraîne que les inclusions $\to_{\mathsf{wh}} \subseteq \to_{\mathsf{h}} \subseteq \to_{\beta}$. Ces inclusions sont strictes. Par exemple, le terme $\mathtt{fun}(x.\mathtt{app}(\mathtt{fun}(y.y), x))$ est normal pour la réduction β de tête faible mais pas

$$\boxed{\mathsf{Nf}^{\mathsf{h}}_{\mathrm{T}}} \qquad \frac{v \in \mathsf{Nf}^{\mathsf{h}}_{\mathrm{T}}}{\mathtt{fun}(x.v) \in \mathsf{Nf}^{\mathsf{h}}_{\mathrm{T}}} \qquad \frac{}{\mathtt{zero} \in \mathsf{Nf}^{\mathsf{wh}}_{\mathrm{T}}} \qquad \frac{v \in \mathsf{Nf}^{\mathsf{h}}_{\mathrm{T}}}{\mathtt{suc}(v) \in \mathsf{Nf}^{\mathsf{h}}_{\mathrm{T}}} \qquad \frac{e \in \mathsf{Nf}^{\mathsf{wh}}_{\mathrm{T}}}{e \in \mathsf{Nf}^{\mathsf{h}}_{\mathrm{T}}}$$

$$\boxed{\mathsf{Nf}^{\mathsf{wh}}_{\mathrm{T}}} \qquad \frac{}{\mathtt{fun}(x.t) \in \mathsf{Nf}^{\mathsf{wh}}_{\mathrm{T}}} \qquad \frac{}{\mathtt{zero} \in \mathsf{Nf}^{\mathsf{wh}}_{\mathrm{T}}} \qquad \frac{}{\mathtt{suc}(t) \in \mathsf{Nf}^{\mathsf{wh}}_{\mathrm{T}}} \qquad \frac{e \in \mathsf{Nf}^{\mathsf{wh}}_{\mathrm{T}}}{e \in \mathsf{Nf}^{\mathsf{wh}}_{\mathrm{T}}}$$

FIG. 7.10. : Termes normaux de tête et de tête faible de T

pour la réduction β de tête, tandis que le terme $\mathtt{app}(z, \mathtt{app}(\mathtt{fun}(x.x), y))$ est normal pour la réduction β de tête mais pas pour la réduction β générale.

Il est utile de caractériser inductivement les formes $\to_{\mathsf{h}}$-normales et $\to_{\mathsf{wh}}$-normales des termes, à l'image du travail réalisé pour la β-réduction générale à la figure 7.8.

Définition 7.3.2. On dira qu'un terme t est *neutre de tête*, et on notera $t \in \mathsf{Nf}^{\mathsf{wh}}_{\mathrm{T}}$, s'il peut s'écrire comme $t = W\{x\}$ avec W un contexte de tête faible.

Propriété 7.3.1. *Un terme bien typé est $\to_{\mathsf{h}}$-normal (resp. $\to_{\mathsf{wh}}$-normal) si et seulement s'il appartient à l'ensemble $\mathsf{Nf}^{\mathsf{h}}_{\mathrm{T}}$ (resp. $\mathsf{Nf}^{\mathsf{wh}}_{\mathrm{T}}$) défini inductivement à la figure 7.10.*

7.3.2. Standardisation

On va utiliser la réduction de tête faible pour définir la réduction *standard*. Il s'agit d'une stratégie (non déterministe) de réduction où l'on choisit de calculer "de l'extérieur vers l'intérieur". Intuitivement, une fois qu'on a commencé à réduire un sous-terme du terme de départ, on s'interdit de réduire plus haut dans l'arbre de syntaxe. On va formaliser cette notion via une définition inductive commode dûe à KASHIMA [5].

Définition 7.3.3. La réduction standard, notée $\rightrightarrows_{\mathsf{std}}$, est la plus petite endorelation des termes de T qui contient $\to^*_{\mathsf{wh}} ; \Sigma^{\bullet}_{\mathrm{T}}(\rightrightarrows_{\mathsf{std}})$. Autrement dit, il s'agit de la plus petite relation close par les règles de la figure 7.11. On étend cette relation aux substitutions terme à terme, et on note également la relation obtenue $\rightrightarrows_{\mathsf{std}}$.

Propriété 7.3.2. *Si $t \rightrightarrows_{\mathsf{std}} u$ alors $t \to^*_{\beta} u$.*

Démonstration. Par induction sur la dérivation de $t \rightrightarrows_{\mathsf{std}} u$, en utilisant l'inclusion $\to_{\mathsf{wh}} \subseteq \to_{\beta}$ et la compatibilité de $\to^*_{\beta}$. □

On va démontrer le théorème de *standardisation*, qui est la réciproque de la propriété 7.3.2. Ce résultat, non trivial, ne concerne que la relation de réduction et ne dépend pas des hypothèses de typage. On en donne une preuve élémentaire dûe à KASHIMA [5].

Les trois résultats qui suivent se prouvent par des inductions de routine.

Propriété 7.3.3. *La réduction standard est une relation réflexive.*

$$\frac{t \to_{\mathsf{wh}}^* x}{t \rightrightarrows_{\mathsf{std}} x} \qquad \frac{t \to_{\mathsf{wh}}^* \mathtt{zero}}{\mathtt{zero} \rightrightarrows_{\mathsf{std}} \mathtt{zero}} \qquad \frac{t \to_{\mathsf{wh}}^* \mathtt{fun}(x.t_1) \qquad t_1 \rightrightarrows_{\mathsf{std}} u_1}{t \rightrightarrows_{\mathsf{std}} \mathtt{fun}(x.u_1)}$$

$$\frac{t \to_{\mathsf{wh}}^* \mathtt{app}(t_1, t_2) \qquad t_1 \rightrightarrows_{\mathsf{std}} u_1 \qquad t_2 \rightrightarrows_{\mathsf{std}} u_2}{t \rightrightarrows_{\mathsf{std}} \mathtt{app}(u_1, u_2)} \qquad \frac{t \to_{\mathsf{wh}}^* \mathtt{suc}(t_1) \qquad t_1 \rightrightarrows_{\mathsf{std}} u_1}{t \rightrightarrows_{\mathsf{std}} \mathtt{suc}(u_1)}$$

$$\frac{t \to_{\mathsf{wh}}^* \mathtt{fold}_{\mathtt{Nat}}(t_1, t_2, x.y.t_3) \qquad t_1 \rightrightarrows_{\mathsf{std}} u_1 \qquad t_2 \rightrightarrows_{\mathsf{std}} u_2 \qquad t_3 \rightrightarrows_{\mathsf{std}} u_3}{t \rightrightarrows_{\mathsf{std}} \mathtt{fold}_{\mathtt{Nat}}(u_1, u_2, x.y.u_3)}$$

FIG. 7.11. : Réduction standard de T

Lemme 7.3.1. *Si* $t \to_{\mathsf{wh}}^* t'$ *et* $t' \rightrightarrows_{\mathsf{std}} u$ *alors* $t \rightrightarrows_{\mathsf{std}} u$.

Lemme 7.3.2. *Si* $t \rightrightarrows_{\mathsf{std}} t'$ *et* $\sigma \rightrightarrows_{\mathsf{std}} \sigma'$ *alors* $t[\sigma] \rightrightarrows_{\mathsf{std}} t'[\sigma']$.

Lemme 7.3.3. *Si* $t \rightrightarrows_{\mathsf{std}} u$ *et* $u \rightsquigarrow_\beta v$ *alors* $t \rightrightarrows_{\mathsf{std}} v$.

Démonstration. On procède par analyse de cas sur $u \rightsquigarrow_\beta v$ puis sur $t \rightrightarrows_{\mathsf{std}} u$. On détaille le cas $u = \mathtt{app}(\mathtt{fun}(x.u_1), u_2)$ et $v = u_1[x \backslash u_2]$, les deux autres étant similaires.

Par inversion de l'hypothèse $t \rightrightarrows_{\mathsf{std}} u$, on a nécessairement $t \to_{\mathsf{wh}}^* \mathtt{app}(t_1, t_2)$ avec les réductions $t_1 \rightrightarrows_{\mathsf{std}} \mathtt{fun}(x.u_1)$ et $t_2 \rightrightarrows_{\mathsf{std}} u_2$. Par une nouvelle inversion de l'hypothèse $t_1 \rightrightarrows_{\mathsf{std}} \mathtt{fun}(x.u_1)$, on a $t_1 \to_{\mathsf{wh}}^* \mathtt{fun}(x.t_1')$ avec $t_1' \rightrightarrows_{\mathsf{std}} u_1$. On en déduit

$$t \to_{\mathsf{wh}}^* \mathtt{app}(\mathtt{fun}(x.t_1'), t_2) \to_{\mathsf{wh}}^* t_1'[x \backslash t_2] \rightrightarrows_{\mathsf{std}} u_1[x \backslash u_2]$$

en utilisant les lemmes 7.3.1 et 7.3.2. □

Lemme 7.3.4. *Si* $t \rightrightarrows_{\mathsf{std}} u$ *et* $u \to_\beta v$ *alors* $t \rightrightarrows_{\mathsf{std}} v$.

Démonstration. Par induction sur la dérivation de $u \to_\beta v$. Le cas de base correspond au lemme 7.3.3 tandis que les cas d'induction sont routiniers. □

Théorème 7.3.1 (Standardisation)**.** *Si* $t \to_\beta^* u$ *alors* $t \rightrightarrows_{\mathsf{std}} u$.

Démonstration. Par la propriété 7.3.3 et le lemme 7.3.4. □

On a déjà discuté du fait que la relation de réduction standard $t \rightrightarrows_{\mathsf{std}} u$ correspond à une réduction $t \to_\beta^* u$ où les réductions sont faites de l'extérieur vers l'intérieur. La relation $\rightrightarrows_\ell$, définie à la figure 7.12, correspond à un cas particulier où on réduit maximalement les redex extérieurs. Puisque la réduction de tête faible est déterministe, sa relation de normalisation $\widehat{\to_{\mathsf{wh}}}$ l'est également, et donc $\rightrightarrows_\ell$ l'est également. De plus, on va montrer que cette relation trouve toujours les formes $\to_\beta$ normales, lorsqu'elles existent.

Propriété 7.3.4. *Si* $t \rightrightarrows_\ell u$ *alors* $t \rightrightarrows_{\mathsf{std}} u$.

Démonstration. Par une induction de routine en utilisant l'inclusion $\widehat{\to_{\mathsf{wh}}} \subseteq \to_{\mathsf{wh}}^*$. □

$$\frac{t \mathrel{\widehat{\to_{\mathsf{wh}}}} x}{t \rightrightarrows_\ell x} \qquad \frac{t \mathrel{\widehat{\to_{\mathsf{wh}}}} \mathtt{zero}}{\mathtt{zero} \rightrightarrows_\ell \mathtt{zero}} \qquad \frac{t \mathrel{\widehat{\to_{\mathsf{wh}}}} \mathtt{fun}(x.t_1) \qquad t_1 \rightrightarrows_\ell u_1}{t \rightrightarrows_\ell \mathtt{fun}(x.u_1)}$$

$$\frac{t \mathrel{\widehat{\to_{\mathsf{wh}}}} \mathtt{app}(t_1, t_2) \qquad t_1 \rightrightarrows_\ell u_1 \qquad t_2 \rightrightarrows_\ell u_2}{t \rightrightarrows_\ell \mathtt{app}(u_1, u_2)} \qquad \frac{t \mathrel{\widehat{\to_{\mathsf{wh}}}} \mathtt{suc}(t_1) \qquad t_1 \rightrightarrows_\ell u_1}{t \rightrightarrows_\ell \mathtt{suc}(u_1)}$$

$$\frac{t \mathrel{\widehat{\to_{\mathsf{wh}}}} \mathtt{fold}_{\mathtt{Nat}}(t_1, t_2, x.y.t_3) \qquad t_1 \rightrightarrows_\ell u_1 \qquad t_2 \rightrightarrows_\ell u_2 \qquad t_3 \rightrightarrows_\ell u_3}{t \rightrightarrows_\ell \mathtt{fold}_{\mathtt{Nat}}(u_1, u_2, x.y.u_3)}$$

FIG. 7.12. : Réduction gauche de T

Lemme 7.3.5. *Si $t \mathrel{\widehat{\to_{\mathsf{wh}}}} t'$ et $t' \rightrightarrows_{\mathsf{std}} u$ alors $t \rightrightarrows_{\mathsf{std}} u$.*

Lemme 7.3.6. *Soient t et u des termes tels que $t \rightrightarrows_{\mathsf{std}} u$.*

- *Si u est normal, alors $t \rightrightarrows_\ell u$.*
- *Si u est neutre, alors il existe v neutre de tête tel que $t \mathrel{\widehat{\to_{\mathsf{wh}}}} v$ et $v \rightrightarrows_\ell u$.*

Démonstration. On raisonne par induction mutuelle sur la neutralité et la normalité de u en utilisant la caractérisation de la figure 7.8. On ne détaille que trois cas représentatifs.

- Traitons d'abord les cas d'induction où u est normal.
  - Cas $u = \mathtt{zero}$: par inversion, on a $t \to_{\mathsf{wh}}^* \mathtt{zero}$. On a $t \mathrel{\widehat{\to_{\mathsf{wh}}}} \mathtt{zero}$ et on conclut immédiatement.
  - Cas $u = \mathtt{fun}(x.u_1)$ avec u_1 normal : par inversion, on a $t \to_{\mathsf{wh}}^* \mathtt{fun}(x.t_1)$ avec $t_1 \rightrightarrows_{\mathsf{std}} u_1$. On a $t \mathrel{\widehat{\to_{\mathsf{wh}}}} \mathtt{fun}(x.t_1)$ et donc on conclut par application de l'hypothèse d'induction.
  - Cas $u = \mathtt{suc}(u_1)$ avec u_1 normal : similaire au cas précédent.
  - Cas u neutre : l'autre partie de l'énoncé nous donne v neutre tel que $t \mathrel{\widehat{\to_{\mathsf{wh}}}} v$ et $v \rightrightarrows_\ell u$. On conclut par le lemme 7.3.5.
- Traitons maintenant les cas d'induction où u est neutre.
  - Cas $u = x$: par inversion, on a $t \to_{\mathsf{wh}}^* x$. Mais x est neutre de tête et donc normal de tête faible. On pose $v \coloneqq u$ et on conclut immédiatement puisque $t \mathrel{\widehat{\to_{\mathsf{wh}}}} x$.
  - Cas $u = \mathtt{app}(u_1, u_2)$ avec u_1 neutre et u_2 normal : par inversion, on a $t \to_{\mathsf{wh}}^* \mathtt{app}(t_1, t_2)$ avec $t_1 \rightrightarrows_{\mathsf{std}} u_1$ et $t_2 \rightrightarrows_{\mathsf{std}} u_2$. En appliquant les hypothèses d'induction, on obtient v_1 neutre de tête tel que $t_1 \mathrel{\widehat{\to_{\mathsf{wh}}}} v_1$ et $v_1 \rightrightarrows_{\mathsf{std}} u_1$. On choisit $v \coloneqq \mathtt{app}(v_1, t_2)$ qui est neutre de tête et donc normal de tête faible. On a $t \mathrel{\widehat{\to_{\mathsf{wh}}}} \mathtt{app}(v_1, t_2)$ et on conclut par application des hypothèses.

– Le cas $u = \mathtt{fold}_{\mathtt{Nat}}(u_1, u_2, x.y.u_3)$ avec u_1 neutre et u_2, u_3 normaux est similaire au précédent. □

Théorème 7.3.2. *Si t a une forme $\to_\beta$-normale u alors $t \rightrightarrows_\ell u$.*

Démonstration. On a $t \to_\beta^* u$ et donc, par le théorème 7.3.1, $t \rightrightarrows_{\mathsf{std}} u$. Puisque u est normal, on conclut par le lemme 7.3.6. □

7.3.3. Une machine abstraite pour la réduction de tête faible

La réduction de tête faible est une relation de réduction déterministe. C'est la conséquence du résultat suivant.

Lemme 7.3.7 (Décomposition unique)**.** *Pour tout terme t, soit t est en forme normale de tête faible, soit il existe un unique contexte de tête faible W et terme t' tel que $t = W\{t'\}$ et t' appartient au domaine de $\rightsquigarrow_\beta$.*

Démonstration. Induction de routine. □

Le lemme 7.3.7 suggère une technique d'implémentation pour la réduction de tête faible qui consisterait à décomposer le terme d'entrée, appliquer la réduction β-atomique, puis recomposer. On peut ensuite enchaîner ces étapes de décomposition / réduction / recomposition, comme dans la séquence qui suit.

$$t_0 = W_0\{t_0'\} \ \to_{\mathsf{wh}} \ W_0\{u_0'\} = t_1 = W_1\{t_1'\} \ \to_{\mathsf{wh}} \ W_1\{u_1'\} = t_2 = W_2\{t_2'\} \ \to_{\mathsf{wh}} \ \dots$$

Cependant, une telle implémentation est peu efficace. On peut faire mieux en observant que les contextes W_i et W_{i+1} ne sont jamais très différents : W_{i+1} est toujours obtenu en étendant W_i de manière purement locale. Cette observation, dûe à Danvy et Nielsen [2], sous-tend la construction de *machines abstraites*, c'est-à-dire de systèmes de transitions qui manipulent termes et contextes localement pour simuler la réduction.

La figure 7.13 présente une machine abstraite qui implémente la réduction de tête faible pour le système T. Une machine abstraite repose sur de nouveaux objets syntaxiques, les *piles* et les *configuration*, décrites à la figure 7.13a. Une configuration $\langle t \parallel s \rangle$ représente un état de la machine abstraite, toujours formé d'un terme t et d'une pile s. La figure 7.13b présente les transitions. Les deux premières transitions déconstruisent le terme pour étendre la pile. Les trois suivantes implémentent la β-réduction atomique.

Une pile s correspond à un contexte de tête faible représenté à l'envers, au sens où le constructeur de pile le plus externe de s est le plus proche du trou $\diamond$, et donc le plus interne, dans le contexte de tête faible correspondant. Pour rendre cette observation précise, on définit la fonction rev qui associe à toute pile s le contexte de tête faible $\mathsf{rev}(s)$ correspondant.

$$\begin{aligned} \mathsf{rev}(\diamond) &= t \\ \mathsf{rev}(\mathtt{app}(\diamond, t_2) :: s) &= \mathsf{rev}(s)\{\mathtt{app}(\diamond, t_2)\} \\ \mathsf{rev}(\mathtt{fold}_{\mathtt{Nat}}(\diamond, t_2, x.y.t_3) :: s) &= \mathsf{rev}(s)\{\mathtt{fold}_{\mathtt{Nat}}(\diamond, t_2, x.y.t_3)\} \end{aligned}$$

$$\mathsf{Stack}_{\mathrm{T}} \ni s ::= \diamond \mid \mathtt{app}(\diamond, t) :: s \mid \mathtt{fold}_{\mathtt{Nat}}(\diamond, t, x.y.t') :: s \qquad \textbf{Piles}$$
$$\mathsf{Conf}_{\mathrm{T}} \ni c ::= \langle t \parallel S \rangle \qquad \textbf{Configuration}$$

(a) composantes de la machine

$$\begin{aligned}
\langle \mathtt{app}(t,u) \parallel W \rangle &\rightarrowtail \langle t \parallel \mathtt{app}(\diamond,u) :: W \rangle \\
\langle \mathtt{fold}_{\mathtt{Nat}}(t_1,t_2,x.y.t_3) \parallel W \rangle &\rightarrowtail \langle t_1 \parallel \mathtt{fold}_{\mathtt{Nat}}(\diamond,t_2,x.y.t_3) :: W \rangle \\
\langle \mathtt{fun}(x.t) \parallel \mathtt{app}(\diamond,u) :: W \rangle &\rightarrowtail \langle t[x\backslash u] \parallel W \rangle \\
\langle \mathtt{zero} \parallel \mathtt{fold}_{\mathtt{Nat}}(\diamond,t_2,x.y.t_3) :: W \rangle &\rightarrowtail \langle t_2 \parallel W \rangle \\
\langle \mathtt{suc}(t) \parallel \mathtt{fold}_{\mathtt{Nat}}(\diamond,t_2,x.y.t_3) :: W \rangle &\rightarrowtail \langle t_3[x\backslash t, y\backslash \mathtt{fold}_{\mathtt{Nat}}(t,t_2,x.y.t_3)] \parallel W \rangle
\end{aligned}$$

(b) transitions de la machine abstraite $\boxed{c \rightarrowtail c'}$

FIG. 7.13. : machine abstraite pour T

La fonction rewind recompose (ou « rembobine ») la pile s sur le terme t d'un configuration $\langle t \parallel s \rangle$, défaisant les transitions de la machine qui ont bâti la pile s.

$$\begin{aligned}
\mathsf{rewind}(\langle t \parallel \diamond \rangle) &= t \\
\mathsf{rewind}(\langle t_1 \parallel \mathtt{app}(\diamond,t_2) :: s \rangle) &= \mathsf{rewind}(\langle \mathtt{app}(t_1,t_2) \parallel s \rangle) \\
\mathsf{rewind}(\langle t_1 \parallel \mathtt{fold}_{\mathtt{Nat}}(\diamond,t_2,x.y.t_3) :: s \rangle) &= \mathsf{rewind}(\langle \mathtt{fold}_{\mathtt{Nat}}(t_1,t_2,x.y.t_3) \parallel s \rangle)
\end{aligned}$$

Les fonction rev et rewind sont connectées par le résultat qui suit.

Lemme 7.3.8. *On a* $\mathsf{rewind}(\langle t \parallel s \rangle) = \mathsf{rev}(s)\{t\}$.

La correction de la machine abstraite est exprimée par les résultats suivants.

Lemme 7.3.9. *Si* $c \rightarrowtail c'$ *alors* $\mathsf{plug}(c) \rightarrow_{\mathsf{wh}}^{?} \mathsf{plug}(c')$.

Démonstration. Par cas sur $c \rightarrowtail c'$. Les deux premières règles correspondent à $\mathsf{plug}(c) = \mathsf{plug}(c')$. Les trois seconds correspondent à la β-réduction atomique. □

Lemme 7.3.10. *On a* $\langle W\{t_0\} \parallel s \rangle \rightarrowtail^* \langle t_0 \parallel s' \rangle$ *avec* $\mathsf{rev}(s') = \mathsf{rev}(s)\{W\}$.

Lemme 7.3.11. *Si* $\mathsf{plug}(c) \rightarrow_{\mathsf{wh}} t'$ *alors il existe* c' *tel que* $t' = \mathsf{plug}(c')$ *et* $c \rightarrowtail^+ c'$.

Théorème 7.3.3. *Soit* t *un terme clos. Si* $t \widehat{\rightarrow_{\mathsf{wh}}} u$ *alors* $\langle t \parallel \diamond \rangle \rightarrowtail^* \langle u \parallel \diamond \rangle$.

7.4. Canonicité et modèle ensembliste

7.4.1. Canonicité

Ce chapitre s'ouvre en affirmant que les programmes écrits en T ont un comportement très discipliné, ce qu'exprime notamment le théorème qui suit.

Théorème 7.4.1 (Canonicité). *Si* $t : \mathtt{Nat}$ *alors il existe un unique entier* k *tel que* $t \equiv_\beta \underline{k}$.

Démontrer ce théorème est plus difficile que les précédents. Il est instructif d'essayer par une induction directe sur la dérivation de typage. Très formellement, on tente donc de démontrer la propriété

$$\forall \Gamma \in \mathsf{Con}_\mathrm{T}, \forall A \in \mathsf{Type}_\mathrm{T}, \forall t \in \mathcal{T}[\Sigma_\mathrm{T}](\Gamma; A), \Gamma = \cdot \wedge A = \mathtt{Nat} \Rightarrow \exists! k \in \mathbb{N}, t \equiv_\beta \underline{k}.$$

Tout d'abord, remarquons que la plupart des cas n'ont pas besoin d'être considérés parce qu'ils sont incompatibles avec nos hypothèses. Par exemple, la règle de la variable ne s'applique que lorsque le contexte Γ n'est pas vide, ce qui est absurde puisque $\Gamma = \cdot$ par hypothèse. De même, la règle de l'abstraction ne s'applique que lorsque le type A est un type fonctionnel, ce qui est aussi absurde puisque $A = \mathtt{Nat}$ par hypothèse.

Considérons ensuite le cas du zéro et du successeur. Pour le zéro, on choisit $k = 0$, puisque $\mathtt{zero} \equiv_\beta \underline{0}$ par réflexivité. En revanche, on ne voit pas comment prouver l'unicité. Pour le successeur, l'hypothèse d'induction donne un unique k' tel que $t \equiv_\beta \underline{k'}$; on choisit $k = k' + 1$, et on a bien $\mathtt{suc}(t) \equiv_\beta \underline{k' + 1} = \mathtt{suc}(\underline{k'})$ par congruence. Là encore l'unicité semble rester inaccessible.

Enfin, les cas des formes d'élimination (application et récursion primitive) sont hautement problématiques et nous empêchent définitivement de compléter la preuve, même restreinte à l'existence. Considérons le cas de l'application. Étant donné nos hypothèses sur Γ et A, la règle doit avoir la forme qui suit.

$$\frac{\cdot \vdash t_1 : A \to \mathtt{Nat} \qquad \cdot \vdash t_2 : A}{\cdot \vdash \mathtt{app}(t_1, t_2) : \mathtt{Nat}}$$

Le problème est que nos hypothèses d'induction ne s'appliquent qu'à des termes de type $\mathtt{Nat}$, et donc ne fournissent aucune information au sujet de t_1 et t_2. Nous ne pouvons donc pas traiter ce cas. Le même problème se pose pour la récursion primitive.

Il faudrait donc généraliser notre hypothèse d'induction pour traiter des types fonctionnels. Mais pour traiter les types fonctionnels, il faudra traiter la règle de l'abstraction, dont la prémisse implique un contexte qui n'est jamais vide. Il faut donc également généraliser l'hypothèse d'induction pour traiter des contextes non vides. Enfin, il faut trouver un argument pour démontrer l'unicité. Pour satisfaire toutes ces contraintes, on va utiliser une méthode très souple, qui est un outil standard en sémantique des langages de programmation : on va bâtir un *modèle* de T.

Informellement, un modèle de T définit une interprétation de chaque type, contexte et terme typé du langage par un objet mathématique. La nature exacte des objets mathématiques considérés dépend du modèle, et en faisant varier celle-ci, on peut déduire diverses conséquences sur le langage. De plus, l'interprétation des termes doit être cohérente avec celle des types et contextes, et doit respecter certaines contraintes :

1. elle doit être *fonctorielle*, c'est-à-dire commuter avec la substitution ;
2. elle doit être *correcte*, c'est-à-dire égaliser la relation d'équivalence considérée.

Dans cette section, nos modèles seront corrects vis-à-vis de l'équivalence β, au sens où l'équivalence $t \equiv_\beta u$ entraîne l'égalité des interprétations de t et u.

7.4.2. Modèle syntaxique

La façon la plus grossière de construire un modèle est simplement de quotienter la syntaxe par la relation d'équivalence β. Pour cela, on a besoin de quelques définitions auxiliaires.

Définition 7.4.1. Un *environnement* définissant un contexte Γ est un élément de l'ensemble $\mathcal{T}[\Sigma_{\mathrm{T}}](\cdot;\Gamma)$.

Définition 7.4.2. On écrit $\Gamma \vdash t \equiv_\beta u : A$ lorsque $\Gamma \vdash t : A$ et $\Gamma \vdash u : A$ avec $t \equiv_\beta u$. Cette équivalence s'étend aux substitutions de manière structurelle : deux substitutions sont équivalentes modulo β lorsqu'elles lient les mêmes variables et que les termes liés sont équivalents, comme exprimé par le jugement inductif ci-dessous.

$$\boxed{\Delta \vdash \sigma \equiv_\beta \varphi : \Gamma} \qquad \frac{\Delta \supseteq \Gamma}{\Delta \vdash \mathtt{id} \equiv_\beta \mathtt{id} : \Gamma} \qquad \frac{\Delta \vdash \sigma \equiv_\beta \varphi : \Gamma \qquad \Delta \vdash t \equiv_\beta u : A}{\Delta \vdash \sigma, x\backslash t \equiv_\beta \varphi, x\backslash u : \Gamma, x : A}$$

Lemme 7.4.1. *Si $\Gamma \vdash t \equiv_\beta u : A$ et $\Delta \vdash \sigma \equiv_\beta \varphi : \Gamma$ alors $\Delta \vdash t[\sigma] \equiv_\beta u[\varphi] : A$.*

Démonstration. Par une induction de routine, en exploitant le fait que l'équivalence β est (par définition) une T-congruence. □

Définition 7.4.3 (Modèle syntaxique)**.** Le modèle syntaxique de T interprète tout terme de T par sa classe d'équivalences modulo β. Plus précisément, l'interprétation $[\![A]\!]_{\mathsf{Syn}}$ d'un type A est A lui-même ; de même pour les contextes de typage ; un terme $\Gamma \vdash t : A$ est interprété par $[t]_\beta$.

Dans ce qui suit, on écrira $[\![A]\!]_{\mathsf{Syn}}$ pour l'ensemble des termes clos de type A quotientés par l'équivalence β. En particulier, si $\cdot \vdash t : A$ alors $[\![t]\!]_{\mathsf{Syn}} \in [\![A]\!]_{\mathsf{Syn}}$. On écrira également $[\![\Gamma]\!]_{\mathsf{Syn}}$ pour l'ensemble des environnements définissant Γ quotientés par l'équivalence β telle qu'exprimée par la définition 7.4.2.

La fonctorialité du modèle correspond ici au lemme 7.4.1, et la correction est immédiate par définition. Toutefois, ce modèle n'est pas utile en tant que tel, puisqu'on ne sait pas démontrer ses propriétés facilement. On va utiliser un modèle plus utile dans ce qui suit.

7.4.3. Modèle ensembliste

Propriété 7.4.1. *Soit X un ensemble, f_0 un élément de X, f_s une fonction de $\mathbb{N} \times X$ dans X. Alors il existe une unique fonction $\mathit{fold}_{\mathbb{N}}(f_0, f_s)$ de $\mathbb{N}$ dans X telle que*

$$\begin{aligned} \mathit{fold}_{\mathbb{N}}(f_0, f_s)(0) &= f_0, \\ \mathit{fold}_{\mathbb{N}}(f_0, f_s)(1+n) &= f_s(n, \mathit{fold}_{\mathbb{N}}(f_0, f_s)(n)). \end{aligned}$$

Chaque type A de T est interprété par un ensemble $[\![A]\!]_{\mathsf{Ens}}$ défini par récurrence sur la structure de A. De même pour les contextes, interprétés par des produits cartésiens.

$$\left[\!\!\left[\frac{}{\Delta \supseteq \cdot}\right]\!\!\right]_{\mathsf{Ens}}(E) = ()$$
$$\left[\!\!\left[\frac{\Delta \supseteq \Gamma}{\Delta, x : A \supseteq \Gamma}\right]\!\!\right]_{\mathsf{Ens}}(E, v) = [\![\Delta \supseteq \Gamma]\!]_{\mathsf{Ens}}(E)$$
$$\left[\!\!\left[\frac{\Delta \supseteq \Gamma}{\Delta, x : A \supseteq \Gamma, x : A}\right]\!\!\right]_{\mathsf{Ens}}(E, v) = ([\![\Delta \supseteq \Gamma]\!]_{\mathsf{Ens}}(E), v)$$

FIG. 7.14. : Modèle ensembliste de T : affaiblissement

Définition 7.4.4 (Modèle ensembliste, interprétation des types et contextes)**.** On interprète chaque type A (respectivement contexte Γ) par un ensemble $[\![A]\!]_{\mathsf{Ens}}$ (respectivement $[\![\Gamma]\!]_{\mathsf{Ens}}$) défini par récurrence sur sa structure comme suit.

$$[\![-]\!]_{\mathsf{Ens}} : \mathsf{Type}_{\mathrm{T}} \to \mathbf{Ens} \qquad [\![-]\!]_{\mathsf{Ens}} : \mathsf{Con}_{\mathrm{T}} \to \mathbf{Ens}$$
$$[\![\mathtt{Nat}]\!]_{\mathsf{Ens}} = \mathbb{N} \qquad [\![\cdot]\!]_{\mathsf{Ens}} = \mathbf{1}$$
$$[\![A \to B]\!]_{\mathsf{Ens}} = [\![A]\!]_{\mathsf{Ens}} \to [\![B]\!]_{\mathsf{Ens}} \qquad [\![\Gamma, x : A]\!]_{\mathsf{Ens}} = [\![\Gamma]\!]_{\mathsf{Ens}} \times [\![A]\!]_{\mathsf{Ens}}$$

Ci-dessus, la notation $\mathbf{1}$ désigne l'ensemble à un seul élément, dénoté $()$. On dénotera v un élément quelconque d'un $[\![A]\!]_{\mathsf{Ens}}$, et E un élément quelconque d'un $[\![\Gamma]\!]_{\mathsf{Ens}}$.

Définition 7.4.5 (Modèle ensembliste, interprétation de l'extension)**.** On interprète une dérivation d'affaiblissement $\Delta \supseteq \Gamma$ par une fonction de $[\![\Delta]\!]_{\mathsf{Ens}}$ dans $[\![\Gamma]\!]_{\mathsf{Ens}}$. La figure 7.14 présente les clauses de l'interprétation.

Définition 7.4.6 (Modèle ensembliste, interprétation des termes et des substitutions)**.** On interprète une dérivation de typage $\Gamma \vdash t : A$ par une fonction de $[\![\Gamma]\!]_{\mathsf{Ens}}$ dans $[\![A]\!]_{\mathsf{Ens}}$. Similairement, on interprète une dérivation de typage $\Delta \vdash \sigma : \Gamma$ par une fonction de $[\![\Delta]\!]_{\mathsf{Ens}}$ dans $[\![\Gamma]\!]_{\mathsf{Ens}}$. La figure 7.15 présente les clauses de l'interprétation.

Remarque 7.4.1. Dans l'interprétation des variables de la figure 7.15, on a $[\![x : A]\!]_{\mathsf{Ens}} = [\![\cdot, x : A]\!]_{\mathsf{Ens}} = \mathbf{1} \times [\![A]\!]_{\mathsf{Ens}}$, et on utilise la deuxième projection $\pi_2 : \mathbf{1} \times [\![A]\!]_{\mathsf{Ens}} \to [\![A]\!]_{\mathsf{Ens}}$.

Lemme 7.4.2 (Fonctorialité)**.** $[\![\Delta \vdash t[\sigma] : A]\!]_{\mathsf{Ens}} = [\![\Gamma \vdash t : A]\!]_{\mathsf{Ens}} \circ [\![\Delta \vdash \sigma : \Gamma]\!]_{\mathsf{Ens}}$.

Démonstration. Par une induction de routine sur la dérivation de $\Gamma \vdash t : A$. □

Théorème 7.4.2 (Correction)**.** *Si* $\Gamma \vdash t \equiv_\beta u : A$ *alors* $[\![\Gamma \vdash t : A]\!]_{\mathsf{Ens}} = [\![\Gamma \vdash u : A]\!]_{\mathsf{Ens}}$.

Démonstration. Par induction sur la dérivation de $t \equiv_\beta u$. Les cas de la réflexivité, symétrie et transitivité sont immédiats par application des hypothèses d'induction, tout comme ceux correspondant à la clôture contextuelle. Les équations (7.2) et (7.3) sont

$$[\![-]\!]_{\mathsf{Ens}} : \mathcal{T}[\Sigma_{\mathrm{T}}](\Gamma; A) \to [\![\Gamma]\!]_{\mathsf{Ens}} \to [\![A]\!]_{\mathsf{Ens}}$$

$$\left[\!\!\left[\frac{\Gamma \supseteq x : A}{\Gamma \vdash x : A} \right]\!\!\right]_{\mathsf{Ens}} (E) = \pi_2([\![\Gamma \supseteq x : A]\!]_{\mathsf{Ens}}(E))$$

$$\left[\!\!\left[\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathtt{app}(t,u) : B} \right]\!\!\right]_{\mathsf{Ens}} (E) = [\![\Gamma \vdash t : A \to B]\!]_{\mathsf{Ens}}(E)(v) \quad \text{où } v = [\![\Gamma \vdash u : A]\!]_{\mathsf{Ens}}(E)$$

$$\left[\!\!\left[\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \mathtt{fun}(x.t) : A \to B} \right]\!\!\right]_{\mathsf{Ens}} (E) = v \mapsto [\![\Gamma, x : A \vdash t : B]\!]_{\mathsf{Ens}}(E, v)$$

$$\left[\!\!\left[\frac{}{\Gamma \vdash \mathtt{zero} : \mathtt{Nat}} \right]\!\!\right]_{\mathsf{Ens}} (E) = 0$$

$$\left[\!\!\left[\frac{\Gamma \vdash t : \mathtt{Nat}}{\Gamma \vdash \mathtt{suc}(t) : \mathtt{Nat}} \right]\!\!\right]_{\mathsf{Ens}} (E) = 1 + [\![\Gamma \vdash t : \mathtt{Nat}]\!]_{\mathsf{Ens}}(E)$$

$$\left[\!\!\left[\frac{\Gamma \vdash t : \mathtt{Nat} \qquad \Gamma \vdash u : A \qquad \Gamma, x : \mathtt{Nat}, y : A \vdash v : A}{\Gamma \vdash \mathtt{fold}_{\mathtt{Nat}}(t,u,x.y.v) : A} \right]\!\!\right]_{\mathsf{Ens}} (E) = \mathit{fold}_{\mathbb{N}}(f_0, f_s)([\![t]\!]_{\mathsf{Ens}}(E)) \quad \text{où } f_0 = [\![u]\!]_{\mathsf{Ens}}(E),\ f_s(m,w) = [\![v]\!]_{\mathsf{Ens}}((E,m),w)$$

$$[\![-]\!]_{\mathsf{Ens}} : \mathcal{T}[\Sigma_{\mathrm{T}}](\Delta; \Gamma) \to [\![\Delta]\!]_{\mathsf{Ens}} \to [\![\Gamma]\!]_{\mathsf{Ens}}$$

$$\left[\!\!\left[\frac{\Delta \supseteq \Gamma}{\Delta \vdash \mathtt{id} : \Gamma} \right]\!\!\right]_{\mathsf{Ens}} (E) = [\![\Delta \supseteq \Gamma]\!]_{\mathsf{Ens}}(E)$$

$$\left[\!\!\left[\frac{\Delta \vdash \sigma : \Gamma \qquad \Delta \vdash t : A}{\Delta \vdash (\sigma, x\backslash t) : (\Gamma, x : A)} \right]\!\!\right]_{\mathsf{Ens}} (E) = ([\![\sigma]\!]_{\mathsf{Ens}}(E), [\![t]\!]_{\mathsf{Ens}}(E))$$

FIG. 7.15. : Modèle ensembliste de T : termes et substitutions

conséquences immédiates de la propriété 7.4.1. Reste l'équation (7.1), pour laquelle on a

$$\begin{aligned}
&[\![\Gamma \vdash \mathtt{app}(\mathtt{fun}(x.t), u) : B]\!]_{\mathsf{Ens}}(E) \\
&= [\![\Gamma, x : A \vdash t : B]\!]_{\mathsf{Ens}}(E, [\![\Gamma \vdash u : A]\!]_{\mathsf{Ens}}(E)) && \text{par définition} \\
&= [\![\Gamma, x : A \vdash t : B]\!]_{\mathsf{Ens}}([\![\Gamma \vdash (\mathtt{id}, x\backslash u) : (\Gamma, x : A)]\!]_{\mathsf{Ens}}(E)) && \text{par définition} \\
&= [\![\Gamma \vdash t[x\backslash u] : B]\!]_{\mathsf{Ens}}(E) && \text{par le lemme 7.4.2.}
\end{aligned}$$

□

Corollaire 7.4.1. *Si* $\Delta \vdash \sigma \equiv_\beta \varphi : \Gamma$ *alors* $[\![\Delta \vdash \sigma : \Gamma]\!]_{\mathsf{Ens}} = [\![\Delta \vdash \varphi : \Gamma]\!]_{\mathsf{Ens}}$.

7.4.3.1. Adéquation du modèle ensembliste

Il reste à prouver que le modèle ensembliste associe à tout terme clos M de type $\mathtt{Nat}$ un entier $[\![t]\!]_{\mathsf{Ens}}$ tel que $\underline{[\![t]\!]_{\mathsf{Ens}}(*)} \equiv_\beta t$. Une preuve directe échoue pour les mêmes raisons que celles expliquées au début de cette section. Pour y remédier, on va généraliser notre énoncé via un outil standard appelé *relation logique*.

De façon générale, une relation logique est une relation définie par récurrence sur les types d'un langage de programmation. Ici, il s'agira de relier les éléments du modèle ensembliste et des classes d'équivalence de terme modulo β, i.e., des éléments du modèle syntaxique. Le point clef est la définition au type des fonctions, qui va être conçue pour fournir une propriété assez forte pour permettre le raisonnement par induction.

Définition 7.4.7. On définit une famille de relations $(\sqsubseteq_A \subseteq [\![A]\!]_{\mathsf{Ens}} \times [\![A]\!]_{\mathsf{Syn}})_{A \in \mathsf{Type}_\mathrm{T}}$ par récurrence sur A. On note $n \sqsubseteq_A t$ lorsque (n, t) appartient à la relation $\sqsubseteq_A$.

$$n \sqsubseteq_{\mathtt{Nat}} t \overset{def}{\iff} [\![\underline{n}]\!]_{\mathsf{Syn}} = t$$

$$f \sqsubseteq_{A\to B} t \overset{def}{\iff} \forall v \in [\![A]\!]_{\mathsf{Ens}}, \forall u \in [\![A]\!]_{\mathsf{Syn}}, v \sqsubseteq_A u \implies f(v) \sqsubseteq_B \mathtt{app}(t, u)$$

Dans la première clause ci-dessus, l'égalité $[\![\underline{n}]\!]_{\mathsf{Syn}} = t$ concerne des éléments de $[\![A]\!]_{\mathsf{Syn}}$, qui sont des classes d'équivalence de termes clos modulo équivalence β. Autrement dit, cette égalité exprime que tout terme dans la classe d'équivalence t est β-équivalent à $\underline{n}$. Dans la seconde clause, l'opération $\mathtt{app}(t, u)$ est bien définie sur les classes d'équivalence puisque l'équivalence β est une T-congruence.

On écrira qu'un élément v de $[\![A]\!]_{\mathsf{Ens}}$ *implémente* un terme clos t de type A modulo β lorsque $v \sqsubseteq_A t$. Cette relation s'étend aux environnements liaison-à-liaison comme suit.

Définition 7.4.8. On définit une famille de relations $(\sqsubseteq_\Gamma \subseteq [\![\Gamma]\!]_{\mathsf{Ens}} \times [\![\Gamma]\!]_{\mathsf{Syn}})_{\Gamma \in \mathsf{Con}_\mathrm{T}}$ par récurrence sur Γ.

$$() \sqsubseteq_\cdot \mathtt{id}$$

$$(E, v) \sqsubseteq_{\Gamma, x:A} \sigma, x\backslash t \overset{def}{\iff} E \sqsubseteq_\Gamma \sigma \text{ et } v \sqsubseteq_A M$$

Lemme 7.4.3. *Si* $E \sqsubseteq_\Delta \delta$ *alors* $[\![\Delta \supseteq \Gamma]\!]_{\mathsf{Ens}}(E) \sqsubseteq_\Gamma \delta|_\Gamma$, *où* $\delta|_\Gamma$ *désigne la restriction de* δ *aux liaisons dont la variable apparaît dans* Γ.

Définition 7.4.9. On écrira que t *réalise* A *sous* Γ, et on notera $\Gamma \models t : A$, lorsque pour tous $E \in [\![\Gamma]\!]_{\mathsf{Ens}}$ et $\sigma \in [\![\Gamma]\!]_{\mathsf{Syn}}$ tels que $E \sqsubseteq_\Gamma \sigma$ on a $[\![t]\!]_{\mathsf{Ens}}(E) \sqsubseteq_A [\![t]\!]_{\mathsf{Syn}}[\sigma]$.

Le lemme qui suit est le résultat principal de cette section, duquel découlent plusieurs corollaires très utiles. C'est un énoncé typique lorsqu'on travaille avec une relation logique : il exprime que les termes bien typés respectent la relation.

Lemme 7.4.4 (Adéquation)**.** *Si* $\Gamma \vdash t : A$ *alors* $\Gamma \models M : A$.

Démonstration. Par induction sur la dérivation de typage.

- Cas $\dfrac{\Gamma \supseteq x : A}{\Gamma \vdash x : A}$: par le lemme 7.4.3 on a $\pi_2([\![\Gamma \supseteq x : A]\!]_{\mathsf{Ens}}(E)) \sqsubseteq_A x[\delta|_{x:A}]$. On conclut immédiatement puisque $x[\delta|_{x:A}] = x[\delta]$.

- Cas $\dfrac{\Gamma \vdash t_1 : A \to B \qquad \Gamma \vdash t_2 : A}{\Gamma \vdash \mathtt{app}(t_1, t_2) : B}$: on a $E \in [\![\Gamma]\!]_{\mathsf{Ens}}$ et $\gamma \in [\![\Gamma]\!]_{\mathsf{Syn}}$ tels que $E \sqsubseteq_\Gamma \gamma$. On doit montrer

$$\begin{aligned} & [\![\mathtt{app}(t_1, t_2)]\!]_{\mathsf{Ens}}(E) \sqsubseteq_B [\![\mathtt{app}(t_1, t_2)]\!]_{\mathsf{Syn}}[\gamma] \\ \Leftrightarrow\ & [\![t_1]\!]_{\mathsf{Ens}}(E)([\![t_2]\!]_{\mathsf{Ens}}(E)) \sqsubseteq_B \mathtt{app}([\![t_1]\!]_{\mathsf{Syn}}[\gamma], [\![t_2]\!]_{\mathsf{Syn}}[\gamma]). \end{aligned}$$

  Par l'hypothèse d'induction sur M_2, on sait que $[\![t_2]\!]_{\mathsf{Ens}}(E) \sqsubseteq_A [\![t_2]\!]_{\mathsf{Syn}}[\gamma]$. On conclut immédiatement par l'hypothèse d'induction sur M_1 et la définition de $\sqsubseteq_{A\to B}$.

- Cas $\dfrac{\Gamma, x : A \vdash t_1 : B}{\Gamma \vdash \mathtt{fun}(x.t_1) : A \to B}$: on a $E \in [\![\Gamma]\!]_{\mathsf{Ens}}$ et $\gamma \in [\![\Gamma]\!]_{\mathsf{Syn}}$ tels que $E \sqsubseteq_\Gamma \gamma$. Soit $v \in [\![A]\!]_{\mathsf{Ens}}$ et $N \in [\![A]\!]_{\mathsf{Syn}}$ tels que $v \sqsubseteq_A N$. On doit montrer

$$\begin{aligned} & [\![\mathtt{fun}(x.t_1)]\!]_{\mathsf{Ens}}(E)(v) \sqsubseteq_B \mathtt{app}([\![\mathtt{fun}(x.t_1)]\!]_{\mathsf{Syn}}[\gamma], u) \\ \Leftrightarrow\ & [\![t_1]\!]_{\mathsf{Ens}}(E, v) \sqsubseteq_B \mathtt{app}(\mathtt{fun}(x.[\![t_1]\!]_{\mathsf{Syn}}[\gamma, x\backslash x]), u) \\ \Leftrightarrow\ & [\![t_1]\!]_{\mathsf{Ens}}(E, v) \sqsubseteq_B [\![t_1]\!]_{\mathsf{Syn}}[\gamma, x\backslash u]. \end{aligned}$$

  Comme $(E, v) \sqsubseteq_{\Gamma, x:A} \gamma, x\backslash u$ on conclut par l'hypothèse d'induction sur t_1.

Les cas du zéro, du successeur et de la récursion primitive sont gérés similairement en utilisant la propriété 7.4.1. □

Corollaire 7.4.2 (Adéquation)**.** *Si* $[\![t : \mathtt{Nat}]\!]_{\mathsf{Ens}} = [\![u : \mathtt{Nat}]\!]_{\mathsf{Ens}}$ *alors* $t \equiv_\beta u$.

Démonstration. Le lemme 7.4.4 et la définition de la relation logique au type $\mathtt{Nat}$ implique en particulier que n'importe quel terme clos v de ce type vérifie $[\![v]\!]_{\mathsf{Ens}}() \equiv_\beta v[\mathtt{id}] = v$. On a donc $t \equiv_\beta [\![t]\!]_{\mathsf{Ens}}() = [\![u]\!]_{\mathsf{Ens}}() \equiv_\beta u$. □

Corollaire 7.4.3 (Injectivité des constructeurs, termes clos)**.** *Si* $\mathtt{suc}(t) \equiv_\beta \mathtt{suc}(u) : \mathtt{Nat}$ *alors* $t \equiv_\beta u : \mathtt{Nat}$. *De plus,* $\mathtt{suc}(t) \not\equiv_\beta \mathtt{zero} : \mathtt{Nat}$.

Démonstration. Pour la première propriété, observons que $1 + [\![t]\!]_{\mathsf{Ens}} = 1 + [\![u]\!]_{\mathsf{Ens}}$, et donc $[\![t]\!]_{\mathsf{Ens}} = [\![u]\!]_{\mathsf{Ens}}$ par injectivité du successeur. On conclut par le corollaire 7.4.2. La deuxième propriété se démontre de façon analogue. □

Corollaire 7.4.4. *Si $\underline{i} \equiv_\beta \underline{j}$ alors $i = j$.*

Démonstration. Par induction sur i puis par cas sur j, en utilisant le corollaire 7.4.3. □

7.4.3.2. Applications du modèle ensembliste

Théorème 7.4.3 (Canonicité)**.** *Si $t : \mathtt{Nat}$ alors il existe un unique entier k tel que*

$$t \equiv_\beta \underline{k}.$$

Démonstration. L'existence découle directement du lemme 7.4.4 et de la définition de la relation logique, en prenant $k \coloneqq [\![t : \mathtt{Nat}]\!]_{\mathsf{Ens}}$. Pour l'unicité, pour tout i tel que $t \equiv_\beta \underline{i}$, on a $\underline{i} \equiv_\beta \underline{k}$ et donc $i = k$ par le corollaire 7.4.4. □

7.4.3.3. Remarques au sujet du modèle ensembliste

On termine cette section par deux remarques au sujet du modèle ensembliste, l'une positive et l'autre critique.

Extensionalité. Le modèle ensembliste fournit un outil utile à l'étude de l'équivalence β. En particulier, on a vu que lorsque deux termes clos de type $\mathtt{Nat}$ ont la même interprétation, alors ils sont équivalents modulo β. Cependant, ce résultat ne s'étend pas au delà du type $\mathtt{Nat}$.

Considérons par exemple les termes clos de type $(\mathtt{Nat} \to \mathtt{Nat}) \to \mathtt{Nat} \to \mathtt{Nat}$ que sont

$$t_1 = \mathtt{fun}(f.f) \qquad \text{et} \qquad t_2 = \mathtt{fun}(f.\mathtt{fun}(x.\mathtt{app}(f, x))).$$

En calculant leurs interprétations respectives dans le modèle ensembliste, on trouve

$$[\![t_1]\!]_{\mathsf{Ens}}() = f \mapsto f \qquad \text{et} \qquad [\![t_2]\!]_{\mathsf{Ens}}() = f \mapsto x \mapsto f(x)$$

qui sont identiques, puisque deux fonctions mathématiques sont égales si et seulement si elles envoient les mêmes entrées dans les mêmes sorties. Pour autant, t_1 et t_2 ne sont pas équivalents modulo β, puisqu'ils sont tous les deux $\to_\beta$-normaux.

Plutôt que de considérer ce décalage entre T et le modèle ensembliste comme un défaut de ce dernier, on peut y voir le signe d'un manque dans la théorie équationnelle choisie. Une façon d'y remédier serait d'ajouter une règle d'équivalence qui stipule que si t est un terme de type $A \to B$, alors t devrait être considéré comme équivalent à $\mathtt{fun}(x.\mathtt{app}(t, x))$. Ce principe appartient à une famille d'équations appelées *équivalences η*, ou encore *équivalence extensionnelle*.

Définissabilité. On attend d'un modèle qu'il nous éclaire sur la structure du langage interprété en explicitant certains phénomènes difficilement visibles au niveau de la syntaxe. C'est pourquoi le modèle syntaxique est peu intéressant. En échange, celui-ci est très économe au sens où l'interprétation d'un type ne contient pas d'éléments extérieurs à la syntaxe.

Définition 7.4.10. Soit $[\![-]\!]$ la fonction d'interprétation d'un modèle quelconque de T et A un type. Un élément v de $[\![A]\!]$ est *définissable* s'il est dans l'image de la fonction d'interprétation, autrement dit s'il existe un terme $\vdash t : A$ tel que $v = [\![\cdot \vdash t : A]\!]$.

Il est évident que tous types du modèle syntaxique ne contiennent que des éléments définissables. Cette propriété n'est pas vraie du modèle ensembliste.

Propriété 7.4.2. *La fonction $F \in (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ telle que*

$$F(f) = \begin{cases} 1 & \textit{si } f \textit{ est la fonction constante 0} \\ 0 & \textit{sinon} \end{cases}$$

n'est pas définissable.

Très informellement, F n'est pas définissable dans T, ni dans aucun autre langage de programmation, puisqu'une hypothétique implémentation devrait tester son entrée sur un nombre infini d'arguments avant de répondre oui ou non. On introduira au chapitre suivant les outils techniques qui permettent de transformer cette intuition en raisonnement rigoureux.

Remarque 7.4.2. L'échec de la définissabilité n'est pas restreint aux fonctions d'ordre supérieur. Le lecteur ou la lectrice qui connaît un peu de théorie de la calculabilité sait que la fonction $g : \mathbb{N} \to \mathbb{N}$ telle que

$$g(n) = \begin{cases} 1 & \text{si } n \text{ est le code d'une machine de Turing qui termine sur toute entrée} \\ 0 & \text{sinon} \end{cases}$$

ne peut pas non plus être définissable.

Exercices

* **Ex. 31** — Reformuler la β-équivalence comme un jugement inductif.

* **Ex. 32** — Démontrer que $\Gamma \vdash M' : A$ et $M \to_\beta M'$ n'entraîne pas $\Gamma \vdash M : A$.

* **Ex. 33** — Donner un exemple d'un système de réécriture abstrait qui soit confluent et normalisant sans être fortement normalisant.

* **Ex. 34** — Donner un exemple d'un système de réécriture abstrait qui soit confluent et normalisant sans avoir l'unicité des formes normales.

* Exercice 35 Mesures

La relation de *sous-terme immédiat*, notée $M <_i N$, est définie inductivement par les sept axiomes exprimés concisément ci-dessous.

$$\begin{aligned} M, N &<_i \mathtt{app}(M, N) \\ M &<_i \mathtt{fun}(x.M), \mathtt{suc}(M) \\ M, N, P &<_i \mathtt{fold}_{\mathtt{Nat}}(M, N, x.y.P) \end{aligned}$$

La relation de *sous-terme*, notée $M < N$, est la clôture transitive de la relation de sous-terme immédiat. Une *mesure* est une fonction $d : \mathsf{PreTerm}_{\mathrm{T}} \to \mathbb{N}$ telle que pour tous termes M et N tels que $M < N$, on ait $d(M) < d(N)$.

1. Démontrer que la fonction sz est une mesure.
2. Donner un autre exemple de mesure que la fonction sz.
3. Démontrer que la relation de sous-terme est irréflexive : $M \not< M$ pour tout M.
4. Soit $\mathbf{P}$ un ensemble de prétermes qui satisfait la propriété suivante.

$$\frac{\forall N, N < M \Rightarrow N \in \mathbf{P}}{M \in \mathbf{P}}$$

   Démontrer que $\mathbf{P}$ contient tous les prétermes.
5. Soit d une mesure. Un ensemble de prétermes $\mathbf{P}$ est dit *d-inductif* s'il satisfait la propriété suivante.

$$\frac{\forall N, (\forall M' <_i M, d(N) \leq d(M')) \Rightarrow N \in \mathbf{P}}{M \in \mathbf{P}}$$

   Démontrer que tout ensemble d-inductif qui contient les variables et $\mathtt{zero}$ contient tous les prétermes.

Dans les deux exercices qui suivent, on s'intéresse uniquement au sous-ensemble $\mathsf{PreTerm}_\lambda$ de $\mathsf{PreTerm}_\mathrm{T}$ qui correspond aux termes du langage appelé *λ-calcul pur*. Il s'agit du fragment du système T sans les constructions permettant de manipuler les entiers naturels, à savoir zéro, successeur et récursion primitive.

$$\mathsf{PreTerm}_\lambda \ni M, N, P ::= \mathtt{var}(x) \mid \mathtt{fun}(x.M) \mid \mathtt{app}(M, N)$$

L'extension des résultats de ces exercices au système T tout entier ne pose aucune difficulté, mais ne demande aucune idée supplémentaire.

** Exercice 36 Syntaxe anonyme et indices de de Bruijn

La définition de l'α-équivalence donnée à la définition 6.2.8 peut être vue comme celle d'une fonction récursive qui parcourt deux termes et décide si oui ou non ils sont α-équivalents. Cette fonction est soit coûteuse en temps ou en espace, puisqu'elle exige, lors de la vérification de $\mathtt{fun}(x_1.M_1) \equiv_\alpha \mathtt{fun}(x_2.M_2)$, soit de parcourir M_1 et M_2 pour calculer l'ensemble de leurs variables libres, soit de stocker ces deux ensembles dans la représentation du terme.

Le but de cet exercice est de remplacer cette définition par une représentation où les occurrences de variables ne sont plus associées à des noms mais à des entiers qui désigne le lieur pertinent. Une telle approche est dite *anonyme* et permet de se dispenser totalement de la notion d'α-équivalence, par opposition à la représentation avec nom qui est dite *nominale*. Les *termes anonymes*, dûs au mathématicien hollandais Nicolaas Govert de Bruijn, obéissent à la grammaire suivante.

$$\mathsf{ATerm}_\lambda \ni M, N, P ::= \mathtt{var}^{\mathsf{db}}(n) \mid \mathtt{app}^{\mathsf{db}}(M, N) \mid \mathtt{fun}^{\mathsf{db}}(M) \qquad (n \in \mathbb{N})$$

L'entier 0 dans $\mathtt{var}^{\mathsf{db}}(0)$ signifie que cette occurence de variable désigne la variable liée par le dernier lieur au dessus de cette occurence. L'entier 1 dans $\mathtt{var}^{\mathsf{db}}(1)$ désigne l'avant-dernier lieur, et ainsi de suite. Par exemple, le terme $\mathtt{fun}^{\mathsf{db}}(\mathtt{var}^{\mathsf{db}}(0))$ correspond au terme $\mathtt{fun}(x.x)$ dans la syntaxe avec noms. On appelle ces entiers des *indices de de Bruijn*. Les occurences libres sont représentées par des indices supérieurs au nombre de lieurs englobants ; par exemple $\mathtt{fun}^{\mathsf{db}}(\mathtt{app}^{\mathsf{db}}(\mathtt{var}^{\mathsf{db}}(0), \mathtt{var}^{\mathsf{db}}(1)))$ a une variable libre.

Il est commode de définir un jugement inductif $k \vdash M$ qui exprime que le terme anonyme M dispose d'*au plus* k variables libres.

$$\textsc{dBVar}\ \frac{n < k}{k \vdash \mathtt{var}^{\mathsf{db}}(n)} \qquad \textsc{dBApp}\ \frac{k \vdash M \qquad k \vdash N}{k \vdash \mathtt{app}^{\mathsf{db}}(M, N)} \qquad \textsc{dBAbs}\ \frac{k+1 \vdash M}{k \vdash \mathtt{fun}^{\mathsf{db}}(M)}$$

Le but de l'exercice est de définir deux fonctions

$$\mathsf{name} : \mathsf{Term}^0_\lambda \to \mathsf{ATerm}^0_\lambda \qquad \text{et} \qquad \mathsf{db} : \mathsf{ATerm}^0_\lambda \to \mathsf{Term}^0_\lambda,$$

où Term^0_λ et ATerm^0_λ désignent les termes clos nominaux et anonymes, et de montrer que ces deux fonctions soient inverses l'une de l'autre.

1. Donner le terme anonyme qui correspond chacun des termes avec noms suivants.

   $\texttt{fun}(x.\texttt{fun}(y.\texttt{app}(x,y)))$ $\quad$ $\texttt{fun}(y.\texttt{fun}(x.\texttt{app}(y,x)))$ $\quad$ $\texttt{fun}(y.\texttt{fun}(x.\texttt{app}(x,y)))$

2. Démontrer que si $k \vdash M$ alors $k+1 \vdash M$.
3. Définir une fonction

   $$\mathsf{name} : \mathsf{ATerm}_\lambda \times \mathbb{A}^* \to \mathsf{PreTerm}_\lambda$$

   telle que si $\mathsf{db}(M, \varepsilon) = \mathsf{db}(M)$ pour tout M clos, où $\mathbb{A}^*$ désigne l'ensemble des listes finies de noms.
4. Définir une fonction

   $$\mathsf{db} : \mathsf{PreTerm}_\lambda \times (\mathbb{A} \to \mathbb{N}) \to \mathsf{ATerm}_\lambda$$

   telle que $\mathsf{db}(M, (x \mapsto 0)) = \mathsf{db}(M)$ pour tout M clos, où la notation $x \mapsto 0$ désigne la fonction constante 0.
5. Démontrer que pour tous M et N clos, si $M \equiv_\alpha N$ alors $\mathsf{db}(M) = \mathsf{db}(N)$.
6. Démontrer que pour tout M clos, $\mathsf{name}(\mathsf{db}(M)) \equiv_\alpha M$.
7. En déduire que pour tous termes nominaux clos M et N, on a $M \equiv_\alpha N$ si et seulement si $\mathsf{db}(M) = \mathsf{db}(N)$.

* Exercice 37 Programmation en T

1. Définir les termes clos $M_+, M_\times, M_{pow} : \texttt{Nat} \to \texttt{Nat} \to \texttt{Nat}$ tels que, pour tout couple d'entiers p, q, on ait

   $$M_+ \ \underline{p}\ \underline{q} \equiv_\beta \underline{p+q}$$
   $$M_\times \ \underline{p}\ \underline{q} \equiv_\beta \underline{pq}$$
   $$M_{pow} \ \underline{p}\ \underline{q} \equiv_\beta \underline{q^p}.$$

2. Définir un terme clos $M_- : \texttt{Nat} \to \texttt{Nat} \to \texttt{Nat}$ tel que, pour tout couple d'entiers p, q, on ait

   $$M_- \ \underline{p}\ \underline{q} \equiv_\beta \underline{p-q}.$$

3. Définir un terme M_{ack} tel que, pour tout couple d'entiers p, q, on ait

   $$M_{ack} \ \underline{p}\ \underline{q} \equiv_\beta \underline{A(p,q)}$$

   où A désigne la fonction d'Ackermann, définie mathématiquement comme suit.

   $$A(0,q) = q+1$$
   $$A(1+p,0) = A(p,1)$$
   $$A(1+p,1+q) = A(p, A(1+p,q))$$

* Exercice 38 Une traduction de T en OCaml

Proposer une traduction de T en OCaml. Cette traduction doit prendre la forme de deux fonctions mathématiques, toutes deux notées $[\![-]\!]_{\mathsf{OCaml}}$. La première associe à tout type A de T un type $[\![A]\!]_{\mathsf{OCaml}}$ d'OCaml. La seconde associe à tout terme $x_1 : B_1, \ldots, x_n : B_n \vdash M : A$ de T une expression $[\![M]\!]_{\mathsf{OCaml}}$ écrite en OCaml tel que

$$x_1 : [\![B_1]\!]_{\mathsf{OCaml}}, \ldots, x_n : [\![B_n]\!]_{\mathsf{OCaml}} \vdash [\![M]\!]_{\mathsf{OCaml}} : [\![A]\!]_{\mathsf{OCaml}}.$$

Votre traduction peut utiliser du code OCaml prédéfini que vous prendrez soin de décrire. De plus, votre traduction est soumise aux contraintes qui suivent.

- Elle doit être *correcte* : si $M \equiv_\beta \underline{n}$ alors l'expression $[\![M]\!]_{\mathsf{OCaml}}$ doit s'évaluer en une représentation de l'entier n en OCaml. La représentation exacte de n dépendra de votre traduction du type `Nat`.
- Elle doit être *fonctorielle* : la traduction d'un terme est définie uniformément à partir de celle de ses sous-termes. Autrement dit, on doit avoir

$$[\![M[x\backslash N]]\!]_{\mathsf{OCaml}} = [\![M]\!]_{\mathsf{OCaml}}[x\backslash [\![N]\!]_{\mathsf{OCaml}}].$$

- Elle doit terminer en temps *linéaire* en la taille du terme. Cette contrainte exprime que votre traduction correspond à un compilateur plutôt qu'à un interprète.

* Exercice 39 Extension de T avec booléens

Dans cet exercice, on se propose d'étendre T avec des booléens. On ajoute à la syntaxe les deux formes d'introduction et la forme d'élimination que vous connaissez bien.

$\mathsf{PreTerm}_{\mathrm{T}} \ni M, N, P ::=$		**Prétermes**
	$\mid \ldots$	
	$\mid$ `true`	Vrai
	$\mid$ `false`	Faux
	$\mid$ `if`(M, N, P)	Conditionnelle
$\mathsf{Type}_{\mathrm{T}} \ni A, B, C ::=$		**Types**
	$\mid \ldots$	
	$\mid$ `Bool`	Type booléen

1. Proposer les équation β exprimant la simplification de la conditionnelle lorsque son premier sous-terme est une forme d'introduction booléenne.
2. Proposer des règles de typage des nouvelles constructions. Étendre la preuve du théorème 7.2.1 aux nouvelles règles.
3. Définir une fonction de traduction $(\!|-|\!)$ de votre extension de T dans le langage de départ. Elle doit respecter les mêmes contraintes que celle de l'exercice 38.

4. Démontrer que votre traduction préserve l'équivalence β.
5. Étendre le modèle ensembliste pour traiter les nouvelles constructions.

* Exercice 40 Extension de T avec produits cartésiens

Dans cet exercice, on se propose d'étendre T avec des produits cartésiens, c'est-à-dire des paires. On ajoute à la syntaxe les formes d'introduction et d'élimination suivantes.

$\mathsf{PreTerm}_{\mathrm{T}} \ni M, N, P ::=$		**Prétermes**
	$\mid \ldots$	
	$\mid \mathtt{pair}(M, N)$	Paires
	$\mid \mathtt{fst}(M)$	Première projection
	$\mid \mathtt{snd}(M)$	Deuxième projection
$\mathsf{Type}_{\mathrm{T}} \ni A, B, C ::=$		**Types**
	$\mid \ldots$	
	$\mid A \times B$	Type produit

1. Proposer les équation β exprimant la simplification des projections lorsque leur sous-terme est une forme d'introduction booléenne.
2. Proposer des règles de typage des nouvelles constructions. Étendre la preuve du théorème 7.2.1 aux nouvelles règles.
3. Étendre le modèle ensembliste pour traiter les nouvelles constructions. Étendre la relation logique et la preuve du lemme 7.4.4 pour en déduire un théorème de canonicité étendu.

8. Le langage PCF

Dans ce chapitre, on s'intéresse à un langage fonctionnel toujours très idéalisé mais plus proche des langages de programmation réels comme Haskell. Ce langage, PCF, pour *Programming with Computation Functions*, remplace la récursion primitive de T par la récursion générale, c'est-à-dire la possibilité de définir des fonctions récursives arbitraires. En échange d'une facilité de programmation accrue, on perd la normalisation : les termes bien typés de PCF peuvent diverger.

8.1. Syntaxe et sémantique opérationnelle

8.1.1. Syntaxe

On passe rapidement sur les définitions élémentaire de la syntaxe de $\mathcal{PCF}$ dans cette section dans la mesure où leur traitement est parfaitement identique à celui développé pour T en section 7.1. En particulier, tous les termes sont immédiatement considérés à renommage des variables liées près, et on ne détaille pas le traitement de la substitution.

Définition 8.1.1. L'ensemble $\mathsf{Term}_{\mathcal{PCF}}$ des *termes* et l'ensemble $\mathsf{Sub}_{\mathcal{PCF}}$ des *substitutions* de $\mathcal{PCF}$ sont définis inductivement par les grammaires présentées à la figure 8.1.

Les termes de $\mathcal{PCF}$ sont donc les mêmes que ceux de T, à ceci près que la récursion primitive a été remplacée par la récursion générale $\texttt{fix}(t)$ et une construction de test. De plus, tout entier naturel n est représenté dans la syntaxe comme le terme $\texttt{lit}(n)$, ce qui va faciliter l'expression de la réduction. On abrègera parfois $\texttt{lit}(n)$ en n lorsqu'il est clair qu'on souhaite parler d'un terme de $\mathcal{PCF}$ plutôt que d'un entier naturel.

L'ensemble des types de PCF est identique à celui du système T, de même pour l'ensemble des contextes de typage. Toutefois, on les dénote $\mathsf{Type}_{\mathcal{PCF}}$ et $\mathsf{Con}_{\mathcal{PCF}}$ dans ce chapitre, pour insister sur le fait qu'on étudie $\mathcal{PCF}$ plutôt que T. Le jugement d'affaiblissement de $\mathcal{PCF}$ est hérité de T puisque $\mathsf{Con}_{\mathcal{PCF}}$ et $\mathsf{Con}_{\mathrm{T}}$ sont identiques.

Définition 8.1.2 (Typage de $\mathcal{PCF}$)**.** Les jugements de typage des termes et de typage des substitutions sont définis inductivement à la figure 8.2.

On écrit $\mathcal{PCF}(\Gamma; A)$ pour l'ensemble des termes habitant le type A sous le contexte de typage Γ, et on abrège $\cdot \vdash t : A$ en $M : A$.

Lemme 8.1.1 (Substitution)**.** *Si* $\Gamma \vdash t : A$ *et* $\Delta \vdash \sigma : \Gamma$ *alors* $\Delta \vdash t[\sigma] : A$.

Démonstration. Induction de routine sur la dérivation de typage. □

$\mathsf{Term}_{\mathcal{PCF}} \ni M, N, P ::=$		**Termes**
	$\mid \mathtt{var}(x)$	Variable
	$\mid \mathtt{fun}(x.t)$	Abstraction
	$\mid \mathtt{app}(t, u)$	Application
	$\mid \mathtt{lit}(n)$	Entier littéral ($n \in \mathbb{N}$)
	$\mid \mathtt{suc}(t)$	Successeur
	$\mid \mathtt{match}_{\mathtt{Nat}}(t, u, x.v)$	Conditionnelle
	$\mid \mathtt{fix}(t)$	Récursion générale
$\mathsf{Sub}_{\mathcal{PCF}} \ni \sigma, \varphi ::=$		**Substitutions**
	$\mid \mathtt{id}$	Substitution identique
	$\mid \sigma, x\backslash t$	Liaison

FIG. 8.1. : Syntaxe de $\mathcal{PCF}$

$$\boxed{\Gamma \vdash t : A} \qquad \frac{\Gamma \supseteq x : A}{\Gamma \vdash \mathtt{var}(x) : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \mathtt{fun}(x.t) : A \to B}$$

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathtt{app}(t, u) : B} \qquad \frac{}{\Gamma \vdash \mathtt{lit}(n) : \mathtt{Nat}} \qquad \frac{\Gamma \vdash t : \mathtt{Nat}}{\Gamma \vdash \mathtt{suc}(t) : \mathtt{Nat}}$$

$$\frac{\Gamma \vdash t : \mathtt{Nat} \qquad \Gamma \vdash u : A \qquad \Gamma, x : \mathtt{Nat} \vdash v : A}{\Gamma \vdash \mathtt{match}_{\mathtt{Nat}}(t, u, x.v) : A} \qquad \frac{\Gamma \vdash t : A \to A}{\Gamma \vdash \mathtt{fix}(t) : A}$$

$$\boxed{\Gamma \vdash \sigma : \Delta} \qquad \frac{\Delta \supseteq \Gamma}{\Delta \vdash \mathtt{id} : \Gamma} \qquad \frac{\Gamma \vdash \sigma : \Delta \qquad \Gamma \vdash t : A}{\Gamma \vdash \sigma, x\backslash t : \Delta, x : A}$$

FIG. 8.2. : Typage des termes et substitutions de $\mathcal{PCF}$

$$\frac{(M_1,\ldots,M_n)\in R}{(M_1,\ldots,M_n)\in\mathcal{PCF}[R]} \qquad \frac{(M_1,\ldots,M_n)\in\mathcal{PCF}[R] \qquad (N_1,\ldots,N_n)\in\mathcal{PCF}[R]}{(\mathtt{app}(t_1,N_1),\ldots,\mathtt{app}(t_n,N_n))\in\mathcal{PCF}[R]}$$

$$\frac{(M_1,\ldots,M_n)\in\mathcal{PCF}[R]}{(\mathtt{fun}(x.t_1),\ldots,\mathtt{fun}(x.t_n))\in\mathcal{PCF}[R]} \qquad \frac{(M_1,\ldots,M_n)\in\mathcal{PCF}[R]}{(\mathtt{suc}(t_1),\ldots,\mathtt{suc}(t_n))\in\mathcal{PCF}[R]}$$

$$\frac{(M_1,\ldots,M_n)\in\mathcal{PCF}[R] \qquad (N_1,\ldots,N_n)\in\mathcal{PCF}[R] \qquad (P_1,\ldots,P_n)\in\mathcal{PCF}[R]}{(\mathtt{match}_{\mathtt{Nat}}(t_1,N_1,x.P_1),\ldots,\mathtt{match}_{\mathtt{Nat}}(t_n,N_n,x.P_n))\in\mathcal{PCF}[R]}$$

$$\frac{(M_1,\ldots,M_n)\in\mathcal{PCF}[R]}{(\mathtt{fix}(t_1),\ldots,\mathtt{fix}(t_n))\in\mathcal{PCF}[R]}$$

FIG. 8.3. : Clôture $\mathcal{PCF}$-contextuelle d'une relation R

8.1.2. Clôture contextuelle

Définition 8.1.3 (Clôture $\mathcal{PCF}$-contextuelle d'une relation)**.** Si R est une relation n-aire sur les termes de $\mathcal{PCF}$, sa *clôture $\mathcal{PCF}$-contextuelle*, notée $\mathcal{PCF}[R]$, est la relation n-aire sur les termes de $\mathcal{PCF}$ définie inductivement par les règles donnée à la figure 8.3.

Définition 8.1.4 ($\mathcal{PCF}$-congruence)**.** Une *$\mathcal{PCF}$-congruence* R est une relation d'équivalence sur les termes de $\mathcal{PCF}$ telle que $\mathcal{PCF}[R]$ soit incluse dans R.

On va maintenant démontrer quelques propriétés utiles concernant la clôture $\mathcal{PCF}$-contextuelle vue comme une opération sur les relations. Dans ce qui suit, R et S désignent des relations quelconques de l'ensemble $\mathsf{Term}_{\mathcal{PCF}}$ dans lui-même.

Propriété 8.1.1. $\mathcal{PCF}[Id_{\mathsf{Term}_{\mathcal{PCF}}}] = Id_{\mathsf{Term}_{\mathcal{PCF}}}$ *et* $\mathcal{PCF}[R;S] \subseteq \mathcal{PCF}[R];\mathcal{PCF}[S]$.

Propriété 8.1.2. *Si* $R \subseteq S$ *alors* $\mathcal{PCF}[R] \subseteq \mathcal{PCF}[S]$.

Propriété 8.1.3. $R \subseteq \mathcal{PCF}[R]$ *et* $\mathcal{PCF}[\mathcal{PCF}[R]] = \mathcal{PCF}[R]$.

Propriété 8.1.4. $\mathcal{PCF}[R^\circ] = \mathcal{PCF}[R]^\circ$.

Propriété 8.1.5. *La clôture contextuelle d'une relation réflexive (respesctivement symétrique, transitive) est réflexive (respectivement symétrique, transitive).*

Lemme 8.1.2. *La plus petite $\mathcal{PCF}$-congruence contenant R est $\mathcal{PCF}[R^{s*}]$.*

Démonstration. On sait que R^{s*} est une relation d'équivalence par la propriété 2.5.1. La propriété 8.1.5 implique que $\mathcal{PCF}[R^{s*}]$ est également une relation d'équivalence, c'est donc une T-congruence.

Supposons maintenant que S est une $\mathcal{PCF}$-congruence qui contient R. On veut montrer qu'elle contient aussi $\mathcal{PCF}[R^{s*}]$. On a $\mathcal{PCF}[R^{s*}] \subseteq \mathcal{PCF}[S^{s*}] \subseteq \mathcal{PCF}[S^*] \subseteq \mathcal{PCF}[S] \subseteq S$ où la première étape découle de la propriété 8.1.2 et les autres du fait que S est une $\mathcal{PCF}$-congruence. □

$\mathsf{Kon}_{\mathcal{PCF}} \ni K ::=$		**Contextes généraux**
	$\mid \diamond$	Trou
	$\mid \mathtt{fun}(x.K)$	Abstraction
	$\mid \mathtt{app}(K, u)$	
	$\mid \mathtt{app}(t, K)$	Application
	$\mid \mathtt{suc}(K)$	Successeur
	$\mid \mathtt{match}_{\mathtt{Nat}}(K, u, x.v)$	
	$\mid \mathtt{match}_{\mathtt{Nat}}(t, K, x.v)$	
	$\mid \mathtt{match}_{\mathtt{Nat}}(t, u, x.K)$	Conditionnelle
	$\mid \mathtt{fix}(K)$	Récursion générale
$\mathsf{WHKon}_{\mathcal{PCF}} \ni E ::=$		**Contextes de tête faibles**
	$\mid \diamond$	Trou
	$\mid \mathtt{app}(E, u)$	Application
	$\mid \mathtt{suc}(E)$	Successeur
	$\mid \mathtt{match}_{\mathtt{Nat}}(E, u, x.v)$	Conditionnelle

FIG. 8.4. : Contextes d'évaluation de $\mathcal{PCF}$

8.1.3. Contextes d'évaluation

On a vu au chapitre précédent deux relations permettant de réduire les termes de T par réécriture progressive, la réduction β et la réduction de tête faible. La première permet la réduction β atomique d'un sous-terme arbitraire. La seconde restreint fortement la première en interdisant par exemple de réduire dans un sous-terme argument ou sous une abstraction. Plus généralement, chaque ensemble de choix de sous-termes où la réduction atomique est autorisée donne lieu à une relation de réduction.

Cette observation suggère d'approcher la réécriture de termes directement via l'ensemble des sous-termes où la réduction β atomique est autorisée. Une formalisation intuitive et commode de cette notion est la notion de *contexte d'évaluation*. Les contextes d'évaluation, qui ne doivent pas être confondus avec les contextes de typage, sont des termes équipés d'un unique sous-terme distingué. L'emplacement de ce sous-terme est marqué par le symbole $\diamond$, qu'on appelle souvent le "trou", de façon imagée.

Définition 8.1.5 (Contextes d'évaluation)**.** L'ensemble $\mathsf{Kon}_{\mathcal{PCF}}$ des *contextes généraux* est défini inductivement par la grammaire présentée à la figure 8.4, tout comme le sous-ensemble $\mathsf{WHKon}_{\mathcal{PCF}} \subseteq \mathsf{Kon}_{\mathcal{PCF}}$ des *contextes de tête faible*.

Définition 8.1.6. Étant donné un contexte général K et un terme M, on note $K\{t\}$ pour le terme obtenu en substituant l'unique trou de K par M. On écrit que K est *de type A sous Γ avec un trou de type $\Delta \vdash B$*, et on note $\diamond : (\Delta \vdash B); \Gamma \vdash K : A$, lorsque

pour tout terme M tel que $\Delta \vdash t : B$, on a $\Gamma \vdash K\{t\} : A$.

Lemme 8.1.3. *Si $\Gamma \vdash K\{t\} : A$ alors il existe Δ et B tels que $\Delta \vdash t : B$ et $\diamond : (\Delta \vdash B); \Gamma \vdash K : A$.*

Démonstration. Par une induction de routine sur K. □

Tout contexte de tête faible est un contexte général, on s'autorisera donc à écrire $E\{t\}$ et $\diamond : (\Delta \vdash B); \Gamma \vdash E : A$ avec E un contexte de tête faible.

Remarque 8.1.1. Le lemme 8.1.3 concernant $K\{t\}$ n'est vrai que parce que le trou apparaît exactement une fois dans K. En particulier, il n'est pas vrai pour la substitution ordinaire $u[x\backslash t]$ où x peut apparaître zéro ou plus d'une fois dans N.

8.1.4. Réductions

Définition 8.1.7 (β-réduction atomique)**.** La *réduction β atomique*, notée $\rightsquigarrow_\beta$, est la relation binaire sur les termes définie par les quatre règles suivantes.

$$\mathtt{app}(\mathtt{fun}(x.t), u) \rightsquigarrow_\beta t[x\backslash u] \tag{8.1}$$
$$\mathtt{match}_{\mathtt{Nat}}(\mathtt{lit}(0), u, x.v) \rightsquigarrow_\beta N \tag{8.2}$$
$$\mathtt{match}_{\mathtt{Nat}}(\mathtt{lit}(n+1), u, x.v) \rightsquigarrow_\beta v[x\backslash \mathtt{lit}(n)] \tag{8.3}$$
$$\mathtt{fix}(t) \rightsquigarrow_\beta \mathtt{app}(t, \mathtt{fix}(t)) \tag{8.4}$$

Propriété 8.1.6. *$\rightsquigarrow_\beta$ est déterministe.*

Définition 8.1.8 (Réduction β)**.** Le terme M se réduit en N modulo β, ce qui est noté $M \to_\beta N$, s'il existe un contexte général K et deux termes M_0 et N_0 tels que $M = K[t_0]$ et $N = K[N_0]$ avec $M_0 \rightsquigarrow_\beta N_0$.

Théorème 8.1.1. *$\to_\beta$ est confluente.*

Démonstration. La méthode de Tait, Martin-Löf et Takahashi employée au chapitre précédent s'adapte sans difficulté à $\mathcal{PCF}$. □

Corollaire 8.1.1. *$\to_\beta$ vérifie la propriété de Church-Rosser, c'est-à-dire, $\to_\beta^{s*} = {\to_\beta}^*; \to_\beta^{op*}$.*

Définition 8.1.9 (Équivalence β)**.** La relation d'*équivalence β* entre termes, notée $M \equiv_\beta N$, est la clôture symétrique réflexive-transitive $\to_\beta^{s*}$ de la réduction β.

Propriété 8.1.7. *$\mathcal{PCF}[{\to_\beta}^*] \subseteq {\to_\beta}^*$.*

Propriété 8.1.8. *$\to_\beta \subseteq \mathcal{PCF}[\rightsquigarrow_\beta^=]$.*

Propriété 8.1.9. *L'équivalence β est la plus petite $\mathcal{PCF}$-congruence contenant $\rightsquigarrow_\beta$.*

Démonstration. Par le lemme 8.1.2 et le corollaire 8.1.1, il suffit de démontrer que les relations ${\to_\beta}^*;{\to_\beta}^{op*}$ et $\mathcal{PCF}[\rightsquigarrow_\beta^{s*}]$ sont identiques. On montre l'inclusion dans les deux sens.

Pour $\mathcal{PCF}[\rightsquigarrow_\beta^{s*}] \subseteq {\to_\beta}^*;{\to_\beta}^{op*}$, on remarque tout d'abord que $\rightsquigarrow_\beta \subseteq \to_\beta$ et donc $\rightsquigarrow_\beta^{s*} \subseteq \to_\beta^{s*} = {\to_\beta}^*;{\to_\beta}^{op*}$ où la dernière équation est conséquence du corollaire 8.1.1. Donc

$$\begin{aligned}
\mathcal{PCF}[\rightsquigarrow_\beta^{s*}] &\subseteq \mathcal{PCF}[{\to_\beta}^*;{\to_\beta}^{op*}] \\
&= \mathcal{PCF}[{\to_\beta}^*];\mathcal{PCF}[{\to_\beta}^{op*}] && \text{par la propriété 8.1.1} \\
&= \mathcal{PCF}[{\to_\beta}^*];\mathcal{PCF}[{\to_\beta}^*]^\circ && \text{par la propriété 8.1.4} \\
&= {\to_\beta}^*;{\to_\beta}^{*\circ} && \text{par la propriété 8.1.7.}
\end{aligned}$$

Pour ${\to_\beta}^*;{\to_\beta}^{op*} \subseteq \mathcal{PCF}[\rightsquigarrow_\beta^{s*}]$, on a

$$\begin{aligned}
{\to_\beta}^*;{\to_\beta}^{op*} &\subseteq \mathcal{PCF}[\rightsquigarrow_\beta^{=}]^*;\mathcal{PCF}[\rightsquigarrow_\beta^{=}]^{op*} && \text{par la propriété 8.1.8} \\
&\subseteq \mathcal{PCF}[\rightsquigarrow_\beta^{s*}]^*;\mathcal{PCF}[\rightsquigarrow_\beta^{s*}]^{op*} \\
&= \mathcal{PCF}[\rightsquigarrow_\beta^{s*}];\mathcal{PCF}[\rightsquigarrow_\beta^{s*}] && \text{par la propriété 8.1.5} \\
&= \mathcal{PCF}[\rightsquigarrow_\beta^{s*}] && \text{par la propriété 8.1.5.}
\end{aligned}$$

□

On a donc montré que l'équivalence β, définie comme la plus petite relation d'équivalence qui contient la réduction β, coïncide avec la plus petite $\mathcal{PCF}$-congruence qui contient la réduction β atomique. On aurait bien sûr pu intervertir théorème et définition en définissant directement $\equiv_\beta$ comme la plus petite $\mathcal{PCF}$-congruence qui contient la réduction β atomique, comme au chapitre 7.

Le fait que la relation d'équivalence engendrée par la réduction β soit une $\mathcal{PCF}$-congruence est intrinsèquement lié à la possibilité de réduire un sous-terme arbitraire. Toutefois, cette définition reflète mal les implémentations de langages fonctionnels, où l'on s'interdit de réduire le corps d'une fonction dont on ne connaît pas l'argument, et où a jamais le choix de la prochaine réduction à effectuer. Pour pallier ce défaut, on introduit la *réduction de tête faible*, plus proche d'un mécanisme d'exécution.

Définition 8.1.10 (Réduction de tête faible)**.** Le terme M se réduit en N par réduction de tête faible, ce qui est noté $M \to_{\mathsf{wh}} N$, s'il existe un contexte de tête faible E et deux termes M_0 et N_0 tels que $M = E[t_0]$ et $N = E[N_0]$ avec $M_0 \rightsquigarrow_\beta N_0$.

Remarque 8.1.2. La réduction de tête faible est une stratégie d'évaluation proche de la mécanique employée dans les langages fonctionnels non-stricts comme Haskell.

Définition 8.1.11. On appelle *radical* un terme M_0 tel qu'il existe N_0 avec $M_0 \rightsquigarrow_\beta N_0$. Une *décomposition de tête faible* d'un terme M est une paire (E, M_0) avec E un contexte de tête faible et M_0 un radical tels que $M = E\{t_0\}$.

Propriété 8.1.10. *Tout terme admet au plus une décomposition de tête faible.*

Corollaire 8.1.2. $\to_{\mathsf{wh}}$ *est déterministe.*

On termine cette sous-section avec les résultats habituels sur le typage, qui expriment que celui-ci est un invariant du calcul.

Lemme 8.1.4 (Réduction du sujet, $\rightsquigarrow_\beta$)**.** *Si* $\Gamma \vdash t : A$ *et* $M \rightsquigarrow_\beta N$ *alors* $\Gamma \vdash u : A$.

Démonstration. On raisonne par cas sur $M \rightsquigarrow_\beta N$ puis on inverse l'hypothèse de typage. Les cas des équations (8.1) et (8.3) sont réglés par le lemme 8.1.1. □

Lemme 8.1.5 (Réduction du sujet, $\to_\beta$)**.** *Si* $\Gamma \vdash t : A$ *et* $M \to_\beta N$ *alors* $\Gamma \vdash u : A$.

Démonstration. Par hypothèse, il existe K, M_0 et N_0 tels que $M = K\{t_0\}$ et $N = K\{N_0\}$. Par le lemme 8.1.3, on a Δ et B tel que $\Delta \vdash t_0 : B$ et $\diamond : (\Delta \vdash B); \Gamma \vdash K : A$. Par le lemme 8.1.4 on a $\Delta \vdash N_0 : B$, et donc $\Gamma \vdash K\{N_0\} : A$. □

Corollaire 8.1.3 (Réduction du sujet, $\to_{\mathsf{wh}}$)**.** *Si* $\Gamma \vdash t : A$ *et* $M \to_{\mathsf{wh}} N$ *alors* $\Gamma \vdash u : A$.

Définition 8.1.12. Une *valeur* V est un terme de $\mathcal{PCF}$ qui obéit à la grammaire suivante.

$$\mathsf{Val}_{\mathcal{PCF}} \ni V, W ::= \mathtt{lit}(n) \mid \mathtt{fun}(x.t)$$

Lemme 8.1.6 (Progrès)**.** *Un terme* $M : A$ *est* $\to_{\mathsf{wh}}$*-normal si et seulement c'est une valeur.*

Le théorème qui clôt cette section exprime qu'un terme clos bien typé soit diverge, soit termine sur une valeur du bon type. En termes informelle, il assure que l'exécution d'un programme, assimilée à la réduction de tête faible, ne peut pas terminer sur autre chose qu'une valeur du type attendu. Ce résultat a joué un rôle historiquement important dans l'étude des langages de programmation, mais est essentiellement une conséquence immédiate de la réduction du sujet (lemme 8.1.5).

Définition 8.1.13. Un terme M_0 *diverge*, ce qu'on note $t_0 \uparrow$, lorsqu'il existe une séquence infinie $(M_i)_{i \geq 0}$ telle que $M_i \to_{\mathsf{wh}} M_{i+1}$.

Théorème 8.1.2 (Sûreté)**.** *Si* $M : A$ *alors soit* $t \uparrow$ *soit il existe* $V : A$ *tel que* $M \to_{\mathsf{wh}}^* V$*. De plus, une telle valeur* V*, si elle existe, est unique.*

Démonstration. Puisque $\to_{\mathsf{wh}}$ est déterministe, si M diverge alors il n'a pas de forme normale pour $\to_{\mathsf{wh}}$ et donc il n'existe pas de tel V. Supposons que M ait une forme normale. Le corollaire 8.1.2 assure que celle-ci est unique. De plus, celle-ci est forcément de type A, par le lemme 8.1.5, et il s'agit forcément d'une valeur par le lemme 8.1.6. □

Corollaire 8.1.4. *Si* $M : \mathtt{Nat}$ *alors soit* M *diverge soit il existe un unique entier* n *tel que* $M \to_{\mathsf{wh}}^* \mathtt{lit}(n)$.

8.1.5. Programmer en $\mathcal{PCF}$

Il est utile d'écrire quelques programmes en $\mathcal{PCF}$ pour se forger des intuitions au sujet du langage. En plus des conventions notationnelles pour l'écriture de termes introduites au chapitre 7, on adopte les raccourcis suivants.

$$\begin{aligned}\texttt{let}(t, x.u) &\coloneqq \texttt{app}(\texttt{fun}(x.u), t)\\ \texttt{letrec}(x.(t,u)) &\coloneqq \texttt{let}(\texttt{fix}(\texttt{fun}(x.t)), x.u)\end{aligned}$$

On vérifie qu'on a bien les règles de typage dérivées qui suivent.

$$\frac{\Gamma \vdash t : A \qquad \Gamma, x : A \vdash u : B}{\Gamma \vdash \texttt{let}(t, x.u) : B} \qquad \frac{\Gamma, x : A \vdash t : A \qquad \Gamma, x : A \vdash u : B}{\Gamma \vdash \texttt{letrec}(x.(t,u)) : B}$$

Ces définitions vérifient les réductions de tête faible

$$\texttt{let}(t, x.u) \to_{\mathsf{wh}} u[x \backslash t] \qquad \texttt{letrec}(x.(t,u)) \to_{\mathsf{wh}} u[x \backslash \texttt{fix}(\texttt{fun}(x.t))]$$

et la liaison récursive valide en particulier l'équation β

$$\texttt{letrec}(x.(t,u)) \equiv_\beta u[x \backslash \texttt{letrec}(x.(t,t))].$$

On peut les utiliser pour définir simplement la fonction addition.

$$\texttt{letrec}(add.(\texttt{fun}(x.\texttt{fun}(y.\texttt{match}_{\texttt{Nat}}(x, y, z.\texttt{suc}(add\ z\ y))))), add\ \texttt{lit}(2)\ \texttt{lit}(3)))$$

Remarque 8.1.3. Du point de vue de l'utilisation, $\mathcal{PCF}$ peut être vu comme un très petit sous-ensemble du langage fonctionnel Haskell. On peut écrire des fonctions d'ordre supérieur, et ces fonctions sont partielles, au sens où elles peuvent diverger pour certaines valeurs de leur argument. De plus, comme en Haskell, les fonctions n'évaluent pas nécessairement leur argument. Ainsi, on a

$$\texttt{app}(\texttt{fun}(x.\texttt{lit}(0)), \Omega) \to_{\mathsf{wh}} \texttt{lit}(0)$$

où Ω est le terme divergent $\texttt{fix}(\texttt{fun}(x.x))$.

Haskell, comme $\mathcal{PCF}$, est parfois dit "pur", au sens où les programmes du langage n'ont pas d'effets de bord observables. Cette terminologie suppose que la partialité ne soit pas observable.

Une grande différence entre Haskell et $\mathcal{PCF}$ du point de vue de l'exécution réside dans la possibilité de partager les calculs. En $\mathcal{PCF}$, la réduction du terme

$$\texttt{fun}(x.add\ x\ x)\ (add\ \texttt{lit}(2)\ \texttt{lit}(2))$$

vers une valeur réduira deux fois le sous-terme $add\ \texttt{lit}(2)\ \texttt{lit}(2)$ vers sa valeur $\texttt{lit}(4)$. En Haskell, le calcul sera fait une seule fois via une mécanique complexe de mémorisation des résultats intermédiaires.

8.2. L'équivalence de programmes

8.2.1. L'équivalence contextuelle

On a vu au chapitre 7 une seule relation d'équivalence sur les termes, l'équivalence β. Celle-ci a du sens dès lors que la réduction β est confluente, ce qui est vrai de T comme de $\mathcal{PCF}$. Toutefois, il est assez clair en $\mathcal{PCF}$ que l'équivalence β n'est pas suffisante. Par exemple, les termes $M_1 = \texttt{fix}(\texttt{fun}(x.x))$ et $M_2 = \texttt{fix}(\texttt{fun}(x.\texttt{suc}(x)))$ divergent tous les deux, mais n'ont clairement pas de réduit commun, et donc ne peuvent pas être équivalents modulo β. Cet exemple suggère de chercher une définition plus générale de l'équivalence de termes.

Pour concevoir une notion générale d'équivalence de programmes, on peut commencer par raisonner au sujet des valeurs (au sens de la définition 8.1.12). L'équivalence de deux valeurs qui sont des abstractions est un problème essentiellement aussi compliqué que celui de deux termes arbitraires. En revanche, l'équivalence de deux valeurs qui sont des entiers littéraux est simple : deux valeurs $\texttt{lit}(n)$ et $\texttt{lit}(m)$ sont égales lorsque $n = m$. On peut partir de cette observation pour définir une forme d'équivalence pour les termes clos de type entier. Deux tels termes seront équivalents lorsqu'ils calculent tous les deux le même entier au sens de la réduction de tête faible.

Définition 8.2.1 (Équiconvergence au type $\texttt{Nat}$)**.** Deux termes M et N sont *équiconvergents au type* $\texttt{Nat}$, ce qu'on note $M \simeq_{\texttt{Nat}} N$, s'ils sont clos de type $\texttt{Nat}$ et que

$$\forall n \in \mathbb{N}, M \to^*_{\mathsf{wh}} \texttt{lit}(n) \Leftrightarrow N \to^*_{\mathsf{wh}} \texttt{lit}(n).$$

Remarquons en particulier que le théorème 8.1.2 nous assure qu'un terme clos de type entier soit diverge, soit converge vers un entier. Donc la définition ci-dessus implique que M diverge si et seulement si N diverge.

On peut maintenant utiliser la notion de contexte général pour étendre la relation $\simeq_{\texttt{Nat}}$ à des types arbitraires. L'idée est que deux termes M et N de même type sont *contextuellement équivalents* lorsqu'il n'existe aucun contexte général K tel que $K\{t\}$ et $K\{u\}$ ne s'évaluent pas vers le même entier.

Définition 8.2.2 (Équivalence contextuelle)**.** Deux termes $\Gamma \vdash t : A$ et $\Gamma \vdash u : A$ sont *contextuellement équivalents*, ce qu'on note $\Gamma \vdash t \cong u : A$, lorsque

$$\forall K \in \mathsf{Kon}_{\mathcal{PCF}}, \diamond : (\Gamma \vdash A); \cdot \vdash K : \texttt{Nat} \Rightarrow K\{t\} \simeq_{\texttt{Nat}} K\{u\}.$$

Une intuition utile est de penser aux contextes K comme à des tests qui permettent d'observer le comportement de M et N sous la forme très simple d'un entier final. En ce sens, les termes M et N sont équivalents lorsqu'ils passent tous les tests avec le même résultat ou, autrement dit, si aucun test ne permet de les distinguer. Pour cette raison, on parle aussi d'équivalence *observationnelle*.

8.2.2. Le préordre contextuel

L'équivalence contextuelle entre M et N capture la possibilité d'*échanger* M et N dans un programme sans changer le résultat final. On peut aussi s'intéresser à une notion plus

fine, où on s'autorise à remplacer M par N mais pas l'inverse. Il suffit de revisiter les définitions précédentes pour remplacer les équivalences par des implications.

Définition 8.2.3 (Raffinement au type $\mathtt{Nat}$)**.** Le terme M *raffine* le terme N *au type* $\mathtt{Nat}$, ce qu'on note $M \preceq_{\mathtt{Nat}} N$, s'ils sont clos de type $\mathtt{Nat}$ et que

$$\forall n \in \mathbb{N}, M \to_{\mathsf{wh}}^{*} \mathtt{lit}(n) \Rightarrow N \to_{\mathsf{wh}}^{*} \mathtt{lit}(n).$$

Définition 8.2.4 (Préordre contextuel)**.** Un terme $\Gamma \vdash t : A$ *approxime contextuellement* un terme $\Gamma \vdash u : A$, ce qu'on note $\Gamma \vdash t \lesssim u : A$, lorsque

$$\forall K \in \mathsf{Kon}_{\mathcal{PCF}}, \diamond : (\Gamma \vdash A); \cdot \vdash K : \mathtt{Nat} \Rightarrow K\{t\} \preceq_{\mathtt{Nat}} K\{u\}.$$

L'approximation contextuelle capture bien l'idée que remplacer M par N dans un programme peut simplement conduire celui-ci à passer de la divergence à la convergence. Si l'on considère que seule la convergence peut être vraiment observée, c'est une notion raisonnable. En particulier, une compilateur pourrait raisonnablement s'autoriser à transformer M en N, mais pas l'inverse.

On peut considérer que l'approximation contextuelle est une notion plus fondamentale que l'équivalence contextuelle, puisqu'on peut définir facilement la seconde à partir de la première mais pas l'inverse.

Propriété 8.2.1. $\Gamma \vdash t \cong u : A$ *si et seulement si* $\Gamma \vdash t \lesssim u : A$ *et* $\Gamma \vdash u \lesssim t : A$.

En des termes plus abstraits, l'équivalence contextuelle est la relation d'équivalence induite par le préordre que constitue la relation d'approximation contextuelle.

La relation d'approximation contextuelle $\Gamma \vdash t \lesssim u : A$ est une notion complexe à appréhender, en particulier lorsque A est un type d'ordre élevé. Au type $\mathtt{Nat}$, elle est assez simple, puisqu'on peut montrer qu'elle coïncide avec la relation $\preceq_{\mathtt{Nat}}$. Au type $\mathtt{Nat} \to \mathtt{Nat}$, la situation est déjà plus subtile puisqu'un terme de ce type peut soit diverger immédiatement, soit converger vers une abstraction qui elle-même diverge lorsqu'appliquée à n'importe quel argument, soit diverge pour certaines entrées et converge pour d'autres, soit toujours converger. Le type $(\mathtt{Nat} \to \mathtt{Nat}) \to \mathtt{Nat}$ est encore plus compliqué, ce qui témoigne de la combinatoire explosive de la relation d'approximation.

On peut cependant observer ou deviner quelques faits vrais de chacune des relations $\Gamma \vdash - \lesssim = : A$, et ce même si à ce stade nous ne disposons pas les outils mathématiques qui permettent de les démontrer. Tout d'abord, il s'agit clairement d'une relation réflexive et transitive. Ensuite, le terme $\Omega \coloneqq \mathtt{fix}(\mathtt{fun}(x.x))$ est minimal, tout comme $\mathtt{fix}(\mathtt{fun}(x.\mathtt{suc}(x)))$, et la relation n'est donc pas antisymétrique. De plus, les deux termes

$$M_0 \coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, \Omega, y.\mathtt{lit}(3))) \qquad M_1 \coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, \mathtt{lit}(2), y.\Omega))$$

sont incomparables, au sens où l'on a ni $M_0 \lesssim M_1 : \mathtt{Nat} \to \mathtt{Nat}$ ni $M_1 \lesssim M_0 : \mathtt{Nat} \to \mathtt{Nat}$. Enfin, si $M : A \to B$ et que $N_0 \lesssim N_1 : A$, on devrait avoir $M\,N_0 \lesssim M\,N_1$, ce qui reflète que M est incapable de tester si son argument diverge.

8.2.3. L'attrait des modèles

La définition de l'équivalence et de l'approximation contextuelles, en plus d'être assez intuitive, ont l'avantage d'être faciles à adapter à d'autres langages de programmation. En revanche, elles sont difficiles à utiliser directement. S'il est facile de démontrer que deux termes M et N ne sont *pas* équivalents, puisqu'il suffit d'exhiber un contexte K qui les distingue, démontrer exige de montrer qu'il n'existe aucun K de la sorte.

Une approche naïve pour montrer $\Gamma \vdash t \cong u : A$ consisterait à procéder par induction sur les contextes. C'est malheureusement une approche trop frustre et très éloignée de la façon qu'on a de raisonner sur les programmes, même informellement.

À la place, on va réutiliser une approche vue pour T et bâtir un modèle de $\mathcal{PCF}$, c'est-à-dire une fonction d'interprétation $[\![-]\!]$ des types et des termes de $\mathcal{PCF}$ dans une famille d'objets mathématiques. Cette interprétation aura la propriété habituelle d'être un invariant du calcul, au sens où

$$M \to_{\mathsf{wh}} N \quad \Rightarrow \quad [\![t]\!] = [\![u]\!]. \tag{8.5}$$

On va de plus montrer qu'elle aura la propriété suivante, beaucoup plus délicate.

$$[\![t]\!] = [\![u]\!] \quad \Rightarrow \quad \Gamma \vdash t \cong u : A. \tag{8.6}$$

Le modèle est donc un invariant du calcul, mais aussi un outil pour raisonner sur l'équivalence. Pour montrer l'équivalence de deux termes, il suffit donc en théorie de calculer leurs interprétations et de constater que celles-ci sont les mêmes.

Remarque 8.2.1. Le terme "calculer" ci-dessus est à prendre au sens mathématique plutôt qu'informatique, le calcul en question étant un raisonnement équationnel arbitrairement complexe. L'équivalence de termes $\mathcal{PCF}$ n'est pas décidable et ne se prête donc pas à un calcul algorithmique exact.

8.3. Un modèle dans les ensembles ordonnés inductifs

8.3.1. Intuitions

Un modèle de $\mathcal{PCF}$ doit autoriser certains principes de raisonnement s'il doit respecter la propriété équation (8.6). En particulier, on voudrait qu'il nous autorise à penser aux termes de types fonctionnels comme à des fonctions mathématiques, qui ont pour caractéristique d'être *extensionnelles*, c'est-à-dire carctérisées uniquement par leur comportement entrée/sortie. Deux fonctions sont égales lorsqu'elles envoient des arguments égaux sur des résultats égaux. Cela suggère de partir du modèle ensembliste de T vu en section 7.4.3.

Essayons d'adopter naïvement la sémantique ensembliste. On interprète comme précédemment le type des entiers par l'ensemble des entiers et le type des fonctions par l'ensemble des fonctions de l'interprétation du domaine dans l'interprétation du codomaine. Interpréter la plupart des constructions ne pose pas de difficulté, étant soit déjà présente en T, soit une simplification d'une construction existante (la conditionnelle,

simplification de la récursion primitive). C'est bien entendu la récursion générale qui va poser problème.

Considérons un terme clos $M : \mathtt{Nat} \to \mathtt{Nat}$, et posons $M_0 = \mathtt{fix}(t)$. Si l'on adopte exactement les même définitions que dans le modèle ensembliste de T, l'interprétation de M doit être une fonction de l'ensemble à un élément dans l'ensemble des fonctions entre entiers, c'est-à-dire essentiellement une fonction $f : \mathbb{N} \to \mathbb{N}$, et l'interprétation de M_0 un entier $x_0 \in \mathbb{N}$. Mais on doit avoir $M_0 \to_{\mathsf{wh}} \mathtt{app}(t, t_0)$ et donc, puisque l'interprétation de M_0 est invariante par réduction, $x_0 = f(x_0)$. Autrement dit, il faut que toute fonction des entiers dans les entiers définissable par un terme de $\mathcal{PCF}$ a un point fixe. Or, tel quel, c'est absurde puisque par exemple le terme

$$\mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, \mathtt{lit}(1), y.\mathtt{lit}(0)))$$

correspond à la fonction qui envoie 0 dans 1 et $n+1$ dans 0. Encore plus simple, le terme $\mathtt{fun}(x.\mathtt{suc}(x))$ serait interprété par la fonction successeur dont un point fixe serait un entier n tel que $n = n + 1$.

Pour échapper à cette difficulté, on peut commencer par remarquer que notre notion de modèle traite de l'équivalence contextuelle mais pas de l'approximation. La relation $\Gamma \vdash - \lesssim = : A$ est clairement réflexive et transitive, ce qui suggère que l'objet mathématique $[\![A]\!]$ devrait être un ensemble muni d'une relation d'ordre, et d'exiger l'analogue de la propriété 8.6 pour cette relation :

$$[\![t]\!] \leq [\![u]\!] \quad \Rightarrow \quad \Gamma \vdash t \lesssim u : A. \tag{8.7}$$

Soit X l'ensemble ordonné interprétant un type A quelconque. Peut-on deviner les propriétés de X qui sont indépendantes de A ? Tout d'abord, puisque le terme Ω est minimal pour l'approximation contextuelle à tous les types, X doit contenir un élément minimal $\bot$ qui lui servira d'inteprétation. Ensuite, toute fonction $f : X \to X$ doit avoir un point fixe à partir du moment où elle est définissable en $\mathcal{PCF}$. Il est clair qu'une telle fonction doit au minimum être croissante, puisqu'il est impossible de tester la divergence d'un argument. De plus, si $f = [\![t]\!]$ a plusieurs point fixe, on doit disposer d'un choisir un bien précis pour interpréter $\mathtt{fix}(t)$. Appelons y_0 ce choix. On doit avoir

$$\begin{aligned}
\bot &\leq y_0 && \text{car } \bot \text{ est minimal} \\
f(\bot) &\leq f(y_0) = y_0 && \text{car } f \text{ est croissante et } y_0 \text{ un point fixe de } f \\
f(f(\bot)) &\leq f(y_0) = y_0 \\
f^3(\bot) &\leq f(y_0) = y_0 \\
&\dots
\end{aligned}$$

et donc $f^n(\bot) \leq y_0$ pour tout $n \in \mathbb{N}$. Cette observation suggère qu'un choix canonique pour y_0 pourrait être la *limite* de la suite $(f^n(\bot))_{n\in\mathbb{N}}$, à supposer que celle-ci soit bien un point fixe de f. Ici, une limite est à prendre au sens de la théorie de l'ordre, c'est-à-dire un supremum.

8.3.2. Les ensembles ordonnés inductifs

8.3.2.1. Définitions et intuitions

Lorsque les ensembles ordonnés sont les interprétations des types d'un langage de programmation permettant la récursion générale, la relation d'ordre correspond au fait d'être mieux défini. Par exemple, dans le cas simple de l'ordre qui interprète le type des entiers de $\mathcal{PCF}$, le terme divergent Ω devrait être minimalement défini, tandis qu'un entier litéral devrait être interprété par un élément maximalement défini puisqu'il n'est pas possible de "converger plus". Il est clair que la relation d'ordre aux types fonctionnels sera nettement plus complexe a décrire.

Nous allons exiger la présence de certains suprema pour être capable de rendre compte de certaines opérations du langage. Par exemple, puisque Ω est de type A, nous allons exiger la présence d'un minimum pour tout ordre qui va interpréter un type A. Or, le minimum d'un ordre est le supremum vide, qui doit donc exister. Mais quels autres suprema demander ?

Une solution serait de ne pas se poser la question et d'adopter les treillis complets comme modèles des langages de programmation. C'est une des solutions historiques. Toutefois, demander l'existence de tous les suprema nous éloigne a priori de la programmation. Reprenons l'exemple du type `Nat`, et supposons qu'il soit interprété dans un certain treillis complet $[\![\texttt{Nat}]\!]$. On peut supposer que les entiers littéraux `lit`(1) et `lit`(2) sont interprétés par les entiers 1 et 2. Le supremum $1 \vee 2$ existe puisque $[\![\texttt{Nat}]\!]$ est un treillis complet et a fortiori un treillis. Mais il est loin d'être clair que ce supremum soit l'interprétation d'un terme clos de type `Nat`, et on souhaite autant que possible éviter d'avoir des éléments qui ne sont pas les interprétations de programmes.

On voudrait se donner, en plus du supremum vide, uniquement les supremum des parties qui tendent à l'infini vers un résultat. Mais comment préciser cette notion de convergence ? Une idée intuitive pour ce faire serait de considérer que de telles parties sont les *ω-chaînes* de X, c'est-à-dire les suites $(x_i)_{i\in\mathbb{N}}$ d'éléments de X avec $x_i \leq x_{i+1}$ pour tout i. C'est la bonne intuition, et on va légèrement généraliser cette définition au delà des ordres totaux grâce à la notion de partie *filtrante*.

Définition 8.3.1 (Ensemble ordonné filtrant, inductif)**.** Un ensemble ordonné est *filtrant* si chacune de ses parties finies a un majorant. On dit qu'une partie P de X est *filtrante dans* X lorsque P, munie de la relation d'ordre de X, est un ensemble ordonné filtrant. Un ensemble ordonné est dit *filtrant-complet* s'il dispose de tous les suprema filtrants. Il est dit *inductif* s'il est filtrant-complet et dispose de surcroît d'un plus petit élément.

Les propriétés suivantes ne sont pas difficiles mais peuvent permettre de mieux comprendre la notion de partie filtrante.

Propriété 8.3.1. *Une partie P de X est filtrante si et seulement si elle n'est pas vide et que pour tout x_1, x_2 dans P, il existe x dans P tel que $x_1 \leq x$ et $x_2 \leq x$.*

Propriété 8.3.2. *Une partie finie P de X est filtrante si et seulement si elle a un maximum.*

Propriété 8.3.3. *Toute ω-chaîne de X est une partie filtrante de X.*

On peut voir une partie filtrante P de X comme un ensemble de points qui converge progressivement vers un point idéal $x \in X$ situé à l'infini, sa limite au sens de la théorie des ordres, c'est-à-dire son supremum. Si P est fini alors la limite est atteinte au sens où x appartient à P, en vertu de la propriété 8.3.2. Si P est infini, ce n'est pas le cas, et le fait que X est inductif indique précisément que x existe dans X.

Remarque 8.3.1. En anglais, les ensemble *filtrants* sont dits *directed* (dirigés), et les ensembles ordonnés inductifs sont parfois appelés *directed-complete partial orders* (DCPO). On rencontre aussi le terme "ordre partiel complet", remarquablement peu informatif. La terminologie "ensemble ordonné inductif" est due à Gordon Plotkin [8] et Paul Taylor.

Les programmes vont être interprétés comme des fonctions entre ensembles ordonnés inductifs qui commutent aux suprema filtrants. Soit P une partie de X et $f : X \to Y$ une fonction. On écrit $f(P)$ pour l'image directe de P par f, c'est-à-dire l'ensemble $\{f(x) \in \mathcal{Y} \mid x \in P\}$.

Propriété 8.3.4. *Si P est une partie filtrante de X et que $f : X \to Y$ est croissante, alors $f(P)$ est une partie filtrante de Y.*

Définition 8.3.2 (Fonction finitaire)**.** Soient X et Y des ensembles ordonnés filtrant-complets. Une fonction croissante $f : X \to Y$ est *finitaire* si elle commute aux suprema filtrants, autrement dit, si

$$f\left(\bigvee_{x \in P} x\right) = \bigvee_{x \in P} f(x) \tag{8.8}$$

pour toute partie filtrante P de X.

Notons que le supremum dans le membre droit de l'équation (8.8) existe en vertu de la propriété 8.3.4, sans quoi la définition n'aurait pas de sens.

Une fonction finitaire est parfois appelée *continue au sens de Scott* ou simplement *Scott-continue*. Il s'agit d'une propriété très importante qu'on peut voir comme une version abstraite de la notion de fonction calculable. Dans ce contexte, il faut penser au supremum x d'une partie filtrante P comme à un objet idéal infini et aux éléments de P comme à des objets finis. Si f est finitaire alors $f(x)$ est entièrement fixé par les résultats de f sur les éléments de P. Autrement dit, le comportement de f sur les objets infinitaires est entièrement dicté par son comportement sur les objets finitaires. L'infini est donc présent par commodité mais ne joue pas de vrai rôle dans les calculs.

Remarque 8.3.2. Il est possible de rendre cette intuition plus exacte en définissant une notion précise d'élément finitaire de X, et en exigeant que tout élément x de X soit le supremum des éléments finitaires qui lui sont inférieurs. On aboutit ainsi aux notions de *treillis algébrique* ou de *domaine de Scott*, qui dépassent le cadre de ces notes.

Propriété 8.3.5. *Soit X un ensemble ordonné filtrant-complet. La fonction identité $id : X \to X$ est finitaire. La composition $gf : X \to Z$ de $f : X \to Y$ et $g : Y \to Z$ finitaires est finitaire.*

Enfin, les fonctions finitaires ont une caractéristique essentielle lorsqu'il s'agit de modéliser un langage de programmation avec la récursion générale. Supposons que X soit un ensemble ordonné inductif dont nous notons $\bot$ le plus petit élément. Si f est une fonction croissante, alors la suite $\bot \leq f(\bot) \leq f(f(\bot)) \leq \dots$ est une ω-chaîne. Le supremum de cette suite est toujours le plus petit point fixe de f.

Théorème 8.3.1 (Kleene)**.** *Soit X un ensemble ordonné inductif. Si $f : X \to X$ est finitaire, alors f a un plus petit point fixe $\mathsf{fix}(f)$ défini par*

$$\mathsf{fix}(f) = \bigvee_{n \geq 0} f^n(\bot).$$

Démonstration. On commence par montrer que $\mathsf{fix}(f)$ est bien un point fixe de f. On a

$$f(\mathsf{fix}(f)) = f(\bigvee_{n\geq 0} f^n(\bot)) = \bigvee_{n\geq 0} f^{n+1}(\bot) = \bigvee_{n\geq 1} f^n(\bot) = \bigvee_{n\geq 0} f^n(\bot) = \mathsf{fix}(f).$$

Supposons maintenant que x soit un point fixe quelconque de f. Puisque $\bot$ est minimal dans X, on a $\bot \leq x$ et donc $f^n(\bot) \leq f^n(x) = x$ pour tout $n \geq 0$. Donc $\mathsf{fix}(f) = \bigvee_{n\geq 0} f^n(\bot) \leq x$ par la propriété universelle du supremum. □

8.3.2.2. Constructions

Dans ce qui suit X, Y, Z désignent des ensembles ordonnés inductifs.

Ordre discret, soulèvement, produits cartésiens. La plupart des constructions de la section 3.2 préservent l'inductivité. De plus, les fonctions croissantes liées à ces constructions sont finitaires. Ainsi :

- l'ensemble ordonné discret $\mathsf{Disc}\, S$ sur un ensemble S est filtrant-complet, et la fonction croissante $f^{\#} : \mathsf{Disc}\, S \to X$ associée à $f : S \to X$ est finitaire,
- le soulèvement $\uparrow X$ d'un ensemble ordonné X filtrant-complet est toujours inductif, la fonction $\langle - \rangle : X \to \uparrow X$ est finitaire, la fonction $\langle f \rangle : \uparrow X \to \uparrow Y$ associée à une fonction $f : X \to Y$ est finitaire,
- le produit cartésien $X \times Y$ de deux ensembles filtrant-complets (resp. inductifs) X et Y est filtrant-complet (resp. inductif), les projections π_i sont finitaires et le pairage de deux fonctions finitaires est finitaire.
- l'ordre $\mathbf{1}$ a un seul élément $()$ est inductif, et la fonction $!_X : X \to \mathbf{1}$ qui envoie tout $x \in X$ sur $()$ est finitaire.

Le lemme suivant concernant les produits cartésiens est essentiel.

Lemme 8.3.1. *Soient X_1, X_2, Y des ensembles ordonnés filtrant-complets. Une fonction croissante $f : X_1 \times X_2 \to Y$ est finitaire si et seulement si elle est séparément finitaire, au sens où pour tout x_1 fixé la fonction $f(x_1, -) : X_2 \to Y$ est finitaire, et pour tout x_2 fixé la fonction $f(-, x_2) : X_1 \to Y$ est finitaire.*

Démonstration. La direction seulement si est évidente. Pour la direction si, supposons f séparément finitaire. Soit P une partie filtrante de $X_1 \times X_2$. On pose $P_i \coloneqq \pi_i(P)$ pour $i \in \{1,2\}$. Ces parties sont filtrantes et on a $\bigvee P = (\bigvee P_1, \bigvee P_2)$.

On doit montrer $f(\bigvee P) = \bigvee_{x\in P} f(x)$. Puisque f est croissante, on a $\bigvee_{x\in P} f(x) \leq f(\bigvee P)$, et il suffit donc de montrer $f(\bigvee P) \leq \bigvee_{x\in P} f(x)$ pour établir l'égalité. Par propriété universelle du supremum, il suffit de montrer qu'on a $f(\bigvee P) \leq x'$ pour tout x' tel que $\bigvee_{x\in P} f(x) \leq x'$.

Soit x' tel que $f(x) \leq x'$ pour tout $x \in P$. Soient $x_1 \in P_1$ et $x_2 \in P_2$. Par définition, il existe $x_2' \in P_2$ et $x_1' \in P_1$ tels que $(x_1, x_2') \in P$ et $(x_1', x_2) \in P$. Comme P est filtrant, il doit exister $(x_1'', x_2'') \in P$ tel que $(x_1, x_2'), (x_1', x_2) \leq (x_1'', x_2'')$, et donc $f(x_1, x_2) \leq f(x_1'', x_2'') \leq x'$. Puisque f est finitaire en son premier argument, on a $f(\bigvee X_1, x_2) \leq x'$. Puisque f est finitaire en son deuxième argument, on a $f(\bigvee X_1, \bigvee X_2) \leq x'$. On conclut $f(\bigvee X) = f(\bigvee X_1, \bigvee X_2) \leq x'$. □

Ordre des fonctions finitaires. Un cas plus délicat est celui de l'ensemble ordonné $[X, Y]$ des fonctions croissantes de X dans Y, avec X et Y inductifs. Pour que l'évaluation soit finitaire, on est amené à se restreindre au sous-ensemble ordonné des fonctions finitaires de X dans Y, qu'on note $[X, Y]_{\mathsf{fin}}$.

Propriété 8.3.6. $[X, Y]_{\mathsf{fin}}$ *est un ensemble ordonné inductif.*

Démonstration. Soit F une partie filtrante de $[X, Y]_{\mathsf{fin}}$. Soit x un élément de X. Puisque l'ordre sur les fonctions est point à point, la partie $F_x \coloneqq \{f(x) \mid f \in F\}$ de Y est filtrante, et son unique supremum $\bigvee_{f\in F} f(x)$ existe donc dans Y. On a ainsi défini une fonction g de X dans Y par $g(x) = \bigvee_{f\in F} f(x)$. Cette fonction est croissante puisque si $x \leq x'$ alors tout élément de F_x est inférieur à un élément de $F_{x'}$, et donc $g(x) = \bigvee F_x \leq \bigvee F_{x'} = g(x')$. Cette fonction est finitaire puisque si P est une partie filtrante de X, on a

$$g\left(\bigvee_{x\in P} x\right) = \bigvee_{f\in F} f\left(\bigvee_{x\in P} x\right) = \bigvee_{f\in F}\bigvee_{x\in P} f(x) = \bigvee_{x\in P}\bigvee_{f\in F} f(x) = \bigvee_{x\in P} g(x).$$

Supposons h tel que h majore F. Alors pour tout $x \in X$ et $f \in F$, on a $f(x) \leq h(x)$. Donc $g(x) = \bigvee_{f\in F} f(x) \leq h(x)$ et g est bien le supremum de F. □

Le morphisme d'évaluation et la curryfication d'un morphisme finitaire sont définis comme pour les ensembles ordonnés. Il faut simplement vérifier qu'ils sont finitaires.

Propriété 8.3.7. *Le morphisme d'évaluation* $ev_{X,Y} : [X, Y]_{\mathsf{fin}} \times X \to Y$ *est finitaire.*

Démonstration. Soit F une partie de $[X, Y]_{\mathsf{fin}}$ et P une partie de X. On a

$$\begin{aligned} ev_{X,Y}\left(\bigvee F, \bigvee X\right) &= \bigvee_{f\in F} f\Big(\bigvee_{x\in X} x\Big) \\ &= \bigvee_{f\in F}\bigvee_{x\in X} f(x) && (f \text{ est finitaire}) \\ &= \bigvee_{f\in F}\bigvee_{x\in X} ev_{X,Y}(f, x). \end{aligned}$$

□

Propriété 8.3.8. *Si $f : Z \times X \to Y$ est une fonction finitaire, alors :*

- *pour tout $z \in Z$ la fonction croissante $f_z : X \to Y$ est finitaire,*
- *$curry_{X,Y,Z}(f) : Z \to [X, Y]_{\mathsf{fin}}$ est finitaire.*

Démonstration. Conséquence immédiate du lemme 8.3.1. □

8.3.2.3. Opérateur de point fixe

Le théorème 8.3.1 exprime l'existence d'un plus petit point fixe pour toute fonction f de X dans X dès lors que X est inductif et f finitaire. Pour interpréter la récursion générale, il faut de plus que le calcul de ce plus petit point fixe soit lui-même finitaire.

Propriété 8.3.9. *La fonction $\mathsf{fix}_X : [X, X]_{\mathsf{fin}} \to X$ qui envoie f dans $\mathsf{fix}(f)$ est finitaire.*

Les entiers naturels. Soient X et Y deux ensembles ordonnés et y un élément de Y On écrit $const(y) : X \to Y$ pour la fonction qui envoie tout x sur y. Cette fonction est finitaire.

Propriété 8.3.10. *Soit X filtrant-complet. Il existe une fonction finitaire*

$$match_X : \uparrow \mathsf{Disc}\,\mathbb{N} \times (X \times [\mathsf{Disc}\,\mathbb{N}, X]_{\mathsf{fin}}) \to X$$

qui satisfait les équations suivantes.

$$match_X(\langle\,\rangle, f_0, f_s) = \bot_X \tag{8.9}$$
$$match_X(\langle 0\rangle, x_0, f_s) = x_0 \tag{8.10}$$
$$match(\langle n+1\rangle, x_0, f_s) = f_s(n) \tag{8.11}$$

8.3.3. Le modèle inductif

On peut maintenant utiliser toute la structure étudiée en section 8.3.2 pour interpréter $\mathcal{PCF}$ dans les ensembles ordonnés inductifs. Cette interprétation va beaucoup ressembler à celle de T dans les ensembles définie en section 7.4.3.

Chaque type A de $\mathcal{PCF}$ est interprété par un ensemble ordonné inductif $[\![A]\!]_{\mathsf{Ind}}$ défini par récurrence sur la structure de A.

Définition 8.3.3 (Modèle inductif, interprétation des types et contextes)**.** On interprète chaque type A (resp. contexte Γ) par un ensemble $[\![A]\!]_{\mathsf{Ind}}$ (resp. $[\![\Gamma]\!]_{\mathsf{Ind}}$) défini par récurrence sur sa structure comme suit.

$$[\![-]\!]_{\mathsf{Ind}} : \mathsf{Type}_{\mathrm{T}} \to \mathbf{Ind} \qquad [\![-]\!]_{\mathsf{Ind}} : \mathsf{Con}_{\mathrm{T}} \to \mathbf{Ind}$$
$$[\![\texttt{Nat}]\!]_{\mathsf{Ind}} = \uparrow \mathsf{Disc}\,\mathbb{N} \qquad [\![\cdot]\!]_{\mathsf{Ind}} = \mathbf{1}$$
$$[\![A \to B]\!]_{\mathsf{Ind}} = [[\![A]\!]_{\mathsf{Ind}}, [\![B]\!]_{\mathsf{Ind}}]_{\mathsf{fin}} \qquad [\![\Gamma, x : A]\!]_{\mathsf{Ind}} = [\![\Gamma]\!]_{\mathsf{Ind}} \times [\![A]\!]_{\mathsf{Ind}}$$

$$[\![-]\!]_{\mathsf{Ind}} : \mathcal{PCF}(\Gamma; A) \to [\![\Gamma]\!]_{\mathsf{Ind}} \to [\![A]\!]_{\mathsf{Ind}}$$

$$\left[\!\!\left[\frac{\Gamma \supseteq x : A}{\Gamma \vdash x : A} \right]\!\!\right]_{\mathsf{Ind}} = \pi_2 \circ [\![\Gamma \supseteq x : A]\!]_{\mathsf{Ind}}$$

$$\left[\!\!\left[\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathtt{app}(t,u) : B} \right]\!\!\right]_{\mathsf{Ind}} = ev_{[\![A]\!]_{\mathsf{Ind}},[\![B]\!]_{\mathsf{Ind}}} \circ \langle f, g \rangle \quad \text{où } f = [\![\Gamma \vdash t : A \to B]\!]_{\mathsf{Ind}},\ g = [\![\Gamma \vdash u : A]\!]_{\mathsf{Ind}}$$

$$\left[\!\!\left[\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \mathtt{fun}(x.t) : A \to B} \right]\!\!\right]_{\mathsf{Ind}} = curry([\![\Gamma, x : A \vdash t : B]\!]_{\mathsf{Ind}})$$

$$\left[\!\!\left[\frac{}{\Gamma \vdash \mathtt{zero} : \mathtt{Nat}} \right]\!\!\right]_{\mathsf{Ind}} = const(\langle 0 \rangle)$$

$$\left[\!\!\left[\frac{\Gamma \vdash t : \mathtt{Nat}}{\Gamma \vdash \mathtt{suc}(t) : \mathtt{Nat}} \right]\!\!\right]_{\mathsf{Ind}} = \langle suc \rangle \circ [\![\Gamma \vdash t : \mathtt{Nat}]\!]_{\mathsf{Ind}}$$

$$\left[\!\!\left[\frac{\Gamma \vdash t : \mathtt{Nat} \qquad \Gamma \vdash u : A \qquad \Gamma, x : \mathtt{Nat} \vdash v : A}{\Gamma \vdash \mathtt{match}_{\mathtt{Nat}}(t,u,x.v) : A} \right]\!\!\right]_{\mathsf{Ind}} = match_{[\![A]\!]_{\mathsf{Ind}}}(f,g,h) \quad \text{où } f = [\![\Gamma \vdash t : \mathtt{Nat}]\!]_{\mathsf{Ind}},\ g = [\![\Gamma \vdash u : A]\!]_{\mathsf{Ind}},\ h = [\![\Gamma, x : \mathtt{Nat} \vdash v : A]\!]_{\mathsf{Ind}}$$

$$\left[\!\!\left[\frac{\Gamma \vdash t : A \to A}{\Gamma \vdash \mathtt{fix}(t) : A} \right]\!\!\right]_{\mathsf{Ind}} = \mathsf{fix}_{[\![A]\!]_{\mathsf{Ind}}} \circ [\![t]\!]_{\mathsf{Ind}}$$

FIG. 8.5. : Modèle inductif de $\mathcal{PCF}$: termes

Comme pour le modèle ensembliste de T, les substitutions sont interprétées par des fonctions à valeur dans le produit cartésien, et le jugement d'affaiblissement par une projection. On ne répète pas ces définitions, identiques à ceci près qu'elles définissent des fonctions finitaires.

Définition 8.3.4 (Modèle inductif, interprétation des termes)**.** On interprète une dérivation de typage $\Gamma \vdash t : A$ par une fonction finitaire de $[\![\Gamma]\!]_{\mathsf{Ind}}$ dans $[\![A]\!]_{\mathsf{Ind}}$. La figure 8.5 présente les clauses de l'interprétation.

Lemme 8.3.2 (Fonctorialité)**.** $[\![\Delta \vdash t[\sigma] : A]\!]_{\mathsf{Ind}} = [\![\Gamma \vdash t : A]\!]_{\mathsf{Ind}} \circ [\![\Delta \vdash \sigma : \Gamma]\!]_{\mathsf{Ind}}$.

Théorème 8.3.2 (Invariance)**.** *Si* $\Gamma \vdash t : A$ *et* $M \to_\beta M'$ *alors*

$$[\![\Gamma \vdash t : A]\!]_{\mathsf{Ind}} = [\![\Gamma \vdash t' : A]\!]_{\mathsf{Ind}}.$$

Démonstration. C'est la conséquence directe du lemme 8.3.2 et de toutes les équations établies à la section 8.3.2. □

Pour établir le lemme qui suit, il faut de nouveau définir une relation logique entre les termes de T et les éléments du modèle.

Lemme 8.3.3 (Adéquation du modèle inductif)**.** *Soit* $\cdot \vdash t : \mathtt{Nat}$*.*

- *Si* $[\![\cdot \vdash t : \mathtt{Nat}]\!]_{\mathsf{Ind}}() = \langle\,\rangle$ *alors* $M \uparrow$*.*
- *Si* $[\![\cdot \vdash t : \mathtt{Nat}]\!]_{\mathsf{Ind}}() = \langle n \rangle$ *alors* $M \to^*_{\mathsf{wh}} \mathtt{lit}(n)$*.*

Corollaire 8.3.1. *Soient* $\Gamma \vdash t : A$ *et* $\Gamma \vdash u : A$*. Si* $[\![\Gamma \vdash t : A]\!]_{\mathsf{Ind}} \leq [\![\Gamma \vdash u : A]\!]_{\mathsf{Ind}}$ *alors* $\Gamma \vdash t \lesssim u : A$*.*

Exercices

* **Ex. 41** — Expliciter la définition de $K\{M\}$.

* **Ex. 42** — Démontrer le lemme 8.1.3. En déduire une définition inductive du jugement $\diamond : (\Delta \vdash B); \Gamma \vdash K : A$.

* **Ex. 43** — Démontrer la propriété 8.1.10.

* **Ex. 44** — Démontrer le lemme 8.1.6.

* **Ex. 45** — Démontrer que la relation $\Gamma \vdash - \lesssim = : A$ équipe l'ensemble $\mathcal{PCF}(\Gamma; A)$ d'une structure de préordre.

* **Ex. 46** — Démontrer la propriété 3.2.4.

** **Ex. 47** — Démontrer la propriété 3.2.18.

* **Ex. 48** — Définir une construction qui ajoute un maximum à un (pré)ordre X uniquement en utilisant les constructions de la section 3.1.

* **Ex. 49** — Démontrer la propriété 8.3.2.

* **Ex. 50** — Soient X_1 et X_2 des ensembles ordonnés inductifs. Soit P une partie filtrante de $X_1 \times X_2$. Soit P_i la partie de X_i définie comme $\pi_i(P)$ pour $i\{1, 2\}$. Démontrer que P_1 et P_2 sont filtrantes, puis que $\bigvee P = (\bigvee P_1, \bigvee P_2)$.

* Exercice 51 Propriétés de la clôture contextuelle

Démontrer les propriétés 8.1.1 à 8.1.5.

*** Exercice 52 Évaluation à grands pas

On définit le jugement inductif $M \Downarrow V$ par les règles suivantes.

$$\boxed{M \Downarrow V} \qquad \frac{}{\texttt{fun}(x.M) \Downarrow \texttt{fun}(x.M)} \qquad \frac{}{\texttt{lit}(n) \Downarrow \texttt{lit}(n)}$$

$$\frac{M \Downarrow \texttt{fun}(x.P) \qquad P[x\backslash N] \Downarrow V}{\texttt{app}(M, N) \Downarrow V} \qquad \frac{M \Downarrow \texttt{lit}(n)}{\texttt{suc}(M) \Downarrow \texttt{lit}(n+1)} \qquad \frac{M \Downarrow \texttt{lit}(0) \qquad N \Downarrow V}{\texttt{match}_{\texttt{Nat}}(M, N, x.P) \Downarrow V}$$

$$\frac{M \Downarrow \texttt{lit}(n+1) \qquad P[x\backslash \texttt{lit}(n)] \Downarrow V}{\texttt{match}_{\texttt{Nat}}(M, N, x.P) \Downarrow V} \qquad \frac{\texttt{app}(M, \texttt{fix}(M)) \Downarrow V}{\texttt{fix}(M) \Downarrow V}$$

Les premières questions ci-dessous utilisent les termes définis à la section 8.1.5.

1. Construire une dérivation de $add\ \texttt{lit}(1)\ \texttt{lit}(2) \Downarrow \texttt{lit}(3)$.
2. Pourquoi ne peut-il pas y avoir de valeur V telle que $\Omega \Downarrow V$, informellement ? Même question pour le terme $\texttt{app}(\texttt{lit}(0), \texttt{lit}(0))$.

3. Démontrer que la relation $\Downarrow : \mathsf{Term}_{\mathcal{PCF}} \nrightarrow \mathsf{Val}_{\mathcal{PCF}}$ est déterministe.
4. Démontrer que si $M \Downarrow V$ alors $M \to_{\mathsf{wh}}^* V$.
5. Démontrer que si $M \to_{\mathsf{wh}}^* V$ alors $M \Downarrow V$.

* Exercice 53 Équivalences et inéquivalences de $\mathcal{PCF}$

Considérons les termes M_i : $\mathtt{Nat} \to \mathtt{Nat}$ ci-dessous. On rappelle que Ω désigne le terme $\mathtt{fix}(\mathtt{fun}(x.x))$.

$$\begin{aligned}
M_1 &\coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, \Omega, y.1))\\
M_2 &\coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, 42, y.\Omega))\\
M_3 &\coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, 42, y.1))\\
M_4 &\coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, 42, y.2))\\
M_5 &\coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, 42, y.\mathtt{match}_{\mathtt{Nat}}(y, 12, z.\Omega)))\\
M_6 &\coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, 42, y.\mathtt{match}_{\mathtt{Nat}}(y, \Omega, z.13)))\\
M_7 &\coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, 42, y.\mathtt{match}_{\mathtt{Nat}}(y, 12, z.13)))\\
M_8 &\coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, 42, y.\mathtt{match}_{\mathtt{Nat}}(x, 42, z.\Omega)))\\
M_9 &\coloneqq \mathtt{fun}(x.\mathtt{match}_{\mathtt{Nat}}(x, 1, y.1))\\
M_{10} &\coloneqq \mathtt{fun}(x.1)
\end{aligned}$$

Donner les couples (M_i, M_j) tels que $M_i \lesssim M_j : \mathtt{Nat} \to \mathtt{Nat}$. Si $M_i \not\lesssim M_j : \mathtt{Nat} \to \mathtt{Nat}$, donner un contexte aussi simple que possible qui sépare les deux termes.

* Exercice 54 Relations d'ordre

Dans ce qui suit, S désigne un ensemble quelconque. On désigne par $\mathsf{POrd}(S)$ et $\mathsf{Ord}(S)$ par

$$\begin{aligned}
\mathsf{POrd}(S) &\coloneqq \{R : S \nrightarrow S \mid R \text{ est une relation de préordre}\}\\
\mathsf{Ord}(S) &\coloneqq \{R : S \nrightarrow S \mid R \text{ est une relation d'ordre}\}.
\end{aligned}$$

Pour chaque énoncé ci-dessous, déterminer s'il est vrai ou faux, et le cas échéant le prouver ou donner un contre-exemple.

1. $\leq_1 \cup \leq_2$ est une relation de préordre sur S.
2. $\leq_1 \cap \leq_2$ est une relation de préordre sur S.
3. Pour tout ensemble S, il existe une relation de préordre maximale sur S.
4. Pour tout ensemble S, il existe une relation d'ordre maximale sur S.

** Exercice 55 Composantes fortement connexes d'un préordre

1. Démontrer que la construction décrite par la définition 3.2.1 construit bien un ensemble ordonné.
2. Démontrer que l'application ι est monotone.

3. Démontrer la propriété universelle de $\mathsf{CC}\,X$, c'est-à-dire la propriété 3.2.2.
4. En déduire que $\mathsf{CC}\,X$ est l'unique ensemble ordonné possédant cette propriété universelle, à isomorphisme d'ordre près. (Un isomorphisme d'ordre est une bijection (f, f^{-1}) telle que f et f^{-1} sont des fonctions monotones.)

* Exercice 56 Treillis

1. Donner l'exemple d'un ensemble ordonné qui n'est pas un treillis.
2. Lister tous les treillis à un, deux, trois, et quatre éléments.
3. Démontrer qu'un treillis fini est nécessairement borné.
4. Donner l'exemple d'un treillis qui n'est pas un treillis borné.
5. Donner l'exemple d'un treillis borné qui n'est pas un treillis complet.

* Exercice 57 L'ordre partiel complet des suites

Liste des exercices

Table des figures

Index

Bibliographie

[1] Henk Barendregt. *The Lambda Calculus : Its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics. College Publications, 2012. isbn : 9781848900660. url : https://books.google.fr/books?id=b8jsMQEACAAJ (cf. p. 81).

[2] Olivier Danvy et Lasse R. Nielsen. « Defunctionalization at work ». In : PPDP '01. Florence, Italy : Association for Computing Machinery, 2001, p. 162-174. isbn : 158113388X. doi : 10.1145/773184.773202. url : https://tidsskrift.dk/brics/article/download/21684/19120 (cf. p. 98).

[3] Brian A. Davey et Hilary A. Priestley. *Introduction to lattices and order*. Cambridge : Cambridge University Press, 1990. isbn : 0521365848. url : http://www.worldcat.org/search?qt=worldcat_org_all&q=0521367662 (cf. p. 32).

[4] Kurt Gödel. « Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes ». In : *Dialectica* 12.3/4 (1958), p. 280-287. issn : 00122017, 17468361. url : http://www.jstor.org/stable/42964248 (cf. p. 85).

[5] Ryu Kashima. *A Proof of the Standardization Theorem in Lambda-Calculus*. Technical Report Research Reports on Mathematical and Computing Sciences, C-145. 2000. url : https://www.kurims.kyoto-u.ac.jp/~kyodo/kokyuroku/contents/pdf/1217-3.pdf (cf. p. 95).

[6] Jean-Louis Krivine. *Lambda-calcul, types et modèles*. Masson, 1990. url : https://www.irif.fr/~krivine/articles/Lambda.pdf (cf. p. 73).

[7] Andrew M. Pitts. *Nominal Sets*. Cambridge University Press, mai 2013. isbn : 9781139084673. doi : 10.1017/cbo9781139084673. url : http://dx.doi.org/10.1017/cbo9781139084673 (cf. p. 81).

[8] Gordon Plotkin. « A Powerdomain Construction ». In : *SIAM Journal on Computing* 5.3 (sept. 1976), p. 452-487. issn : 1095-7111. doi : 10.1137/0205035. url : https://homepages.inf.ed.ac.uk/gdp/publications/Powerdomain_Construction.pdf (cf. p. 126).

[9] W. W. Tait. « Intensional Interpretations of Functionals of Finite Type I ». In : *The Journal of Symbolic Logic* 32.2 (1967), p. 198-212. issn : 00224812. url : http://www.jstor.org/stable/2271658 (visité le 16/01/2023) (cf. p. 85).

[10] Masako Takahashi. « Parallel reductions in -calculus ». In : *Journal of Symbolic Computation* 7.2 (fév. 1989), p. 113-123. issn : 0747-7171. doi : 10.1016/s0747-7171(89)80045-8. url : http://dx.doi.org/10.1016/s0747-7171(89)80045-8 (cf. p. 92).

Table des matières