Institut National Polytechnique de Grenoble
ENSIMAG

Université Joseph Fourier
U.F.R.
Informatique & Mathématiques Appliquées

I.M.A.G.

Laboratoire de Génie Informatique

DEA D'INFORMATIQUE

Projet présenté par:

Catherine ORIAT

Spécification de types abstraits et de structures de contrôle dans un cadre catégoriel

Responsables de projet:
Didier BERT
Marie-Laure POTET

JURY:
Michel ADIBA
Farid OUABDESSELAM
Jean-Claude FERNANDEZ

Juin 1992

Remerciements

Je remercie Didier Bert et Marie-Laure Potet d'avoir accepté la responsabilité de ce projet de DEA.

Je remercie Jean-Claude Fernandez pour sa participation au jury.

Je remercie Jean-Claude Reynaud, qui a répondu à mes questions souvent peu claires et qui a relu ce rapport.

Je remercie également toute l'équipe SCOP et le reste du LGI pour leur accueil sympatique.

Enfin je remercie Hélène et Daniel pour leur soutien dans nos nombreuses "réunions de travail".

Résumé

Les types abstraits jouent un rôle fondamental en spécification des logiciels, puisqu'ils structurent l'information, et constituent la base de la construction et vérification de programmes. La définition de types abstraits est étudiée ici dans deux cadres différents: spécification algébrique et théorie des catégories. L'approche catégorielle des types abstraits est due à Plotkin, Smyth et Wand, et est issue de la théorie des domaines. Notre travail a consisté à mettre en rapport ces deux méthodes de construction de types abstraits, et à utiliser des idées de spécification algébrique dans cette approche catégorielle. Nous avons en particulier introduit la notion d'équation et d'algèbre satisfaisant des équations. Nous avons montré qu'il existe une algèbre initiale dans la classe des algèbres qui satisfont un ensemble d'équations, ce qui permet de donner une sémantique à des types abstraits non libres dans le formalisme des catégories.

Sommaire

Introduction

Spécifier un programme consiste à décrire ce que le programme *doit* faire, sans préciser *comment* sera effectuée la mise en œuvre. La phase de spécification est primordiale dans la conception de logiciels, ne serait-ce que pour donner au programmeur une idée claire du travail à réaliser. Se pose le problème du langage employé pour la spécification. L'avantage d'un langage *formel* par rapport à une langue naturelle, c'est qu'on peut lui donner une sémantique précise. Ainsi ces spécifications peuvent elles-mêmes être traitées par des outils informatiques: on peut ainsi éventuellement vérifier la cohérence du problème posé, vérifier que l'implémentation réalise la spécification, ou même générer automatiquement du code exécutable.

Les *structures* ou *types* de données sont à la base de la programmation et de la spécification formelle, puisque programmer consiste en fait à écrire des fonctions qui manipulent des données. Un type de données ne représente pas seulement un *ensemble de valeurs* que peut prendre une variable, mais également un ensemble d'*opérateurs* qui agissent sur ces données. En spécification, on ne doit pas distinguer deux types conceptuellement identiques mais dont l'implémentation diffère. Les types sont donc considérés comme indépendants de leur représentation, et on parle de *types abstraits*.

Les types abstraits peuvent être étudiés dans plusieurs cadres théoriques. En spécification algébrique, un type abstrait est une *algèbre initiale*. Plus précisément, définir un type algébrique consiste à se donner un ensemble de *sortes* (qui correspondent aux *types* dans les langages de programmation courants), et des *opérateurs* sur ces sortes. On peut également spécifier des *équations* que doivent satisfaire les opérateurs. Lorsque que la spécification d'un type ne comporte pas d'équation, on parle de *type libre* (et de type *non libre* dans le cas contraire).

Une autre théorie permet de spécifier des types abstraits: la théorie des domaines. Cette théorie provient de recherches en sémantique dénotationnelle des langages de programmation. L'idée de départ, due à Scott est de définir des espaces de valeurs (ou *domaines*) pour définir ces fonctions sémantiques. Ces types de données peuvent être *récursifs* c'est-à-dire être définis circulairement: il s'agit typiquement de résoudre une "équation" de domaines de la forme $X \cong F(X)$. La théorie des domaines a été appliquée en λ-calcul pour trouver des modèles du λ-calcul. Ensuite, Plotkin, Smyth, Wand ont repris et généralisé ces idées en se plaçant dans un cadre catégoriel.

Dans ce rapport, nous présentons l'approche catégorielle de construction de types

abstraits due à Plotkin, Smyth et Wand, et nous tentons de la généraliser à des types abstraits non libres.

Dans chapitre 1, nous présentons rapidement la définition de types abstraits par la méthode algébrique. Nous expliquons l'intérêt de la notion d'initialité. Nous parlons également d'applications en construction et preuve de programmes.

Dans le chapitre 2, nous présentons l'approche catégorielle de spécification de types abstraits. Nous rappelons les définitions utiles de théorie des catégories. Enfin nous détaillons un théorème dû à Plotkin et Smyth, qui définit un type abstrait dans le cadre catégoriel.

Dans le chapitre 3, nous montrons que ce théorème peut s'appliquer dans deux catégories particulières: catégorie des ensembles et catégorie des ordres partiels complets. Ce chapitre consiste essentiellement à montrer que les hypothèses du théorème de Plotkin et Smyth sont vérifiées dans ces deux catégories. Cette partie a été étudiée de manière beaucoup plus générale par Wand.

L'approche de Plotkin et Smyth ne permet de spécifier que des types abstraits libres. Dans le chapitre 4, nous introduisons la notion d'équation et d'algèbre satisfaisant des équations dans le cadre catégoriel. Nous montrons l'existence d'une algèbre initiale dans la catégorie des algèbres satisfaisant un ensemble d'équations, ce qui permet de donner un sémantique à des types abstraits non libres.

Chapitre 1

Motivation

1.1 Spécification

Le développement de gros logiciels a montré l'importance de la spécification: spécifier un problème consiste à le décrire sans décider prématurément de la façon dont il sera résolu. Il s'agit de détailler les fonctionnalités d'un programme, sans forcément dire comment les implémenter: c'est une description des structures de données sous forme externe, c'est-à-dire du point de vue de l'utilisateur. Une spécification *formelle* est une spécification pour laquelle on a une sémantique bien définie. A partir d'une telle spécification, on peut vérifier la cohérence du problème posé (c'est-à-dire savoir si on peut effectivement résoudre le problème) et quelquefois générer automatiquement du code exécutable. On distingue la spécification ensembliste (VDM, Z sont des exemples de langages de spécification ensembliste), et la spécification algébrique, à laquelle on s'intéresse plus particulièrement. Le développement de la spécification algébrique est dû principalement au groupe ADJ (Goguen, Thatcher, Wagner), et à Ehrig et Mahr ([$GM85$], [$EM85$]).

1.2 Approche algébrique des types abstraits

Programmer consiste essentiellement à manipuler de l'information. Cette information est structurée sous forme de données. Les *structures* ou *types de données* sont donc à la base de la programmation et de la spécification. Ehrig et Mahr ([$EM85$]) définissent un type de données comme une collection de *domaines de valeurs* (dont certains sont appelés *domaines de base* et contiennent des *éléments de base*), et d'opérations sur ces domaines. De plus, tous les domaines sont des ensembles dénombrables, et tous les éléments des domaines peuvent être générés à partir des éléments de base et des opérations. Cette définition correspond à un type de données "concret", proche de l'implémentation. En spécification, on doit se placer d'un point de vue plus abstrait, moins dépendant de la représentation du type. Ehrig et Mahr définissent donc un *type abstrait* comme une classe de types de données, fermée par renommage des domaines, opérateurs et éléments. D'un point de vue plus pratique, Meseguer et Goguen ([$MG85$]) font remarquer que

définir un *type abstrait* consiste à regrouper dans un seul *module* toutes les fonctions de définition et de manipulation de ce type. L'utilisateur du type de données ne doit avoir à sa disposition que des fonctions "abstraites" de traitement de ce type; il ne doit pas utiliser d'opérations propres à la représentation choisie. De cette manière, un changement éventuel de représentation du type ne concerne qu'une partie très localisée du programme (le module dans lequel on a regroupé tout ce qui définit ce type).

1.2.1 Algèbres multi-sortes

Les types abstraits sont représentés par des algèbres multi-sortes. L'utilisation des algèbres multi-sortes est motivée par l'analogie entre type de données et algèbre: les sortes représentent les domaines de valeurs, et les opérations sur les sortes sont les fonctions entre les domaines de valeurs.

Les définitions qui suivent sont tirées de [$MG85$]. Etant donné un ensemble S de *sortes*, un ensemble indexé par S (ou S-ensemble) est une famille d'ensembles A_s pour chaque $s \in S$. Les éléments de A_s sont *les éléments de sorte* s.

Définition: signature

Une signature Σ est définie par:

- un ensemble de sortes S
- un ensemble OP d'opérateurs indexé par $S^* \times S$.

Soit un opérateur f de sorte $((s_1,\ s_2,\ \cdots s_n),\ s) \in S^* \times S$. On note: $f : (s_1,\ s_2,\ \cdots s_n) \rightarrow s$. On utilise cette notation parce que f sera ensuite interprété comme une fonction qui prend un argument de "type" $(s_1,\ s_2,\ \cdots,\ s_n)$, et retourne un argument de "type" s.

Un exemple très classique de signature est la signature *Nat*:
$S = \{nat\}$
$OP = \{0, s\}$

```
0: -> nat
s: nat -> nat
```

Définition: algèbre

Soit une signature Σ. Définir une algèbre sur Σ consiste à *interpréter* les sortes par des ensembles, et les opérateurs par des fonctions entre ces ensembles. Plus formellement, une Σ-algèbre $\mathbf{A}$ est définie par:

- Pour chaque sorte $s \in S$, on a un ensemble A_s. A_s est le *domaine d'interprétation de* s.

- Pour chaque opérateur $f : (s_1,\ s_2,\ \cdots,\ s_n) \to s\ \ \in OP$, on donne une *fonction d'interprétation* $f^A : A_{s_1} \times A_{s_2} \cdots \times A_{s_n} \to A_s$.

Exemples classiques d'algèbres sur la signature *Nat*:

- L'algèbre des entiers naturels:
  La sorte `nat` est interprétée comme l'ensemble des entiers naturels $\mathbb{N}$.
  L'interprétation de `0: -> nat` est l'entier naturel $0 \in \mathbb{N}$.
  L'interprétation de `s: nat -> nat` est la fonction successeur dans les entiers naturels: $succ : \mathbb{N} \to \mathbb{N}$.

- L'algèbre des entiers modulo 2:
  L'interprétation de `nat` est l'ensemble $\mathbf{Z/2Z} = \{0, 1\}$.
  L'interprétation de `0: -> nat` est $0 \in \mathbf{Z/2Z}$.
  L'interprétation de `s: nat -> nat` est la fonction successeur $succ : \mathbf{Z/2Z} \to \mathbf{Z/2Z}$.

Définition: homomorphisme d'algèbre

Soit deux Σ-algèbres $\mathbf{A}$ et $\mathbf{B}$. Un homomorphisme h de $\mathbf{A}$ vers $\mathbf{B}$ est une famille de fonctions $\{h_s : A_s \to B_s,\ s \in S\}$, telle que:

$$\forall f : (s_1,\ s_2,\ \cdots,\ s_n) \to s\ \ \in OP,\quad \forall a_i \in A_{s_i}\ (i = 1, \cdots, n) :$$

$$h_s(f^A(a_1, a_2, \cdots, a_n)) = f^B(h_{s_1}(a_1), h_{s_2}(a_2), \cdots, h_{s_n}(a_n))$$

Un homomorphisme est une fonction qui à tout élément de sorte s de $\mathbf{A}$ associe un élément de sorte s de $\mathbf{B}$, et qui conserve la structure des éléments des deux algèbres.

Définition: algèbre des termes

Soit X un S-ensemble de *variables*. L'ensemble des termes $T_\Sigma[X]$ est un S-ensemble: pour chaque sorte $s \in S$, l'ensemble $T_\Sigma[X]_s$ des termes de sorte s est le plus petit ensemble tel que:

- $x \in X_s \Rightarrow x \in T_\Sigma[X]_s$
- $c :\to s \in OP \Rightarrow c \in T_\Sigma[X]_s$
- $f : (s_1,\ s_2,\ \cdots,\ s_n) \to s,\ \ t_i \in T_\Sigma[X]_{s_i},\ i = 1, \cdots, n$
  $\Rightarrow f(t_1, t_2, \cdots, t_n) \in T_\Sigma[X]_s$

L'ensemble $T_\Sigma[X]$ est muni d'une structure d'algèbre:

- L'interprétation d'une sorte $s \in S$ est $T_\Sigma[X]_s$.

- L'interprétation d'un opérateur $f : (s_1,\ s_2,\ \cdots,\ s_n) \to s\ \in OP$ est la fonction:

$$\begin{array}{rccl} f^T : & T_\Sigma[X]_{s_1} \times T_\Sigma[X]_{s_2} \times \cdots \times T_\Sigma[X]_{s_n} & \to & T_\Sigma[X]_s \\ & (t_1,\ t_2,\ \cdots,\ t_n) & \mapsto & f(t_1, t_2, \cdots, t_n) \end{array}$$

$T_\Sigma = T_\Sigma[\emptyset]$ est l'ensemble des termes *fermés* sur Σ.

L'algèbre $T_\Sigma[X]$ est *libre* dans la classe des Σ-algèbres:
Soit l'inclusion $i : X \to T_\Sigma[X]$. Soit une algèbre $\mathbf{A}$. Soit une *affectation* $aff : X \to A$, qui à tout élément de X_s associe un élément de A_s.
Il existe un unique homomorphisme $\overline{aff} : T_\Sigma[X] \to A$ qui *étend* aff à $T_\Sigma[X]$, c'est-à-dire tel que: $\overline{aff} \circ i = aff$.

Définition: algèbre initiale

Une algèbre $\mathbf{I}$ est initiale dans une classe de Σ-algèbres, si pour toute Σ-algèbre $\mathbf{A}$ de cette classe, il existe un unique homomorphisme h_A de $\mathbf{I}$ vers $\mathbf{A}$.

L'algèbre des termes fermés T_Σ est initiale: pour toute algèbre $\mathbf{A}$, il existe un unique homomorphisme de T_Σ vers $\mathbf{A}$.

Deux algèbres initiales sont isomorphes.

Equations

Etant donné une signature Σ, T_Σ représente un type abstrait *libre*: T_Σ est l'ensemble des valeurs que peut prendre un élément de ce type. Aucune contrainte n'est imposée sur les fonctions qui manipulent ces éléments. En spécification algébrique, on peut spécifier des *équations* que ces fonctions doivent satisfaire. Certains types abstraits comme le type *ensemble d'entiers* ne peuvent pas être spécifiés sans équations.

On fixe un ensemble de variables X. Une équation consiste en deux termes $t_1, t_2 \in T_\Sigma[X]$. Cette équation est notée $(t_1 = t_2)$. On dit qu'une algèbre $\mathbf{A}$ satisfait l'équation $(t_1 = t_2)$ si pour toute affectation $aff : X \to A$, $\overline{aff}(t_1) = \overline{aff}(t_2)$.

Etant donné un ensemble E d'équations, une Σ-algèbre $\mathbf{A}$ est une Σ-E-algèbre si $\mathbf{A}$ satisfait toutes les équations de E.

Exemple:
Pour la signature *Nat*, on considère l'équation:
`s(s(s(s(x)))) = x`.
(L'ensemble des variables est ici: $X = \{x\}$)
Les algèbres $\mathbf{Z/2Z}$, $\mathbf{Z/4Z}$ sont des Σ-E-algèbres. $I\!N$ n'est pas une Σ-E-algèbre.

Algèbre quotient

Soit un ensemble d'équations E. Soit $\equiv_E$ la plus petite congruence telle que: pour toute équation $(t_1 = t_2) \in E$, pour toute affectation $aff : X \to T_\Sigma[X]$, $\overline{aff}(t_1) \equiv_E \overline{aff}(t_2)$.

Autement dit, $\equiv_E$ est la plus petite relation telle que:

- $\forall (t_1 = t_2) \in E,\ \forall aff : X \to T_\Sigma[X] : \overline{aff}(t_1) \equiv_E \overline{aff}(t_2)$
- $\forall t \in T_\Sigma[X] :\ t \equiv_E t$
- $\forall t_1, t_2 \in T_\Sigma[X] :\ t_1 \equiv_E t_2 \Rightarrow t_2 \equiv_E t_1$
- $\forall t_1, t_2, t_3 \in T_\Sigma[X] :\ t_1 \equiv_E t_2,\ t_2 \equiv_E t_3 \Rightarrow t_1 \equiv_E t_3$
- $\forall f :\ (s_1,\ s_2,\ \cdots,\ s_n) \to s,\ \forall t_1, t'_1, \cdots, t_n, t'_n \in T_\Sigma[X] :$
  $\forall i \in \{1, \cdots, n\},\ t_i \equiv_E t'_i\ \Rightarrow f(t_1, t_2, \cdots, t_n) \equiv_E f(t'_1, t'_2, \cdots, t'_n)$

On considère l'ensemble quotient $T_\Sigma[X]/\equiv_E$. Cet ensemble a une structure d'algèbre, appelée ici **Q**:
Une sorte $s \in S$ est interprétée par l'ensemble des classes d'équivalence des termes de sorte s : $Q_s = \{[t],\ t \in T_\Sigma[X]_s\}$.
Soit $f : (s_1,\ s_2,\ \cdots,\ s_n) \to s\ \in OP$.
f^Q est défini par:

$$\begin{array}{rccl} f^Q : & Q_{s_1} \times Q_{s_2} \times \cdots \times Q_{s_n} & \to & Q_s \\ & ([t_1], [t_2], \cdots, [t_n]) & \mapsto & [f(t_1, t_2, \cdots, t_n)] \end{array}$$

Cela définit bien une fonction car:
$\forall f :\ (s_1,\ s_2,\ \cdots,\ s_n) \to s,\ \forall t_1, t'_1, \cdots, t_n, t'_n \in T_\Sigma[X] :$
$\forall i \in \{1, \cdots, n\},\ t_i \equiv_E t'_i\ \Rightarrow f(t_1, t_2, \cdots, t_n) \equiv_E f(t'_1, t'_2, \cdots, t'_n)$

L'algèbre $T_\Sigma[X]/\equiv_E$ est libre dans la classe des Σ-E-algèbres.
L'algèbre $T_\Sigma/\equiv_E$ est initiale dans la classe des Σ-E-algèbres.

1.2.2 Sémantique initiale

Il est naturel de représenter un type abstrait par une algèbre, parce qu'un type et une algèbre ont la même structure: les sortes sont des ensembles de valeurs, et les opérations sont des fonctions entre ces ensembles. La sémantique précise d'un type abstrait défini par une signature Σ et un ensemble d'équations E est la classe des algèbres initiales parmi les Σ-E-algèbres. Le choix de cette sémantique est motivé par deux remarques:

- On peut construire tous les éléments d'une algèbre initiale à partir des constantes et des opérateurs de l'algèbre. Elle contient donc le nombre minimal d'éléments. Cela permet de raisonner par induction sur la structure des termes. Cette propriété est résumée par l'expression *"no junk"* ("pas de superflu": l'algèbre ne contient pas de termes autres que ceux auquels on s'intéresse).

- Si deux éléments sont égaux, on peut le prouver en utilisant les équations: aucune propriété supplémentaire n'a été introduite par rapport à celles spécifiées par les équations. Une algèbre initiale est une algèbre qui satisfait le nombre minimal d'équations. Cette propriété est résumée par l'expression *"no confusion"*.

La sémantique du type abstrait est la classe des algèbres initiales parmi les Σ-E-algèbres. Cette classe est la classe des algèbres isomorphes à $T_\Sigma/\equiv_E$. $T_\Sigma/\equiv_E$ représente une implémentation du type abstrait. Prendre pour sémantique d'un type abstrait toute la classe des algèbres initiales revient à dire que la signification du type est indépendante de sa représentation.

1.2.3 Exemples

On va donner quelques exemples simples de spécifications de types abstraits.

Entiers naturels avec fonctions successeur et addition

Signature:

```
0: -> nat
s: nat -> nat
+: (nat,nat) -> nat
```

Equations:

```
0+y = y
s(x)+y = s(x+y)
```

Dans cette exemple, la fonction d'addition `+` est en fait entièrement déterminée par les équations, et tout terme de $T_\Sigma/\equiv$ admet un représentant qui ne contient pas l'opérateur `+`

Listes d'entiers

Signature:

```
nil: -> liste
cons: (nat,liste) -> liste
```

Ici, le type `liste` est libre. On peut ajouter la fonction de concaténation `append`:

```
append: (liste,liste) -> liste

append(nil,l) = l
append(cons(n,l),m) = cons(n,append(l,m))
```

On peut faire la même remarque que pour les entiers naturels: on peut toujours réduire une liste à un terme qui ne contient pas de `append`.

Encore des listes d'entiers

On peut spécifier les listes d'entiers naturels d'une autre façon: une liste est soit la liste vide (`nil`), soit une liste à un élément (`one(n)`), soit la concaténation de deux listes (`append`).

Signature:

```
nil: -> liste
one: nat -> liste
append: (liste,liste) -> liste
```

Equations:

```
append(nil,x) = x
append(x,nil) = x
append(append(x,y),z) = append(x,append(y,z))
```

Dans cet exemple, on ne peut pas éliminer la fonction `append` de tous les termes.

Multi-ensemble d'entiers

Signature:

```
vide: -> mens
ins:  (nat,mens) -> mens
```

Equations:

```
ins(x,ins(y,m)) = ins(y,ins(x,m))
```

$T_\Sigma/\equiv_E$ est l'algèbre qui représente le type abstrait des multi-ensembles d'entiers. On considère une autre algèbre $\mathbf{A}$ sur cette signature:
L'interprétation de `nat` et `mens` est l'ensemble des entiers naturels $\mathbb{N}$.
L'interprétation de `vide` est $0 \in \mathbb{N}$.
L'interprétation de `ins` est l'addition: $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.
Comme l'addition est associative, l'algèbre $\mathbf{A}$ satisfait l'équation:

```
ins(x,ins(y,m)) = ins(y,ins(x,m))
```

$\mathbf{A}$ est donc une Σ-E-algèbre.

$T_\Sigma/\equiv_E$ est initiale dans la catégorie des Σ-E-algèbres, donc il existe un unique homomorphisme h de $T_\Sigma/\equiv_E$ vers $\mathbf{A}$. Autrement dit, il existe un unique h tel que:

```
h: mens -> nat
h(vide)     = 0
h(ins(x,m)) = x+h(m)
```

Cette fonction h calcule la somme des éléments d'un multi-ensemble d'entiers. On reconnait ici un style de programmation fonctionnelle. Cependant, comme le type multi-ensemble est un type non libre, l'existence et l'unicité d'une telle fonction ne sont pas évidentes.

Ensemble d'entiers

Pour spécifier un ensemble d'entiers, il suffit de reprendre la spécification d'un multi-ensemble et d'ajouter une équation qui indique qu'insérer deux fois le même élément dans un ensemble revient à l'insérer une seule fois.

```
vide: -> ens
ins: (nat,ens) -> ens

ins(x,ins(y,e)) = ins(y,ins(x,e))      (1)
ins(x,ins(x,e)) = ins(x,e)             (2)
```

Si on veut calculer la somme des éléments d'un ensemble, on peut être tenté d'écrire la même fonction h que précédemment:

```
h: ens -> nat
h(vide)     = 0                        (H1)
h(ins(x,m)) = x+h(m)                   (H2)
```

Cette définition n'est pas correcte: en effet, on a par exemple:

```
ins(1,ins(1,vide)) = ins(1,vide)
h(ins(1,ins(1,vide))) = 1+h(ins(1,vide))
                      = 1+1+h(vide)
                      = 1+1+0
                      = 2
h(ins(1,vide)) = 1+h(vide)
               = 1+0
               = 1
```

On aboutit donc à une contradiction: $1 = 2$.
Cela provient du fait que l'algèbre **A** ne satisfait pas l'équation `(2)`. En effet, pour $x, e \in \mathbb{N}$, l'égalité $x + x + e = x + e$ n'est pas valide. **A** n'est donc pas une Σ-E-algèbre, et ici il n'existe pas de fonction h qui satisfait les deux égalités `(H1)` et `(H2)`.

1.3 Autres approches des types abstraits

La spécification de types abstraits à l'aide d'algèbres présente un inconvénient: l'interprétation d'une sorte est un ensemble, et l'interprétation d'un opérateur une application, ce qui ne permet pas de modéliser tous les comportements des objets informatiques. En particulier, on ne peut pas représenter de fonctions partielles ou la non-terminaison d'un programme. En sémantique des langages de programmation, par exemple, on utilise souvent des ordres partiels complets (CPO), qui permettent de représenter les différentes étapes d'un calcul comme une séquence d'objets contenant une quantité croissante d'information.

Il existe d'autres théories qui permettent de spécifier des types abstraits en tenant compte de ces problèmes. Il est possible de généraliser l'approche algébrique en interprétant les sortes par des CPO, et les opérateurs par des fonctions continues entre CPO ([$Wec92$]). On peut également se placer dans un cadre catégoriel, comme pour la théorie des esquisses (cf [$BW91$], [$DR91$]). Toujours dans un cadre catégoriel, les travaux de Scott, Plotkin, Smyth, Wand permettent également de définir des types abstraits. Cette approche est issue des travaux de Scott sur les modèles du λ-calcul et sur la recherche de solutions à l'"équation" $D \cong [D \to D]$. Elle consiste à définir des types abstraits à l'aide d'"équations" de domaines. (Cette construction est présentée dans le chapitre 2).

1.4 Application en programmation

La spécification de types abstraits est à la base de la programmation. Dans cette partie, nous donnons quelques indications bibliographiques d'applications en construction et vérification de programmes.

On observe que la définition de fonctions simples est fondée sur la structure du type de l'argument ou du type du résultat de ces fonctions. Autrement dit, beaucoup de fonctions peuvent être considérées comme des homomorphismes entre deux algèbres adéquates.

Von Henke ([$Hen75$]) est peut-être le premier à avoir eu l'idée de programmer des fonctions à l'aide d'homomorphismes. Son approche est basée sur la définition de types abstraits à partir d'équations de domaines, et il utilise le langage de Milner LCF pour représenter types et fonctions.

Plus récemment, Malcolm ([$Mal89$],[$Mal90$]), Jeuring ([$Jeu91$]), Meijer, Fokkinga et Paterson ([$MFP91$]) ont développé l'idée de programmation structurée par les types de données d'une façon plus formelle et systématique. Tous ces articles ont pour point de départ ce qu'on appelle *le formalisme de Bird-Meertens*. Malcolm définit ce formalisme: "ce formalisme consiste en des notations fonctionnelles très consises, et en un petit nombre de théorèmes remarquablement puissants pour prouver des égalités entre fonctions" ([$Mal90$]). Au départ, Bird et Meertens se sont concentrés sur le type des listes, puis les théorèmes ont été étendus à d'autres types de données.

Pour prouver des égalités entre fonctions, Bird utilise le *théorème de promotion*. Dans le cadre de la spécification algébrique, ce théorème découle directement de l'initialité de l'algèbre qui représente le type abstrait. Avec les notations algébriques, cela donne:
Soit deux algèbres $\mathbf{A}$ et $\mathbf{B}$. Soit $h_A : T_\Sigma \to A$ et $h_B : T_\Sigma \to B$ les deux uniques homomorphismes de T_Σ vers $\mathbf{A}$ et $\mathbf{B}$. Soit $f : A \to B$. On a:

$$f \text{ est un homomorphisme} \Rightarrow \ f \circ h_A = h_B$$

Malcolm d'une part, et Meijer, Fokkinga, Paterson d'autre part, proposent une généralisation du travail initial de Bird et Meertens, afin d'obtenir des théorèmes pour des types de données *quelconques*. Pour cela, ils se placent dans le cadre de la théorie des catégories.

Le cadre théorique de Meijer, Fokkinga et Paterson est fourni par les travaux de Plotkin, Smyth et Wand ([$SP77$], [$Wan79$]): les types abstraits sont définis comme des

plus petits points fixes d'*endofoncteurs* dans une catégorie. Cette construction est une généralisation pour les catégories du théorème du point fixe de Tarski dans des ordres partiels complets. Cette construction est explicitée dans le chapitre 2.

Il y a des différences entre l'approche de Meijer, Fokkinga et Paterson, et l'approche de Malcolm:

- Malcolm travaille dans la catégorie des ensembles **SET**, alors que Meijer, Fokkinga et Paterson travaillent dans la catégorie **CPO** des ordres partiels complets avec fonctions continues. Malcolm ne peut donc pas définir de structures infinies (qui peuvent être obtenues dans la catégorie **CPO** comme bornes supérieures d'objets finis). Par contre, par un résultat de dualité, Malcolm peut construire des types infinis comme coalgèbres terminales dans la classe des coalgèbres. Cette construction ne définit pas de nouveaux types dans la catégorie **CPO**.

- Chez Malcolm, la propriété d'initialité est fondamentale: c'est à partir de cette propriété que sont dérivés tous les théorèmes, en particulier le théorème de promotion. Chez Meijer, Fokkinga et Paterson, l'initialité ne joue aucun rôle: les types abstraits sont définis comme chez Malcolm (en se plaçant dans la catégorie **CPO** au lieu de se placer dans **SET**). Toutes les fonctions intéressantes sont ensuites construites comme des plus petits points fixes de fonctions dans des ordres partiels complets. Toutes les théorèmes qui sont alors dérivés proviennent des propriétés de cet opérateur de plus petit point fixe dans les CPO, et non de l'initialité d'une certaine algèbre.

Nous n'entrons pas dans les détails maintenant, d'une part parce qu'il faudrait introduire beaucoup de notations, d'autre part parce que nous allons en présenter les fondements théoriques dans les deux prochains chapitres. Nous reviendrons donc sur les différences entre ces deux approches en 3.4.

Le cadre théorique de Plotkin, Smyth et Wand ne permet pas de définir des types abstraits *non libres*, (c'est-à-dire dont la spécification comporte des équations). L'idée de ce travail est de mettre en rapport les approches algébrique et catégorielle. Dans le cadre catégoriel, l'équivalent d'une signature est un endofoncteur, et il est possible d'introduire des équations, afin de spécifier des types abstraits non libres. Nous montrons dans le chapitre 4 qu'il existe une algèbre initiale dans la catégorie des algèbres satisfaisant un ensemble d'équations. Cela donne le moyen de définir des fonctions à l'aide d'homomorphismes, et de prouver des propriétés sur ces fonctions.

D'autres approches permettent de définir des types abstraits non libres. En théorie des esquisses, on peut écrire des équations logiques (cf 4.5). D'autre part, Wechler ([*Wec*92]) introduit des *inéquations* dans le cadre de la spécification algébrique avec algèbres ordonnées, ce qui semble représenter une extension par rapport à la notion d'équation.

Chapitre 2

Approche catégorielle

2.1 Introduction

Dans ce chapitre, on se propose d'étudier les types abstraits dans le formalisme des catégories. Ces travaux sont dus essentiellement à Scott, Plotkin, Smyth et Wand. Cette théorie est issue de théorie des domaines et de recherche de modèles pour le λ-calcul réalisées par Scott. Plotkin, Smyth et Wand ont ensuite utilisé et généralisé ces idées pour définir des types abstraits dans le cadre des catégories.

Le formalisme des catégories permet d'étudier les types abstraits d'une manière plus globale: On définit un type abstrait comme une construction dans une catégorie possédant certaines propriétes. Cette construction peut alors s'appliquer dans différentes catégories: par exemple dans la catégorie des ensembles, ou dans une catégorie de CPO.

Les deux concepts de base en théorie des catégories sont l'*objet* et la *flèche*. Dans la catégorie des ensembles, un objet représente un *ensemble* et une flèche représente une *application* entre deux ensembles. L'idée centrale de ce formalisme est que toutes les propriétés intéressantes s'expriment avec les flèches. En informatique, et plus particulièrement pour l'étude des langages fonctionnels typés, les objets représentent les *types* et les flèches les *fonctions* entre deux types. Dans ces langages, tout est défini en termes de fonctions. Le formalisme des catégories s'applique donc naturellement puisque tous les raisonnements se font sur les flèches.

Enfin, le formalisme des catégories utilise des notations très courtes. On peut définir par exemple une algèbre, un homomorphisme d'algèbre de façon plus compacte qu'avec l'approche algébrique.

Dans ce chapitre, nous présentons d'abord quelques concepts de théorie des catégories. Ensuite, nous définissons dans ce formalisme les notions d'algèbre et d'homomorphisme d'algèbre. Enfin nous présentons un théorème de Plotkin et Smyth. Ce théorème définit un type abstrait comme le plus petit point fixe d'un endofoncteur, et montre que ce plus petit point fixe correspond à une algèbre initiale.

2.2 Généralités sur les catégories

2.2.1 Catégorie

Définition: catégorie

On définit une catégorie $\mathbf{C}$ de la façon suivante:
On considère une classe d'*objets*, notés $A, B, C...$
Pour chaque couple d'objets (A, B), on a un ensemble de *flèches* $\mathbf{C}(A, B)$. $f \in \mathbf{C}(A, B)$ est noté: $f : A \to B$.
Pour chaque triplet d'objets (A, B, C), on a une fonction de composition:

$$\begin{array}{rccc} \circ: & \mathbf{C}(B,C)\times\mathbf{C}(A,B) & \to & \mathbf{C}(A,C) \\ & (g, f) & \mapsto & g \circ f \end{array}$$

$\mathbf{C}$ est une catégorie si:

- $\forall f : A \to B,\ g : B \to C,\ h : C \to D$, on a:
  $(h \circ g) \circ f = h \circ (g \circ f)$ (associativité de la composition)
- Pour tout objet B, il existe une flèche $id_B : B \to B$ telle que:
  $\forall f : A \to B,\ g : B \to C$, on a:
  $id_B \circ f = f$ et $g \circ id_B = g$.

Exemples de catégories:

- La catégorie **SET** dont les objets sont les ensembles et les flèches les applications entre deux ensembles.
- La catégorie **CPO** dont les objets sont les CPO et les flèches les applications continues entre CPO.
- La catégorie **CPOS** dont les objets sont les CPO et les flèches les applications continues *strictes* entre CPO.
- Soit une signature Σ. La classe des Σ-algèbres forme une catégorie dont les objets sont les Σ-algèbres et les flèches sont les homomorphismes de Σ-algèbres.

Langage fonctionnel en tant que catégorie:

Soit un langage de programmation fonctionnel typé L. On peut lui associer une catégorie $\mathbf{C}$:

- Les objets de $\mathbf{C}$ sont les *types* de L.
- Les flèches $f : A \to B$ de $\mathbf{C}$ sont les *fonctions* définissables dans le langage L, qui prennent un argument de type A et retournent un argument de type B.

On définit ansi une catégorie:

- La composition des flèches dans **C**, qui correspond à la composition des fonctions dans L, est associative.

- Pour chaque type de donnée, il existe une fonction de L qui retourne son argument sans le modifier, donc:
  Pour tout objet B, il existe une flèche $id_B : B \to B$ telle que:
  $\forall f : A \to B,\ g : B \to C$, on a:
  $id_B \circ f = f$ et $g \circ id_B = g$.

Définition: isomorphisme

Une flèche $f : A \to B$ est un isomorphisme s'il existe une flèche $g : B \to A$ telle que: $f \circ g = id_B$ et $g \circ f = id_A$.
Les deux objets A et B sont alors *isomorphes*.

Définition: objet initial, objet terminal

Un objet I est initial si pour tout objet A, il existe une flèche unique $j_A : I \to A$.
Un objet T est terminal si pour tout objet A, il existe une flèche unique $t_A : A \to T$.

Deux objets initiaux A et B sont isomorphes:
Comme B est initial, il existe un unique $j_A : B \to A$. De même, comme A est initial, il existe un unique $j_B : A \to B$.
La flèche $j_A \circ j_B : A \to A$ est unique, puisque A est initial, donc $j_A \circ j_B = id_A$.
La flèche $j_B \circ j_A : B \to B$ est unique, puisque B est initial, donc $j_B \circ j_A = id_B$.

De même, deux objets terminaux sont isomorphes.

Exemples

- Dans **SET**, $\emptyset$ est l'objet initial. L'ensemble $1 = \{*\}$ est terminal. Tout autre ensemble à un élément est isomorphe à 1.

- Dans **CPO**, $\{\perp\}$ est terminal, mais pas initial.

- Dans **CPOS**, $\{\perp\}$ est à la fois initial et terminal.

2.2.2 Foncteurs et bifoncteurs

Définition: foncteur

Soit deux catégories **C** et **D**. Un foncteur F de **C** dans **D** est une fonction qui associe à tout objet A de **C**, un objet $F(A)$ de **D**, et à toute flèche $f : A \to B$ de **C**, une flèche $F(f) : F(A) \to F(B)$ de **D**, tel que:

- $\forall f : A \to B,\ g : B \to C$, on a: $F(g \circ f) = F(g) \circ F(f)$

- Pour tout objet A de **C**, on a: $F(id_A) = id_{F(A)}$

Exemples de foncteurs:

- **Foncteur constant:**
  Soit un objet Z. Le foncteur constant $Z\bullet$ est défini par:
  Pour tout objet A, $(Z\bullet)(A) = Z$
  Pour toute flèche $f : A \to B$, $(Z\bullet)(f) = id_Z$.

- **Foncteur identité:**
  Le foncteur identité $I\!I$ est défini par:
  Pour tout objet A, $I\!I(A) = A$
  Pour toute flèche $f : A \to B$, $I\!I(f) = f$.

Catégorie produit:

Etant donné deux catégories **C** et **C'**, on peut définir la catégorie produit **C** $\times$ **C'**. Les objets de **C** $\times$ **C'** sont des couples (A, A'), où A est un objet de **C** et A' un objet de **C'**. Les flèches de **C** $\times$ **C'** sont des couples (f, f'), où f est une flèche de **C** et f' une flèche de **C'**. La composition des flèches est définie par: $(g, g') \circ (f, f') = (g \circ f, g' \circ f')$.

Définition: bifoncteur

Soit trois catégories **C**, **C'** et **D**. Un bifoncteur † de **C** et **C'** dans **D** est un foncteur de **C** $\times$ **C'** dans **D**. On note les bifoncteurs de manière infixée: $\dagger(A, A')$ est noté $A \dagger A'$, et $\dagger(f, f')$ est noté $f \dagger f'$.

† doit donc vérifier le propriétés:

- Soit A, B, C des objets de **C**, et A', B', C' des objets de **C'**.
  $\forall f : A \to B, g : B \to C, f' : A' \to B', g' : B' \to C'$, on a:
  $(g \circ f) \dagger (g' \circ f') = (g \dagger g') \circ (f \dagger f')$

- Pour tout objet A de **C**, et A' de **C'**, on a:
  $id_A \dagger id_{A'} = id_{A \dagger A'}$

Définition: endofoncteur

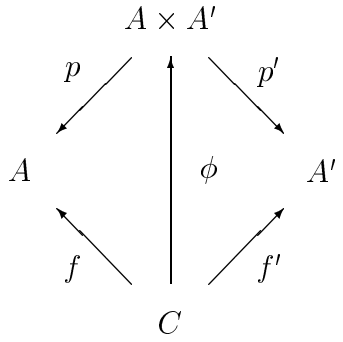
Un **endofoncteur** est un foncteur d'une catégorie sur elle-même.

2.2.3 Produit et somme

Définition: produit de deux objets

Soit deux objets A et A' dans une catégorie **C**. Le produit de A et A' est la donnée d'un objet $A \times A'$ et de deux flèches $p : A \times A' \to A$ et $p' : A \times A' \to A'$ tels que:

Pour tout objet C, pour tout $f : C \to A$ et $f' : C \to A'$, il existe une unique flèche $\phi : C \to A \times A'$ telle que: $p \circ \phi = f$ et $p' \circ \phi = f'$.



$p : A \times A' \to A$ et $p' : A \times A' \to A'$ sont les *projections* du produit.
ϕ est souvent noté $< f, f' >$, ou $f \Delta f'$.

Si cette construction est possible pour tous les objets de la catégorie **C**, on dit que *la catégorie a des produits.*

Exemple:

Dans la catégorie **SET**, le produit catégoriel correspond au produit cartésien.

Définition: produit de deux flèches

On se place dans une catégorie ayant des produits. On vient de définir le produit de deux objets, on définit maintenant le produit de deux flèches.
Soit les objets A, B, A', B'. On considère les produits $A \times A'$ et $B \times B'$.
Soit $p : A \times A' \to A$, $p' : A \times A' \to A'$, $q : B \times B' \to B$, $q : B \times B' \to B'$ les projections correspondantes.
Le produit de deux flèches $f : A \to B$ et $f' : A' \to B'$ est l'unique flèche $f \times f' : A \times A' \to B \times B'$ telle que:
$q \circ (f \times f') = f \circ p$ et $q' \circ (f \times f') = f' \circ p'$.

Autrement dit: $f \times f' =< f \circ p, f' \circ p' >$.

$$\begin{array}{ccccc} & & \overset{p}{\longleftarrow} & & \overset{p'}{\longrightarrow} \\ A & & A \times A' & & A' \\ f \downarrow & & \downarrow f \times f' & & \downarrow f' \\ B & & B \times B' & & B' \\ & & \underset{q}{\longleftarrow} & & \underset{q'}{\longrightarrow} \end{array}$$

Résultat:

$\times$ est un bifoncteur.

Preuve:

Il suffit de montrer que:

1. Pour toutes flèches $f : A \to B$, $g : B \to C$, $f' : A' \to B'$ et $g' : B' \to C'$, on a: $(g \circ f) \times (g' \circ f') = (g \times g') \circ (f \times f')$.
2. Pour tout objet A, A': $id_A \times id_{A'} = id_{A \times A'}$.

1. Soit p et p', les projections associées à $A \times A'$, q et q', les projections associées à $B \times B'$, et r et r', les projections associées à $C \times C'$.

$$\begin{array}{ccccc}
 & \overset{p}{\longleftarrow} & & \overset{p'}{\longrightarrow} & \\
A & & A \times A' & & A' \\
\Big\downarrow{\scriptstyle f} & & \Big\downarrow{\scriptstyle f \times f'} & & \Big\downarrow{\scriptstyle f'} \\
B & \underset{q}{\longleftarrow} & B \times B' & \underset{q'}{\longrightarrow} & B' \\
\Big\downarrow{\scriptstyle g} & & \Big\downarrow{\scriptstyle g \times g'} & & \Big\downarrow{\scriptstyle g'} \\
C & \underset{r}{\longleftarrow} & C \times C' & \underset{r'}{\longrightarrow} & C'
\end{array}$$

Par définition, $(g \circ f) \times (g' \circ f')$ est l'unique flèche ϕ telle que:
$r \circ \phi = g \circ f \circ p$ et $r' \circ \phi = g' \circ f' \circ p'$.

$$\begin{array}{lll}
r \circ (g \times g') \circ (f \times f') & = g \circ q \circ (f \times f') & \text{(définition de } g \times g') \\
 & = g \circ f \circ p & \text{(définition de } f \times f')
\end{array}$$

De même:

$$\begin{array}{lll}
r' \circ (g \times g') \circ (f \times f') & = g' \circ q' \circ (f \times f') & \text{(définition de } g \times g') \\
 & = g' \circ f' \circ p' & \text{(définition de } f \times f')
\end{array}$$

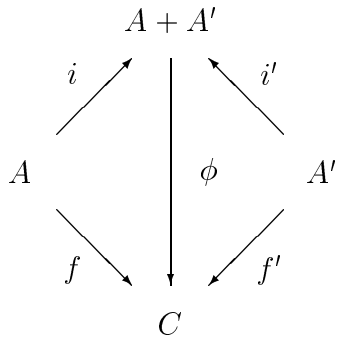
Par conséquent: $(g \circ f) \times (g' \circ f') = (g \times g') \circ (f \times f')$.

2. Par définition, $id_A \times id_{A'}$ est l'unique flèche telle que:
$p \circ (id_A \times id_{A'}) = p$ et $p' \circ (id_A \times id_{A'}) = p'$.
$id_{A \times A'}$ satisfait ces deux égalités, donc: $id_A \times id_{A'} = id_{A \times A'}$.

Le bifoncteur *somme* est défini de façon duale par rapport au bifoncteur produit, c'est-à-dire en retournant toutes les flèches.

Définition: somme de deux objets

Soit deux objets A et A'. La somme de A et A' est la donnée d'un objet $A + A'$ et de deux flèches $i : A \to A + A'$ et $i' : A' \to A + A'$ tels que:
Pour tout objet C, pour tout $f : A \to C$ et $f' : A' \to C$, il existe une unique flèche $\phi : A + A' \to C$ telle que: $\phi \circ i = f$ et $\phi \circ i' = f'$.



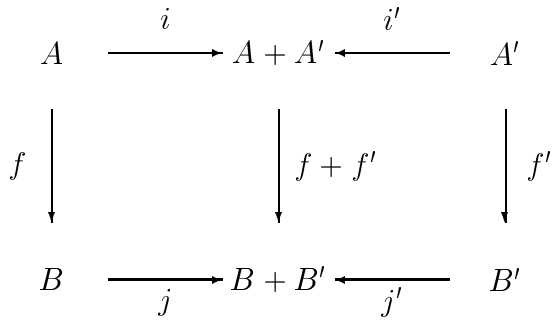
ϕ est noté $f \nabla f'$.
Les flèches i et i' sont appelées *injections*; ce sont effectivement des injections au sens habituel dans la catégorie **SET**.

Exemple:

Dans la catégorie **SET**, la somme catégorielle correspond à l'union disjointe de deux ensembles.
L'union disjointe de A et A' consiste à "marquer" chaque objet de A et chaque objet de A', et à faire l'union:
$A + A' = (\{1\} \times A) \cup (\{2\} \times A')$.

Définition: somme de deux flèches

On se place dans une catégorie ayant des sommes. On considère le diagramme suivant:



$f + f'$ est l'unique flèche $f + f' : A + A' \to B + B'$ telle que:
$(f + f') \circ i = j \circ f$ et $(f + f') \circ i' = j' \circ f'$.

Autrement dit: $f + f' = (j \circ f) \nabla (j' \circ f')$.

Résultat:

$+$ est un bifoncteur.
Autrement dit, on a les égalité suivantes:
$(g \circ f) + (g' \circ f') = (g + g') \circ (f + f')$
$id_A + id_{A'} = id_{A+A'}$

Preuve:

La preuve est semblable à celle qui montre que $\times$ est un bifoncteur.

2.2.4 Limites et colimites

Définition: diagramme

Soit I un ensemble d'index. Un diagramme **D** dans une catégorie **C** est un graphe dont les nœuds $i \in I$ sont étiquetés par des objets D_i de **C** et les arcs $i \to j$ sont étiquetés par des flèches $f : D_i \to D_j$ de **C**.

Considérer un diagramme **D** dans une catégorie **C** consiste en fait à "isoler" certains objets de **C**, et certaines flèches entre ces objets, parce qu'ils possèdent des propriétés particulières.

Un diagramme **D** est toujours défini à partir d'un ensemble I d'index, mais cet ensemble apparait souvent de manière implicite.

Image d'un diagramme par un foncteur

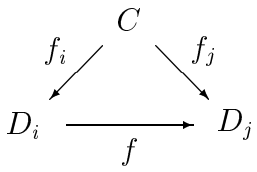
Soit deux catégories **C** et **C'**.
Soit F :**C**$\to$**C'** un foncteur.
Soit un diagramme **D** dans **C**.
L'image de **D** par F est un diagramme **D'** dans **C'**, construit sur le même graphe que **D**.
Les nœuds i sont étiquetés par les objets $F(D_i)$ et les arcs $i \to j$ sont étiquetés par les flèches $F(f) : F(D_i) \to F(D_j)$.

Exemple:
Soit **D** le diagramme: $D_0 \xrightarrow{a} D_1 \xrightarrow{b} D_2$

L'image de **D** par F est le diagramme $F(\mathbf{D})$:
$F(D_0) \xrightarrow{F(a)} F(D_1) \xrightarrow{F(b)} F(D_2)$

Définition: cône

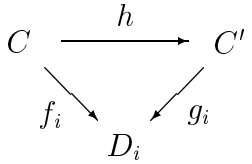
Soit un diagramme **D** dans une catégorie **C**. Un cône de **D** est un objet C de **C** et une famille de flèches $\{f_i : C \to D_i,\ i \in I\}$ tel que:
Pour toute flèche $f : D_i \to D_j$ du diagramme **D**, on a: $f \circ f_i = f_j$.



Remarque:
Soit deux catégories **C** et **C'**, et un foncteur F :**C**→**C'**. L'image par F d'un cône sur le diagramme **D** est un cône sur le diagramme $F(\mathbf{D})$.
Il faut montrer que $(F(C), \{F(f_i : F(C) \to F(D_i),\ i \in I\})$ est un cône, c'est-à-dire que pour toute flèche $F(f) : F(D_i) \to F(D_j)$ de $F(\mathbf{D})$, on a:
$F(f) \circ F(f_i) = F(f_j)$.
$f \circ f_i = f_j$, donc $F(f \circ f_i) = F(f_j)$, et donc, comme F est un foncteur: $F(f) \circ F(f_i) = F(f_j)$.

Catégorie des cônes sur un diagramme

On définit la catégorie **Cône(C,D)** des cônes sur **D**:
Les objets sont les cônes sur **D**.
Soit deux cônes:
$Z = (C, \{f_i : C \to D_i,\ i \in I\})$ et $Z' = (C', \{g_i : C' \to D_i,\ i \in I\})$.
Une flèche de Z dans Z' est une flèche $h : C \to C'$ de **C** telle que:
$\forall i \in I : g_i \circ h = f_i$

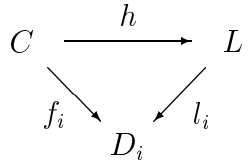


Cône(C,D) est une catégorie.

Définition: limite d'un diagramme

Une limite du diagramme **D** dans une catégorie **C** est un objet terminal de la catégorie **Cône(C,D)**.

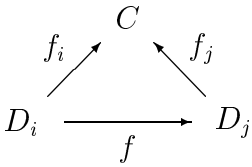
Autrement dit:
Une limite du diagramme **D** est un cône $(L, \{l_i : L \to D_i, i \in I\})$ tel que:
Pour tout cône $(C, \{f_i : C \to D_i,\ i \in I\})$, il existe une unique flèche $h : C \to L$ telle que:
$\forall i \in I : l_i \circ h = f_i$.



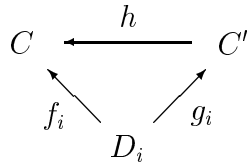
On définit de façon duale un cocône et la colimite d'un diagramme.

Cocône

Soit un diagramme **D** dans une catégorie **C**. Un cocône de **D** est un objet C de **C** et une famille de flèches $\{f_i : D_i \to C,\ i \in I\}$ tel que:
Pour toute flèche $f : D_i \to D_j$ du diagramme **D**, on a: $f_j \circ f = f_i$.



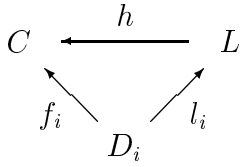
La classe des cocônes sur un diagramme **Cocône(C,D)** est une catégorie:
Les objets de **Cocône(C,D)** sont les cocônes sur **D**.
Soit deux cocônes:
$Z = (C, \{f_i : D_i \to C,\ i \in I\})$ et $Z' = (C', \{g_i : D_i \to C',\ i \in I\})$.
Une flèche de Z dans Z' est une flèche $h : C \to C'$ de **C** telle que:
$\forall i \in I : h \circ f_i = g_i$



Colimite d'un diagramme

Une colimite d'un diagramme **D** est un objet initial de la catégorie **Cocône(C,D)**.

Le cocône $(L, \{l_i : D_i \to L,\ i \in I\})$ est une colimite de **D** si:
Pour tout cocône $(C, \{f_i : D_i \to C,\ i \in I\})$, il existe un unique $h : L \to C$ tel que $\forall i \in I : h \circ l_i = f_i$.



Remarque:

La construction d'un produit est en réalité un cas particulier de limite: le produit de deux objets A et B est le cône limite du diagramme (A, B), c'est-à-dire du diagramme composé des objets A et B et ne comportant aucune flèche.

De même, la somme de deux objets A et B est le cocône colimite du diagramme (A, B).

ω-diagramme

Un ω-diagramme est un diagramme **D**, dont les objets sont indexés par les entiers naturels, et tel qu'il y a une flèche unique f_i entre les index i et $i+1$.

$$D_0 \xrightarrow{f_0} D_1 \xrightarrow{f_1} D_2 \xrightarrow{f_2} D_3 \cdots$$

La colimite d'un ω-diagramme est appelée ω-colimite.

Si dans une catégorie, tout ω-diagramme a une colimite, on dit que *la catégorie a des ω-colimites.*

Interprétation des ω-colimites dans un CPO:

On considère un CPO **E**. **E** peut être considéré comme une catégorie **C** de la façon suivante: Les objets de **C** sont les éléments de **E**. Il existe une unique flèche entre les objets A et B de **C** si et seulement si: $A \sqsubseteq B$, où $\sqsubseteq$ est l'ordre partiel sur **E**.

Soit un ω-diagramme **D**: $D_0 \xrightarrow{f_0} D_1 \xrightarrow{f_1} D_2 \xrightarrow{f_2} D_3 \cdots$ sur **C**.

Cet ω-diagramme correspond à la chaine croissante $D_0 \sqsubseteq D_1 \sqsubseteq D_2 \sqsubseteq ...$

On a: $(A, \{a_i : D_i \to A;\ i \in I\!N\})$ est la colimite de **D** si et seulement si A est la

borne supérieure de la chaine $D_0 \sqsubseteq D_1 \sqsubseteq D_2 \sqsubseteq ...$

Par conséquent, la notion d'ω-colimite est une généralisation de la notion de borne supérieure dans un CPO sur des classes d'objets possédant une structure plus complexe qu'un ordre partiel.

2.2.5 Foncteur conservant des colimites

Définition:

Soit deux catégories **C** et **C'**.
Soit un diagramme **D** sur **C**.
On suppose que **D** a un cocône colimite $(L, \{l_i : D_i \to L,\ i \in I\})$. Soit F un foncteur de **C** dans **C'**.
On dit que F *conserve la colimite* de **D** si:
$(F(L), \{F(l_i) : F(D_i) \to F(L),\ i \in I\})$ est un cocône colimite du diagramme F(**D**) dans **C'**.

Interprétation dans un CPO:

Soit **E** un CPO, et **C** la catégorie associée à ce CPO. Soit F un foncteur conservant les colimites d'ω-diagrammes.

- F est croissant: soit A et B deux éléments de **E** tels que $A \sqsubseteq B$. $\exists! f : A \to B$, donc on a une flèche $F(f) : F(A) \to F(B)$, et donc $F(A) \sqsubseteq F(B)$.

- F est continu:
  Soit un ω-diagramme **D**: $D_0 \xrightarrow{f_0} D_1 \xrightarrow{f_1} D_2 \xrightarrow{f_2} D_3 \ \ldots$ sur **C**.
  Soit $(L, \{l_i : D_i \to L,\ i \in I\!N\})$ la colimite de ce diagramme.
  Dans **E**, cet ω-diagramme correspond à la chaine croissante:
  $D_0 \sqsubseteq D_1 \sqsubseteq D_2 \sqsubseteq ...$, et on a: $L = \bigsqcup_{i=0}^{\infty} D_i$.

  F conserve les ω-colimites donc $(F(L), \{F(l_i) : F(D_i) \to F(L),\ i \in I\!N\})$ est un cocône colimite de $F(\mathbf{D})$.

  Donc $F(L) = \bigsqcup_{i=0}^{\infty} F(D_i)$,
  donc: $F(\bigsqcup_{i=0}^{\infty} D_i) = \bigsqcup_{i=0}^{\infty} F(D_i)$, et donc F est continu.

2.3 Catégorie des F-algèbres

Soit un endofoncteur F dans une catégorie **C**.

Définition: F-algèbre

Une F-algèbre est un couple (A, a), où A est un objet de $\mathbf{C}$ et $a : F(A) \to A$ est une flèche de $\mathbf{C}$.

Homomorphisme de F-algèbre

Soit deux F-algèbres (A, a) et (B, b). $h : A \to B$ est un homomorphisme de (A, a) dans (B, b) si: $h \circ a = b \circ F(h)$.

$$\begin{array}{ccc} & F(A) & \xrightarrow{a} & A & \\ F(h) & \downarrow & & \downarrow & h \\ & F(B) & \xrightarrow[b]{} & B & \end{array}$$

Catégorie des F-algèbres

On obtient la catégorie $(F : \mathbf{C})$ des F-algèbres, dont les objets sont les F-algèbres et les flèches les homomorphismes de F-algèbre. En effet:

- Pour chaque F-algèbre (A, a), id_A est un homomorphisme de (A, a) dans (A, a) car $id_A \circ a = a \circ F(id_A)$
  De plus id_A vérifie les axiomes de l'identité pour les homomorphismes, puisque id_A les vérifie pour les flèches de $\mathbf{C}$.

- Soit trois F-algèbres (A, a), (B, b) et (C, c). Soit h un homomorphisme de (A, a) dans (B, b) et k un homomorphisme de (B, b) dans (C, c). Montrons que $k \circ h$ est un homomorphisme de (A, a) dans (C, c):

  $$\begin{array}{lll} (k \circ h) \circ a & = k \circ (h \circ a) & \text{(associativité de la composition)} \\ & = k \circ (b \circ F(h)) & (h \text{ homomorphisme}) \\ & = (k \circ b) \circ F(h) & \text{(associativité de la composition)} \\ & = (c \circ F(k)) \circ F(h) & (k \text{ homomorphisme}) \\ & = c \circ (F(k) \circ F(h)) & \text{(associativité de la composition)} \\ & = c \circ F(k \circ h) & (F \text{ est un foncteur}) \end{array}$$

  L'associativité de la composition des homomorphismes provient de l'associativité de la composition des flèches dans $\mathbf{C}$.

Exemple

On se place dans la catégorie des ensembles **SET**.
On considère le foncteur F défini par:

- Pout tout ensemble A, $F(A) = 1 + A$

- Pour toute application $f : A \to B, F(f) : 1 + A \to 1 + B = id_1 + f$

On vérifie facilement que F est un foncteur.

Soit $\mathbb{N}$, l'ensemble des entiers naturels, et *succ* l'opération successeur sur les naturels. Soit l'application c_1 définie par:

$$\begin{array}{rll} c_1 : & 1 + \mathbb{N} & \to \mathbb{N} \\ & (1, *) & \mapsto 0 \\ & (2, n) & \mapsto succ(n) \end{array}$$

$(\mathbb{N}, c_1)$ est une F-algèbre.

Soit $Z_2 = \{0, 1\}$, l'ensemble des entiers modulo 2. Soit $succ_2$, l'application successeur dans Z_2.
On définit l'application c_2 par:

$$\begin{array}{rll} c_2 : & 1 + Z_2 & \to Z_2 \\ & (1, *) & \mapsto 0 \\ & (2, n) & \mapsto succ_2(n) \end{array}$$

(Z_2, c_2) est une F-algèbre.

Il existe un unique homomorphisme d'algèbre entre $(\mathbb{N}, c_1)$ et (Z_2, c_2).

$$\begin{array}{rll} h : & \mathbb{N} & \to Z_2 \\ & n & \mapsto n \ mod \ 2 \end{array}$$

On vérifie facilement que h est un homomorphisme, c'est-à-dire que:
$h \circ c_1 = c_2 \circ F(h)$.

2.4 Construction de points fixes

Dans cette section, on se place dans une catégorie possédant un objet initial. On considère un endofoncteur F qui conserve les colimites. On montre qu'on peut construire un point fixe pour F, c'est-à-dire un objet X tel que X et $F(X)$ sont isomorphes. Cette construction est due à Plotkin et Smyth ([$SP77$]), et peut être considérée comme une généralisation du théorème du point fixe de Tarski pour les fonctions continues dans un CPO.

On commence par montrer trois lemmes élémentaires, utiles pour démontrer le théorème de Plotkin et Smyth.

Lemme 1:

Soit un diagramme **D** sur **C**. Soit $Z = (L, \{l_i : D_i \to L, \ i \in I\})$, un cocône colimite de **D**.
Soit $a : L \to M$, et $b : L \to M$.
Si $\forall i \in I : a \circ l_i = b \circ l_i$ alors on a: $a = b$.

Preuve:

On considère $(M, \{b \circ l_i : D_i \to M,\ i \in I\})$.
C'est un cocône car:
Pour toute flèche $f : D_i \to D_j$, on a: $l_j \circ f = l_i$, (Z est un cocône colimite)
donc $b \circ l_j \circ f = b \circ l_i$.

Par conséquent, il existe un unique $h : L \to M$ tel que: $h \circ l_i = b \circ l_i$.
a et b satisfont tous les deux cette propriété, donc $a = b$.

Lemme 2:

Soit un diagramme **D** dans **C**. Soit $\mathbb{L} = (L, \{\lambda_i : D_i \to L,\ i \in I\}$ et $\mathbb{M} = (M, \{\mu_i : D_i \to M,\ i \in I\}$ deux cocônes colimites sur **D**.
Alors il existe $\phi : L \to M$ et $\psi : M \to L$ tels que:

$$\begin{array}{rcl} \phi \circ \lambda_i & = & \mu_i \\ \psi \circ \mu_i & = & \lambda_i \\ \phi \circ \psi & = & id_M \\ \psi \circ \phi & = & id_L \end{array}$$

Preuve:

$\mathbb{L}$ est un cocône colimite de **D**, $\mathbb{M}$ est un cocône sur **D**, donc il existe un unique $\phi : L \to M$ tel que: $\forall i \in I, \phi \circ \lambda_i = \mu_i$
$\mathbb{M}$ est un cocône colimite de **D**, $\mathbb{L}$ est un cocône sur **D**, donc il existe un unique $\psi : M \to L$ tel que: $\forall i \in I, \psi \circ \mu_i = \lambda_i$
De plus: $\forall i \in I : \phi \circ \psi \circ \mu_i = \phi \circ \lambda_i = \mu_i$. Donc, d'apres le lemme 1, $\phi \circ \psi = id_M$ De même: $\forall i \in I : \psi \circ \phi \circ \lambda_i = \psi \circ \mu_i = \lambda_i$. Donc, d'après le lemme 1, $\psi \circ \phi = id_L$.

Lemme 3:

Soit un ω-diagramme **D**:
$L_0 \xrightarrow{\theta_0} L_1 \xrightarrow{\theta_1} L_2 \xrightarrow{\theta_2} L_3 \cdots$

On suppose que **D** a un cocône colimite: $(L, \{\xi_i : L_i \to L,\ i \in \mathbb{N}\})$.

Soit le ω-diagramme **D'**, obtenu à partir de **D** en supprimant L_0:
$L_1 \xrightarrow{\theta_1} L_2 \xrightarrow{\theta_2} L_3 \xrightarrow{\theta_3} L_4 \cdots$
Alors **D'** a un cocône colimite qui est: $(L, \{\xi_i : L_i \to L,\ i > 0\})$.

Preuve:

D'abord, $(L, \{\xi_i : L_i \to L,\ i > 0\})$ est bien un cocône sur **D'**.
Il faut montrer que pour tout cocône $(M, \{\mu_i : L_i \to M,\ i > 0\})$, il existe un unique $\phi : L \to M$ tel que: $\forall i > 0 : \phi \circ \xi_i = \mu_i$.
On pose: $\mu_0 = \mu_1 \circ \theta_0$.
Existence: $(M, \{\mu_i : L_i \to M,\ i \in I\!N\})$ est un cocône sur **D**, donc il existe un unique $\phi : L \to M$ tel que: $\forall i \in I\!N : \phi \circ \xi_i = \mu_i$.
Ce ϕ satisfait donc: $\forall i > 0 : \phi \circ \xi_i = \mu_i$.
Unicité: Soit $\phi' : L \to M$ tel que $\forall i > 0 : \phi' \circ \xi_i = \mu_i$.
$\phi' \circ \xi_1 = \mu_1$, donc $\phi' \circ \xi_1 \circ \theta_0 = \mu_1 \circ \theta_0$, et donc $\phi' \circ \xi_0 = \mu_0$.
Par conséquent, d'après le lemme 1, $\phi = \phi'$.

2.4.1 Théorème (Plotkin, Smyth)

On se place dans une catégorie **C** ayant un objet initial L_0. Soit F un endofoncteur sur **C**. On construit un ω-diagramme de la façon suivante:
On pose: $\forall i > 0 : L_{i+1} = F(L_i)$
Soit θ_0 l'unique flèche $\theta_0 : L_0 \to L_1$.
On pose: $\forall i > 0 : \theta_{i+1} = F(\theta_i)$.
On obtient ainsi l'ω-diagramme:
$\mathbf{L} = \quad L_0 \xrightarrow{\theta_0} L_1 \xrightarrow{\theta_1} L_2 \xrightarrow{\theta_2} L_3 \quad \cdots$

Théorème:

On suppose que le diagramme **L** a un cocône colimite: $Z = (L, \{\xi_i : L_i \to L,\ i \in I\!N\})$.
On suppose que F conserve cette colimite.
Alors il existe un isomorphisme $\psi : F(L) \to L$.
De plus, l'algèbre (L, ψ) est initiale dans la catégorie $(F : \mathbf{C})$ des F-algèbres.

Preuve:

Le foncteur F conserve la colimite de **L**, donc
$Z' = (F(L), \{F(\xi_i) : F(L_i) \to F(L),\ i \in I\!N\})$ est un cocône colimite du diagramme:

$$F(L_0) \xrightarrow{F(\theta_0)} F(L_1) \xrightarrow{F(\theta_1)} F(L_2) \xrightarrow{F(\theta_2)} F(L_3) \quad \cdots$$

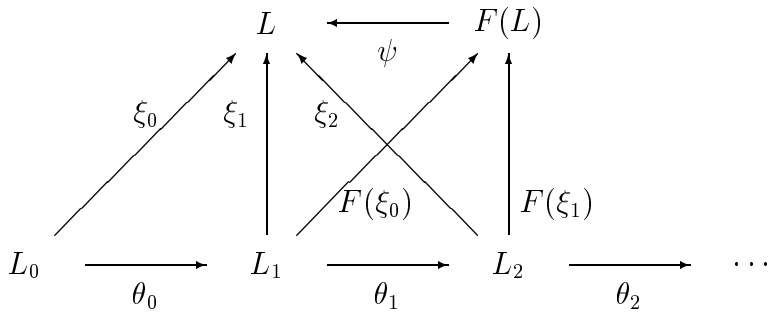
Ce diagramme peut se réécrire en:

$$L_1 \xrightarrow{\theta_1} L_2 \xrightarrow{\theta_2} L_3 \xrightarrow{\theta_3} L_4 \quad \cdots$$

D'après le lemme 3, ce diagramme a pour colimite: $Z = (L, \{\xi_i : L_i \to L,\ i > 0\})$.

D'après le lemme 2, il existe $\psi : F(L) \to L$, isomorphisme, tel que:

$$\forall i \in I\!N : \psi \circ F(\xi_i) = \xi_{i+1}$$



Montrons que (L, ψ) est initial dans la catégorie $(F : \mathbf{C})$ des F-algèbres, c'est-à-dire: $\forall \eta : F(M) \to M, \exists! \sigma : L \to M$ tel que: $\sigma \circ \psi = \eta \circ F(\sigma)$.

$$\begin{array}{ccccc} & F(L) & \xrightarrow{\psi} & L & \\ F(\sigma) & \downarrow & & \downarrow & \sigma \\ & F(M) & \xrightarrow[\eta]{} & M & \end{array}$$

Existence:
Soit ν_0, l'unique flèche $\nu_0 : L_0 \to M$. Soit $\nu_{i+1} = \eta \circ F(\nu_i)$. On montre par induction que: $\nu_{i+1} \circ \theta_i = \nu_i$.

- $\nu_1 \circ \theta_0 = \nu_0$, car L_0 est initial.
- Supposons que $\nu_i \circ \theta_{i-1} = \nu_{i-1}$.

$$\begin{array}{lll} \nu_{i+1} \circ \theta_i & = \eta \circ F(\nu_i) \circ \theta_i & \text{(définition de } \nu_{i+1}) \\ & = \eta \circ F(\nu_i) \circ F(\theta_{i-1}) & \text{(définition de } \theta_i) \\ & = \eta \circ F(\nu_i \circ \theta_{i-1}) & (F \text{ est un foncteur}) \\ & = \eta \circ F(\nu_{i-1}) & \text{(Hypothèse d'induction)} \\ & = \nu_i & \text{(définition de } \nu_i). \end{array}$$

Donc $(M, \{\nu_i : L_i \to M,\ i \in I\!N\})$ est un cocône sur le diagramme:

$$L_0 \xrightarrow{\theta_0} L_1 \xrightarrow{\theta_1} L_2 \xrightarrow{\theta_2} L_3 \quad \cdots$$

Par conséquent, comme Z est un cocône colimite sur ce diagramme, il existe un unique $\sigma : L \to M$ tel que: $\forall i \in I\!N : \sigma \circ \xi_i = \nu_i$.

D'autre part, $Z = (F(L), \{F(\xi_i) : F(L_i) \to F(L),\ i \in I\!N\})$ est un cocône colimite, donc d'après le lemme 1:

$\forall i \in I\!N, \sigma \circ \psi \circ F(\xi_i) = \eta \circ F(\sigma) \circ F(\xi_i) \Rightarrow \sigma \circ \psi = \eta \circ F(\sigma)$.

Il suffit donc de montrer que $\forall i \in I\!N, \sigma \circ \psi \circ F(\xi_i) = \eta \circ F(\sigma) \circ F(\xi_i)$

$\sigma \circ \psi \circ F(\xi_i)$	$= \sigma \circ \xi_{i+1}$	(propriété de ψ)
	$= \nu_{i+1}$	(définition de σ)
	$= \eta \circ F(\nu_i)$	(définition de ν_{i+1})
	$= \eta \circ F(\sigma \circ \xi_i)$	(définition de σ)
	$= \eta \circ F(\sigma) \circ F(\xi_i)$	(F est un foncteur)

Unicité:
On montre que: $\sigma \circ \psi = \eta \circ F(\sigma) \Rightarrow \forall i \in I\!N : \sigma \circ \xi_i = \nu_i$.
Comme il existe un unique σ possédant cette propriété, σ est unique.
Par induction sur i:

- L_0 est initial, donc il existe une flèche unique $L_0 \to M$, donc $\sigma \circ \xi_0 = \nu_0$.
- On suppose $\sigma \circ \xi_i = \nu_i$.

$\sigma \circ \xi_{i+1}$	$= \sigma \circ \psi \circ F(\xi_i)$	(propriété de ψ)
	$= \eta \circ F(\sigma) \circ F(\xi_i)$	(hypothèse)
	$= \eta \circ F(\sigma \circ \xi_i)$	(F est un foncteur)
	$= \eta \circ F(\nu_i)$	(hypothèse d'induction)
	$= \nu_{i+1}$	(définition de ν_{i+1})

2.4.2 Rapport avec le théorème du point fixe de Tarski

Soit **E** un CPO. On considère E comme une catégorie. L'objet initial de **E** est le plus petit élément $\perp$. Soit F un foncteur conservant les ω-colimites: C'est une application continue de **E** dans **E**.

Dans un CPO, tout ω-diagramme a une colimite (puisque toute chaine croissante a une borne supérieure).

On pose $L = \bigsqcup_{i=0}^{\infty} F^i(\perp)$.

Le théorème de Plotkin et Smyth dit:

- $L = F(L)$, c'est-à-dire que L est un point fixe de F.
- Si M est tel que $F(M) \sqsubseteq M$, alors $L \sqsubseteq M$. (La propriété d'initialité dans la catégorie des F-algèbres correspond à la propriété de *plus petit* point fixe.)

Chapitre 3

Application dans les catégories CPO et SET

3.1 Introduction

On a présenté dans le chapitre précédent le théorème de Plotkin et Smyth. Dans ce chapitre, on applique ce théorème dans deux catégories: la catégorie des ordres partiels complets **CPO** et la catégorie des ensembles **SET**. Il s'agit de vérifier que les hypothèses du théorème sont satisfaites. Pour cela, Wand a introduit le concept d'O-catégorie ([$Wan79$]), c'est-à-dire de catégories dont les *ensembles de flèches entre deux objets* sont des ordres partiels complets. Plutôt que de présenter tout ce formalisme, nous appliquons ces idées dans la catégorie **CPO**. Ensuite nous nous plaçons dans la catégorie **SET**, où les résultats démontrés dans **CPO** s'appliquent assez facilement. Dans **SET**, lorsqu'on considère des foncteurs polynomiaux, on retrouve la définition des types abstraits algébriques.

3.2 Types abstraits dans CPO

Pour appliquer le théorème de Plotkin et Smyth, deux problèmes se posent: il faut examiner dans quelles conditions un ω-diagramme a une colimite. Scott a montré que si toutes les flèches du ω-diagramme sont des *plongements*, le diagramme a une colimite. Le second problème est l'absence d'élément initial dans **CPO**. La solution est d'appliquer la construction de point fixe dans la catégorie **CPOS** des CPO avec les fonctions continues strictes. Dans cette catégorie le CPO à un élément $\{\bot\}$ est initial.

3.2.1 Quelques rappels sur les CPO

Définition: ordre partiel complet (CPO)

Un ensemble A muni d'une relation d'ordre partielle $\sqsubseteq$ est un CPO si:

- Il existe un plus petit élément, noté $\perp_A$ (ou plus simplement: $\perp$).
  $\forall a \in A : \perp_A \sqsubseteq a$.
  $\perp$ représente la quantité minimale d'information qu'une fonction peut renvoyer; cette quantité d'information est minimale lorsque la fonction ne termine pas.

- Toute chaine croissante $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq ...$ d'éléments de A a une borne supérieure, notée $\bigsqcup_{i=0}^{\infty} x_i$.
  $\bigsqcup_{i=0}^{\infty} x_i$ est caractérisée par:

  - $\forall i \in I\!N : \; x_i \sqsubseteq \bigsqcup_{i=0}^{\infty} x_i$.
  - Soit $d \in A$.
    Si $\forall i \in I\!N, \; x_i \sqsubseteq d$, alors $\bigsqcup_{i=0}^{\infty} x_i \sqsubseteq d$.

  La chaine croissante est la quantité croissante d'information que représente les différentes étapes du calcul. Le résultat du calcul est la borne supérieure de cette quantité d'information.

Définition: fonction continue

Les fonctions calculables sont représentées par des fonctions *continues*:
$f : A \to B$ est continue si pour toute chaine croissante $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq ...$ d'éléments de A, on a: $f(\bigsqcup_{i=0}^{\infty} x_i) = \bigsqcup_{i=0}^{\infty} f(x_i)$.
Cette définition modélise le fait que plus l'argument d'une fonction contient d'information, plus la fonction renvoie d'information. De plus, une fonction ne peut pas renvoyer plus d'information que n'en contiennent toutes les approximations de son argument.

Remarque: Toute fonction continue est croissante, c'est-à-dire: $\forall a, a' \in A : \; a \sqsubseteq a' \Rightarrow f(a) \sqsubseteq f(a')$.

Définition: fonction stricte

La fonction f est stricte si chaque fois que son argument ne termine pas, l'appel de fonction ne termine pas:
$f : A \to B$ est *stricte* si: $f(\perp_A) = \perp_B$.

L'ensemble des fonctions continues entre deux CPO est un CPO:

Soit deux CPO A et B. On définit un ordre partiel sur les fonctions continues de A dans B par:

$f \sqsubseteq g$ si et seulement si $\forall a \in A : f(a) \sqsubseteq f(b)$.

On obtient un CPO, noté $[A \to B]$, dont le plus petit élément est la fonction constante égale à $\perp$.
Soit $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq ...$ une chaine croissante de fonctions continues de A dans B.
La fonction $\bigsqcup_{i=0}^{\infty} f_i$, définie par: $(\bigsqcup_{i=0}^{\infty} f_i)(a) = \bigsqcup_{i=0}^{\infty}(f_i(a))$ est la borne supérieure de

$$f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \cdots$$

Catégories CPO et CPOS

La composition de deux fonctions continues est continue. L'identité est continue. On a donc une catégorie **CPO**, dont les objets sont les CPO, et les flèches les fonctions continues.
De même, la composition de deux fonctions continues strictes est continue stricte. L'identité est continue stricte. On a donc une catégorie **CPOS**, dont les objets sont les CPO, et les flèches les fonctions continues *strictes*.

3.2.2 Quelques bifoncteurs dans CPO et CPOS

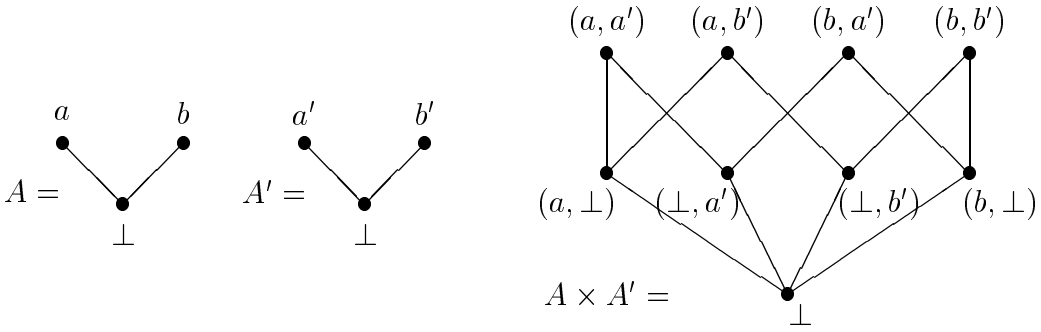
Dans ce paragraphe, nous présentons quelques constructions classiques de CPO: produit, somme ... Les notations sont empruntées à Scott ([$SG90$]). Par rapport à la présentation de Scott, nous insistons un peu plus sur l'aspect fonctoriel de ces constructions: à chaque construction de CPO correspond une construction de fonction continue, les deux constructions définissant un foncteur dans **CPO** ou dans **CPOS**.

Produit cartésien

Le produit cartésien de deux CPO A et A' est défini par:
$A \times A' = \{(a, a'), a \in A, a' \in A'\}$.
On a une relation d'ordre partielle sur $A \times A'$:
$(a, a') \sqsubseteq (b, b')$ si et seulement si $a \sqsubseteq b$ et $a' \sqsubseteq b'$.

$A \times A'$, muni de cette relation d'ordre, est un CPO.
$(\perp_A, \perp_{A'})$ est le plus petit élément.
$\bigsqcup_{i=0}^{\infty}(a_i, a'_i) = (\bigsqcup_{i=0}^{\infty} a_i, \bigsqcup_{i=0}^{\infty} a'_i)$.

Exemple:



Soit $f : C \to A$, et $f' : C \to A'$, deux fonctions continues.
On définit $< f, f' >: C \to A \times A'$ par:
$\forall c \in C : < f, f' > (c) = (f(c), f'(c))$.
$< f, f' >$ est une fonction continue.

On définit les projections p et p' de la manière habituelle:

$$\begin{array}{llll} p: & A \times A' & \to A & \\ & (a, a') & \mapsto a & \end{array} \qquad \begin{array}{llll} p': & A \times A' & \to A' & \\ & (a, a') & \mapsto a' & \end{array}$$

$\times$ et ces deux projections définissent un *produit catégoriel* (cf 2.2.3), c'est-à-dire que $< f, f' >: C \to A \times A'$ est l'unique fonction continue telle que: $p \circ < f, f' >= f$, et $p' \circ < f, f' >= f'$.

Produit de deux fonctions:
Etant donné deux fonctions $f : A \to B$, et $f' : A' \to B'$, le produit de f et f' est:
$f \times f' =< f \circ p, f' \circ p' >$

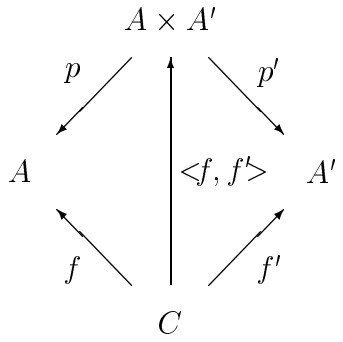
On peut vérifier que ce produit correspond au produit cartésien de f et f':

$$\begin{array}{llll} f \times f': & A \times A' & \to B \times B' & \\ & (a, b) & \mapsto (f(a), f(b)) & \end{array}$$

$\times$ est un foncteur dans CPO:
$(g \times g') \circ (f \times f') = (g \circ f) \times (g' \circ f')$
$id_{A \times A'} = id_A \times id_{A'}$

$\times$ est également le produit catégoriel dans **CPOS**.
En effet, les projections p et p' sont strictes.
Si f et f' sont strictes, alors $< f, f' >$ est stricte.

Considérons le diagramme dans la catégorie **CPO**:



Comme toutes les flèches sont strictes, le même diagramme peut être considéré comme un diagramme de **CPOS**.
On a les égalités:
$p \circ < f, f' >= f$, et $p' \circ < f, f' >= f'$.
On montre que $< f, f' >$ est l'unique flèche de **CPOS** qui satisfait ces deux égalités: si une autre flèche ϕ de **CPOS** satisfait les égalités, alors ϕ, considéré comme une flèche de **CPO** les satisfait aussi, et donc: $\phi =< f, f' >$.

Par conséquent, la catégorie **CPOS** a un produit, qui correspond au produit dans **CPO**.

Somme disjointe liftée

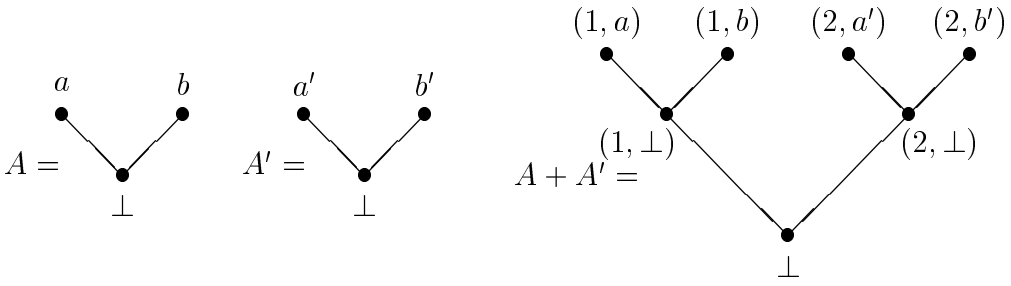
La somme disjointe liftée de deux CPO A et A' est définie par:
$A + A' = \{\perp_{A+A'}\} \cup \{(1, a), a \in A\} \cup \{(2, a'), a' \in A'\}$

Relation d'ordre sur $A + A'$:
Soit $x, y \in A + A'$:
$x \sqsubseteq y$ si et seulement si:

$x = \perp$	ou
$x = (1, a), y = (1, b)$, et $a \sqsubseteq b$	ou
$x = (2, a'), y = (2, b')$, et $a' \sqsubseteq b'$	

$A + A'$ est un CPO.

Exemple:



On définit les injections:

$$\begin{array}{rlll} i: & A & \to A+A' & \\ & a & \mapsto (1,a) & \end{array} \qquad \begin{array}{rlll} i': & A' & \to A+A' & \\ & a' & \mapsto (2,a) & \end{array}$$

Soit $f : A \to C$, et $f' : A' \to C$, deux fonctions continues. On définit:

$$\begin{array}{rlll} f\nabla f': & A+A' & \to & C \\ & \bot_{A+A'} & \mapsto & \bot_C \\ & (1,a) & \mapsto & f(a) \\ & (2,a') & \mapsto & f'(a') \end{array}$$

On a les égalités:

$$\begin{array}{rcl} (f\nabla f')(\bot) & = & \bot \\ (f\nabla f')\circ i & = & f \\ (f\nabla f')\circ i' & = & f' \end{array}$$

Remarque:
On n'a pas défini ainsi une somme catégorielle car il peut exister plusieurs fonctions ψ telles que $\psi \circ i = f$, et $\psi \circ i' = f'$. (+ est une somme catégorielle *faible*.)
Par contre, il existe une unique fonction ϕ **stricte** telle que: $\psi \circ i = f$, et $\psi \circ i' = f'$
Il ne faut pas conclure que + est la somme catégorielle dans **CPOS**, car i et i' ne sont pas strictes. On verra par la suite que **CPOS** a une somme catégorielle, qui est la *somme fusionnée* $\oplus$.

Somme disjointe liftée de deux fonctions:
Soit $f : A \to B$, et $f' : A' \to B'$. Soit $f + f' : A + A' \to B + B'$ définie par: $f + f' = (i \circ f)\nabla(i' \circ f')$. On vérifie que cette définition correspond à la définition suivante:

$$\begin{array}{rlll} f+f': & A+A' & \to & B+B' \\ & \bot_{A+A'} & \mapsto & \bot_{B+B'} \\ & (1,a) & \mapsto & (1,f(a)) \\ & (2,a') & \mapsto & (2,f'(a')) \end{array}$$

+ est un foncteur:
$(g + g') \circ (f + f') = (g \circ f) + (g' \circ f')$
$id_{A+A'} = id_A + id_{A'}$

Produit fusionné

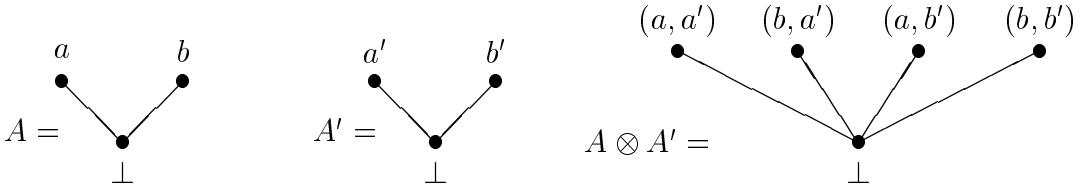
Dans les langages de programmation stricts, c'est-à-dire pour lesquels toute fonction définie est stricte, lorsque l'un des composants d'un couple d'arguments ne termine pas, le couple ne termine pas. Autrement dit, on ne veut pas distinguer les éléments $(a, \bot)$ et $\bot$. Ceci motive la définition du produit fusionné, (*smash product* en anglais), qui fusionne tous les couples d'éléments qui ont un composant égal à $\bot$.

Le produit fusionné de deux CPO A et A' est défini par:

$A \otimes A' = \{\perp_{A\otimes A'}\} \cup \{(a, a'), a \in A - \{\perp_A\}, a' \in A' - \{\perp_{A'}\}\}$

Relation d'ordre sur $A \otimes A'$:
$\forall x \in A \otimes A', \perp \sqsubseteq x$.
Soit $(a, a'), (b, b') \in A \otimes A'$:
$(a, a') \sqsubseteq (b, b')$ si et seulement si $a \sqsubseteq b$ et $a' \sqsubseteq b'$.

Exemple:



Produit fusionné de deux fonctions strictes:
Soit $f : A \to B$, et $f' : A' \to B'$, deux fonctions continues strictes. On définit:

$$
\begin{array}{lllll}
f \otimes f' : & A \times A' & \to & B \times B' & \\
& \perp_{A\otimes A'} & \mapsto & \perp_{B\otimes B'} & \\
& (a, a') & \mapsto & (f(a), f'(a')) & \text{si } f(a) \neq \perp_A \text{ et } f'(a') \neq \perp_{A'} \\
& & \mapsto & \perp_{B\otimes B'} & \text{sinon.}
\end{array}
$$

$f \otimes f'$ est continue stricte.

$\otimes$ est un foncteur dans la catégorie **CPOS**:
$(g \otimes g') \circ (f \otimes f') = (g \circ f) \otimes (g' \circ f')$
$id_{A\otimes A'} = id_A \otimes id_{A'}$

Remarque: On peut définir le produit fusionné de deux fonctions non strictes de la même façon. Le problème est que $\otimes$ n'est pas un foncteur dans la catégorie **CPO**.
En effet, prenons pour f la fonction constante égale à $\perp$, pour g la fonction constante égale à g_0, et l'identité pour f' et g'.
Soit $(a, a') \in A \times A'$ On a:
$(g \otimes g') \circ (f \otimes f')(a, a') = (g \otimes g')(\perp) = \perp$
$(g \circ f) \otimes (g' \circ f')(a, a') = (g \circ f(a), g' \circ f'(a')) = (g_0, a')$.

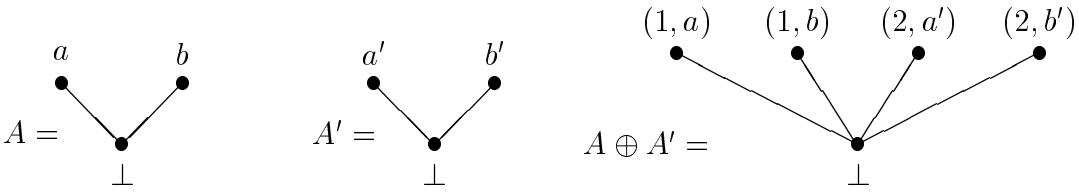
Somme fusionnée

Soit deux CPO A et A'. Dans certaines circonstances, on ne veut pas distinguer les éléments $(1, \perp_A)$, $(2, \perp_{A'})$, et $\perp$ dans $A + A'$. Ceci motive la définition de la somme fusionnée (*coalesced sum* en anglais), qui fusionne ces éléments:

La somme fusionnée de deux CPO A et A' est définie par:
$A \oplus A' = \{\perp_{A\oplus A'}\} \cup \{(1, a), a \in A - \{\perp_A\}\} \cup \{(2, a'), a' \in A' - \{\perp_{A'}\}\}$

Relation d'ordre sur $A \oplus A'$:
Soit $x, y \in A \oplus A'$:
$x \sqsubseteq y$ si et seulement si:
$x = \perp$ ou
$x = (1, a), y = (1, b)$, et $a \sqsubseteq b$ ou
$x = (2, a'), y = (2, b')$, et $a' \sqsubseteq b'$.

Exemple:



On définit les injections:

$$
\begin{array}{lrcll}
i: & A & \to & A \oplus A' & \\
 & a & \mapsto & (1, a), & \text{si } a \neq \perp_A \\
 & & \mapsto & \perp, & \text{si } a = \perp_A \\
i': & A' & \to & A \oplus A' & \\
 & a' & \mapsto & (1, a'), & \text{si } a' \neq \perp_{A'} \\
 & & \mapsto & \perp, & \text{si } a' = \perp_{A'}
\end{array}
$$

$\otimes$, i, et i' définissent une somme catégorielle (cf 2.2.3) dans **CPOS**.
Soit $f : A \to C$, et $f' : A' \to C$, deux fonctions continues strictes. Il existe une unique fonction continue stricte $f \vee f' : A \oplus A' \to C$, telle que: $(f \vee f') \circ i = f$, et $(f \vee f') \circ i' = f'$
On vérifie que cette fonction $f \vee f'$ est définie par:

$$
\begin{array}{lrcl}
f \vee f': & A \oplus A' & \to & C \\
 & \perp_{A \oplus A'} & \mapsto & \perp_C \\
 & (1, a) & \mapsto & f(a) \\
 & (2, a') & \mapsto & f'(a')
\end{array}
$$

Somme fusionnée de deux fonctions strictes:
Soit $f : A \to B$, et $f' : A' \to B'$, deux fonctions continues strictes.
$f \oplus f' = (i \circ f) \vee (i' \circ f')$.

On peut vérifier que cette définition correspond à la définition suivante:

$$
\begin{array}{lrcll}
f \oplus f': & A \oplus A' & \to & B \oplus B' & \\
 & \perp_{A \oplus A'} & \mapsto & \perp_{B \oplus B'} & \\
 & (1, a) & \mapsto & (1, f(a)) & \text{si } f(a) \neq \perp_B \\
 & & \mapsto & \perp_{B \oplus B'} & \text{sinon} \\
 & (2, a') & \mapsto & (2, f'(a')) & \text{si } f'(a') \neq \perp_{B'} \\
 & & \mapsto & \perp_{B \oplus B'} & \text{sinon}
\end{array}
$$

$\oplus$ est un foncteur dans **CPOS**:
$(g \oplus g') \circ (f \oplus f') = (g \circ f) \oplus (g' \circ f')$
$id_{A \oplus A'} = id_A \oplus id_{A'}$

Remarque: Comme pour le produit fusionné, on peut définir la somme fusionnée de deux fonctions non strictes. Cette somme fusionnée définie dans **CPO** n'est pas un foncteur.

Lemme 1:

- Les fonctions (d'odre supérieur) $\times$ et $+$ sont continues dans **CPO**:
  Pour toute chaine croissante de fonctions continues:
  $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq ...$, et $f'_0 \sqsubseteq f'_1 \sqsubseteq f'_2 \sqsubseteq ...$, on a:

$$(\bigsqcup_{i=0}^{\infty} f_i) \times (\bigsqcup_{i=0}^{\infty} f'_i) = \bigsqcup_{i=0}^{\infty} (f_i \times f'_i)$$

$$(\bigsqcup_{i=0}^{\infty} f_i) + (\bigsqcup_{i=0}^{\infty} f'_i) = \bigsqcup_{i=0}^{\infty} (f_i + f'_i)$$

- Les fonctions $\otimes$ et $\oplus$ sont continues dans **CPOS**:
  Pour toute chaine croissante de fonctions continues *strictes*:
  $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq ...$, et $f'_0 \sqsubseteq f'_1 \sqsubseteq f'_2 \sqsubseteq ...$, on a:

$$(\bigsqcup_{i=0}^{\infty} f_i) \otimes (\bigsqcup_{i=0}^{\infty} f'_i) = \bigsqcup_{i=0}^{\infty} (f_i \otimes f'_i)$$

$$(\bigsqcup_{i=0}^{\infty} f_i) \oplus (\bigsqcup_{i=0}^{\infty} f'_i) = \bigsqcup_{i=0}^{\infty} (f_i \oplus f'_i)$$

3.2.3 Plongements

La notion de plongement (*embedding* en anglais) a été introduite par Scott, puis utilisée intensivement par Wand avec les O-catégories. L'idée est de construire une relation d'ordre entre les CPO, qui permette de modéliser que un CPO est "inclus" dans un autre. On veut par exemple modéliser que, étant donné deux CPO A et B, A est "inclus" dans $A + B$. Si on considère l'inclusion des ensembles, ce n'est pas le cas. Néanmoins, on "retrouve" tous les éléments de A "codés" dans $A + B$. La notion de plongement est en fait une extension pour les CPO de la notion d'injection pour les ensembles.

Définition: plongement

Soit deux CPO A et B.
Soit $f : A \to B$, une fonction continue.
f est un plongement s'il existe $g : B \to A$ continue telle que:
$g \circ f = id_A$

$f \circ g \sqsubseteq id_B$.

Une telle fonction g est alors unique:
Soit $g' : B \to A$ telle que:
$g' \circ f = id_A$
$f \circ g' \sqsubseteq id_B$.
On a:
$f \circ g \sqsubseteq id_B \Rightarrow g' \circ f \circ g \sqsubseteq g' \Rightarrow g \sqsubseteq g'$
$f \circ g' \sqsubseteq id_B \Rightarrow g \circ f \circ g' \sqsubseteq g \Rightarrow g' \sqsubseteq g$
d'où: $g = g'$.

Lorsque g existe, on note: $g = f^P$.

La composition de deux plongements f et f' est un plongement, et on a:
$(f \circ f')^P = f'^P \circ f^P$.
L'identité est un plongement, et $id_A^P = id_A$.
On a donc une catégorie **PLG**, dont les objets sont les CPO, et les flèches les plongements.

Remarque: un plongement est une fonction stricte.
Preuve:

	$\perp_A \sqsubseteq f^P(\perp_B)$	($\perp_A$ plus petit élément)
$\Rightarrow$	$f(\perp_A) \sqsubseteq f(f^P(\perp_B))$	(f continue $\Rightarrow$ f croissante)
$\Rightarrow$	$f(\perp_A) \sqsubseteq \perp_B$	($f \circ f^P \sqsubseteq id_B$)
$\Rightarrow$	$f(\perp_A) = \perp_B$	($\perp_B$ plus petit élément).

De même, on montre que $f^P(\perp_B) = \perp_A$.

La catégorie **PLG** des plongements est donc une sous catégorie de **CPOS**.

Lemme 2:

Soit $\dagger$ un bifoncteur croissant, c'est-à-dire tel que:

$$f \sqsubseteq f', g \sqsubseteq g' \;\Rightarrow\; f \dagger g \sqsubseteq f' \dagger g'$$

Soit $f : A \to B$, et $g : A' \to B'$ deux plongements.
Alors: $f \dagger g$ est un plongement, et $(f \dagger g)^P = f^P \dagger g^P$.

Preuve:

$$\begin{array}{lll}
(f^P \dagger g^P) \circ (f \dagger g) & = (f^P \circ f) \dagger (g^P \circ g) & (\dagger \text{ est un bifoncteur}) \\
& = id_A \dagger id_{A'} & (\text{définition du plongement}) \\
& = id_{A \dagger A'} & (\dagger \text{ est un bifoncteur}) \\
\\
(f \dagger g) \circ (f^P \dagger g^P) & = (f \circ f^P) \dagger (g \circ g^P) & (\dagger \text{ est un bifoncteur}) \\
& \sqsubseteq id_B \dagger id_{B'} & (f \circ f^P \sqsubseteq id_B,\ g \circ g^P \sqsubseteq id_{B'}) \\
& & (\dagger \text{ est croissant}) \\
& \sqsubseteq id_{B \dagger B'} & (\dagger \text{ est un bifoncteur})
\end{array}$$

Corollaires:

Soit $f : A \to B$, et $g : A' \to B'$ deux plongements.
$f \times g$, $f + g$, $f \otimes g$, $f \oplus g$ sont des plongements.

En effet, $\times$, $+$, $\otimes$, et $\oplus$ sont continus, donc croissants.

3.2.4 Construction d'ω-colimites

Dans ce paragraphe, on se place dans la catégorie **CPOS**, et on montre qu'un ω-diagramme dont les flèches sont des plongements a un cocône colimite.

Lemme 3:

On considère des CPO $L_0, L_1, L_2...$, tels que qu'on ait des plongements:
$f_i : L_i \to L_{i+1}$.
Alors le diagramme $L_0 \stackrel{f_0}{\to} L_1 \stackrel{f_1}{\to} L_2 \cdots$ a un cocône colimite dans **CPOS**.

Preuve:

Pour montrer ce lemme, il faut:

1. Construire un CPO L
2. Construire des flèches $\xi_i : L_i \to L$
3. Montrer que $\mathbb{L} = (L, \{\xi_i : L_i \to L,\ i \in \mathbb{N}\})$ est un cocône.
4. Montrer que $\mathbb{L}$ est un cocône *colimite* dans **CPOS**.

Cette construction est attribuée par Wand à Scott. Les démonstrations sont largement inspirées de celles données par Tennent ([*Ten*91]).

1. Si à la place des plongements f_i, on avait des inclusions d'ensemble, il suffirait de prendre pour L la réunion de tous les L_i. Il faut faire une construction qui garde la même idée: on considère comme élément de L un élément de L_k, mais en plus, on note sous quelle forme apparait cet élément dans tous les L_i pour $i > k$. On est donc conduit à considérer comme éléments de L des séquences infinies $(l_0, l_1, l_2, \cdots)$, avec $l_i \in L_i$:

   Soit L l'ensemble des séquences infinies $(l_0, l_1, l_2, \cdots)$ telles que:
   $\forall i \in I\!N :\ l_i \in L_i$
   $\quad\quad\quad\quad\ \ l_i = f_i^P(l_{i+1})$
   On a donc: $\forall i \in I\!N : f_i(l_i) = f_i(f_i^P(l_{i+1})) \sqsubseteq l_{i+1}$.
   L est un CPO:
   La séquence $(\bot_{L_0}, \bot_{L_1}, \bot_{L_2}, \cdots)$ est le plus petit élément.
   Soit une séquence croissante $s_0 \sqsubseteq s_1 \sqsubseteq s_2 \sqsubseteq \cdots$ d'éléments de L, avec $s_i = (l_0^i, l_1^i, l_2^i \cdots)$. La borne supérieure des s_i est:

$$\bigsqcup_{i=0}^{\infty} s_i = (\bigsqcup_{i=0}^{\infty} l_0^i,\ \bigsqcup_{i=0}^{\infty} l_1^i,\ \bigsqcup_{i=0}^{\infty} l_2^i,\ \cdots)$$

2. On construit maintenant des fonctions $\xi_i : L_i \to L$. Soit $l \in L_i$. $\xi_i(l)$ est une séquence $(x_0, x_1, x_2, \cdots)$ définie par:
   $x_j = (f_{j-1} \circ f_{j-2} \circ \cdots \circ f_{i+1} \circ f_i)(l)$ si $i < j$
   $x_j = l$ si $i = j$
   $x_j = (f_j^P \circ f_{j+1}^P \circ \cdots \circ f_{i-1}^P)(l)$ si $i > j$

   Les fonction ξ_i ainsi définies sont continues et strictes. De plus, ce sont des plongements, et $\xi_i^P(x_0, x_1, x_2, \cdots) = x_i$.

   En effet:

   - $(\xi_i^P \circ \xi_i)(l) = \xi_i^P(x_0, x_1, x_2, \cdots) = x_i = l$
     donc $\xi_i^P \circ \xi_i = id_{L_i}$.
   - Soit $(y_0, y_1, y_2, \cdots) = \xi_i \circ \xi_i^P(x_0, x_1, x_2, \cdots) = \xi_i(x_i)$.
     Il faut montrer que $\xi_i \circ \xi_i^P(x_0, x_1, x_2, \cdots) \sqsubseteq (x_0, x_1, x_2, \cdots)$, c'est-à-dire que:
     $\forall j \in I\!N : y_j \sqsubseteq x_j$.
     Si $i < j$: $y_j = (f_{j-1} \circ f_{j-2} \circ \cdots \circ f_{i+1} \circ f_i)(x_i) \sqsubseteq x_j$,
     car $\forall k, f_k(x_k) = x_{k+1}$.
     Si $i = j$: $x_j = y_j$.
     Si $i > j$: $y_j = (f_j^P \circ f_{j+1}^P \circ \cdots \circ f_{i-1}^P)(x_i) = x_j$,
     car $\forall k, f_k^P(x_{k+1}) = x_k$.

3. Pour montrer que $I\!L = (L, \{\xi_i : L_i \to L,\ i \in I\!N\})$ est un cocône, il faut montrer que $\forall i \in I\!N : \xi_{i+1} \circ f_i = \xi_i$.

Soit $l \in L_i$, et $\xi_i(l) = (x_0, x_1, x_2, \cdots)$.
Soit $\xi_{i+1}(f_i(l)) = (y_0, y_1, y_2 \cdots)$.

Pour $j < i$:
$x_j = (f_{j-1} \circ f_{j-2} \circ \cdots \circ f_{i+1} \circ f_i)(l)$
$= (f_{j-1} \circ f_{j-2} \circ \cdots \circ f_{i+1})(f_i)(l)$
$= y_j$
Pour $j = i$: $x_j = l = f_{i-1}^P(f_i(l)) = y_j$
Pour $j > i$:
$x_j = (f_j^P \circ f_{j+1}^P \circ \cdots \circ f_{i-1}^P)(l)$
$= (f_j^P \circ f_{j+1}^P \circ \cdots \circ f_{i-1}^P)(f_i^P(f_i(l)))$
$= y_j$

On montre que les plongements ξ_i satisfont l'égalité suivante:

$$\bigsqcup_{i=0}^{\infty} \xi_i \circ \xi_i^P = id_L$$

Soit $j \in I\!N$:

	$f_j \circ f_j^P \sqsubseteq id_{L_j}$	(f_j est un plongement)
$\Rightarrow$	$\xi_{j+1} \circ f_j \circ f_j^P \circ \xi_{j+1}^P \sqsubseteq \xi_{j+1} \circ \xi_{j+1}^P$	(ξ_{j+1} est croissante)
$\Rightarrow$	$\xi_j \circ \xi_j^P \sqsubseteq \xi_{j+1} \circ \xi_{j+1}^P$	($I\!L$ est un cocône, donc $\xi_j = \xi_{j+1} \circ f_j$), et donc $\xi_j^P = f_j^P \circ \xi_{j+1}^P$)

On a donc bien une chaine croissante:

$$\xi_0 \circ \xi_0^P \sqsubseteq \xi_0 \circ \xi_0^P \sqsubseteq \xi_0 \circ \xi_0^P \sqsubseteq \cdots$$

De plus: $\forall i,\ \xi_i \circ \xi_i^P \sqsubseteq id_L$.

Il reste à montrer que:
$\forall h : L \to L$, tel que $\forall i,\ \xi_i \circ \xi_i^P \sqsubseteq h$, on a: $id_L \sqsubseteq h$.
Soit $l = (l_0, l_1, l_2, \cdots) \in L$.
Soit $h(l) = (h_0, h_1, h_2, \cdots)$.
$l \sqsubseteq h(l) \Leftrightarrow \forall k :\ l_k \sqsubseteq h_k$.
$\xi_k \circ \xi_k^P(l) \sqsubseteq h(l) \Rightarrow \xi_k(l_k) \sqsubseteq h(l)$.
La k-ième composante de $\xi_k(l_k)$ est l_k. La k-ième composante de $h(l)$ est h_k. Par conséquent: $l_k \sqsubseteq l_k$.

4. Pour montrer que $I\!L = (L, \{\xi_i : L_i \to L,\ i \in I\!N\})$ est un cocône colimite dans **CPOS**, on considère un autre cocône dans **CPOS**: $I\!M = (M, \{\mu_i : L_i \to M,\ i \in I\!N\})$.
Il faut montrer qu'il existe une unique fonction continue stricte $\psi : L \to M$, telle

que: $\forall i : \psi \circ \xi_i = \mu_i$.

Soit $j \in \mathbb{N}$:

	$f_j \circ f_j^P \sqsubseteq id_{L_j}$	(f_j est un plongement)
$\Rightarrow$	$\mu_{j+1} \circ f_j \circ f_j^P \circ \xi_{j+1}^P \sqsubseteq \mu_{j+1} \circ \xi_{j+1}^P$	(μ_{j+1} est croissante)
$\Rightarrow$	$\mu_j \circ f_j^P \circ \xi_{j+1}^P \sqsubseteq \mu_{j+1} \circ \xi_{j+1}^P$	($\mathbb{M}$ est un cocône)
$\Rightarrow$	$\mu_j \circ \xi_j^P \sqsubseteq \mu_{j+1} \circ \xi_{j+1}^P$	($\mathbb{L}$ est un cocône, donc $\xi_j = \xi_{j+1} \circ f_j$, et donc $\xi_j^P = f_j^P \circ \xi_{j+1}^P$)

On a donc une chaine croissante de fonctions de L dans M:
$\mu_0 \circ \xi_0^P \sqsubseteq \mu_1 \circ \xi_1^P \sqsubseteq \mu_2 \circ \xi_2^P \sqsubseteq \cdots$.
On peut donc considérer la fonction:

$$\psi = \bigsqcup_{j=0}^{\infty} (\mu_j \circ \xi_j^P)$$

Soit $j > i$. On a:

$$\xi_i = \xi_j \circ f_{j-1} \circ \cdots \circ f_{i+1} \circ f_i \Rightarrow$$
$$\xi_j^P \circ \xi_i = f_{j-1} \circ \cdots \circ f_{i+1} \circ f_i \Rightarrow$$
$$\mu_j \circ \xi_j^P \circ \xi_i = \mu_j \circ f_{j-1} \circ \cdots \circ f_{i+1} \circ f_i = \mu_i$$

On a donc:

$$\psi \circ \xi_i = \bigsqcup_{j=0}^{\infty} \mu_j \circ \xi_j^P \circ \xi_i$$
$$= \bigsqcup_{j>i}^{\infty} \mu_j \circ \xi_j^P \circ \xi_i$$
$$= \bigsqcup_{j>i}^{\infty} \mu_i = \mu_i$$

Il faut maintenant montrer que ψ est l'unique fonction telle que:

$$\forall i : \psi \circ \xi_i = \mu_i$$

Soit ψ' vérifiant cette propriété.

On a:

$$\bigsqcup_{i=0}^{\infty} \xi_i \circ \xi_i^P = id_L$$

Donc:

$$\psi' = \psi' \circ (\bigsqcup_{i=0}^{\infty} \xi_i \circ \xi_i^P)$$

$$= \bigsqcup_{i=0}^{\infty} \psi' \circ \xi_i \circ \xi_i^P = \bigsqcup_{i=0}^{\infty} \mu_i \circ \xi_i^P = \psi$$

Il reste à montrer que ψ est stricte, ce qui est évident.

Remarque 1:

Pour montrer que $\mathbb{L}$ est un cocône colimite, on a utilisé uniquement le fait que les ξ_i sont des plongements et que:

$$\bigsqcup_{i=0}^{\infty} \xi_i \circ \xi_i^P = id_L$$

On n'a pas utilisé la construction de de L et ξ_i. On a donc le lemme suivant:

Lemme 4:

Soit un ω-diagramme dans **PLG**: $L_0 \stackrel{f_0}{\to} L_1 \stackrel{f_1}{\to} L_2 \cdots$.
Soit $\mathbb{L} = (L, \{\xi_i : L_i \to L,\ i \in I\!N\})$ un cocône dans **CPOS**.
On a: $\mathbb{L}$ est un cocône colimite dans **CPOS** du diagramme
$L_0 \stackrel{f_0}{\to} L_1 \stackrel{f_1}{\to} L_2 \cdots$ si et seulement si:

- $\forall \xi_i : L_i \to L$ est un plongement
- $\bigsqcup_{i=0}^{\infty} \xi_i \circ \xi_i^P = id_L$.

Remarque 2:

On s'est placé dans la catégorie **CPOS**. On aurait pu faire la même démonstration dans la catégorie **CPO**. (C'est-à-dire: $\mathbb{L}$ est un cocône colimite dans **CPO**.) Tennent ([$Ten91$]) montre la même chose dans la catégorie **PLG** des plongements: si $\forall i, \mu_i$ est un plongement, alors ψ est un plongement.
On s'est placé dans la catégorie **CPOS** pour la raison suivante: pour appliquer la théorème de Plotkin et Smyth, on doit se placer dans une catégorie possédant un objet initial. On ne peut donc pas se placer dans la catégorie **CPO**. Tennent redémontre le théorème de Plotkin et Smyth dans **PLG**. Nous nous intéressons à la propriété d'initialité dans une catégorie la plus grande possible, c'est pourquoi nous nous plaçons dans **CPOS**, plutôt que dans **PLG**.

3.2.5 Conservation d'ω-colimites

Pour appliquer le théorème de Plotkin et Smyth, on doit considérer des foncteurs conservant les ω-colimites. On montre dans ce paragraphe que tous les foncteurs et bifoncteurs que nous avons considérés jusqu'à maintenent conservent les ω-colimites.

Lemme 5:

Dans toute catégorie, on a:
Pour tout objet Z, le foncteur constant $Z\bullet$ conserve les colimites.
Le foncteur identité $I\!I$ conserve les colimites.

Preuve:

évident.

Lemme 6:

Soit C une catégorie admettant des ω-colimites. Soit $F, G : \mathbf{C} \to \mathbf{C}$ deux endofoncteurs conservant les ω-colimites. Alors le foncteur $G \circ F$ conserve les ω-colimites.

Preuve:

évident.

Lemme 7:

Soit $\dagger$ un bifoncteur continu.
Soit $\mathbf{D} = D_0 \stackrel{d_0}{\to} D_1 \stackrel{d_1}{\to} D_2 \cdots$ un diagramme sur **PLG**, ayant pour colimite le cocône $I\!D = (D, \{\alpha_i : D_i \to D,\ i \in I\!N\})$.
Soit $\mathbf{E} = E_0 \stackrel{e_0}{\to} E_1 \stackrel{e_1}{\to} E_2 \cdots$ un diagramme sur **PLG**, ayant pour colimite le cocône $I\!E = (E, \{\beta_i : E_i \to E,\ i \in I\!N\})$.
Alors le diagramme $\mathbf{D} \dagger \mathbf{E} = D_0 \dagger E_0 \stackrel{d_0 \dagger e_0}{\to} D_1 \dagger E_1 \stackrel{d_1 \dagger e_1}{\to} D_2 \dagger E_2 \cdots$ a pour colimite: $I\!D \dagger I\!E = (D \dagger E, \{\alpha_i \dagger \beta_i : D_i \dagger E_i \to D \dagger E,\ i \in I\!N\})$.

Preuve:

α_i et β_i sont des plongements, $\dagger$ est continu, donc $\alpha_i \dagger \beta_i$ est un plongement, et $(\alpha_i \dagger \beta_i)^P = \alpha_i^P \dagger \beta_i^P$. (Lemme 2).
Pour montrer que le diagramme $\mathbf{D} \dagger \mathbf{E}$ a pour colimite $I\!D \dagger I\!E$, il suffit de montrer que: (lemme 4)

$$\bigsqcup_{i=0}^{\infty} (\alpha_i \dagger \beta_i) \circ (\alpha_I \dagger \beta_i)^P = id_{D \dagger E}$$

$$
\begin{aligned}
&\bigsqcup_{i=0}^{\infty}(\alpha_i \dagger \beta_i) \circ (\alpha_i \dagger \beta_i)^P \\
&= \bigsqcup_{i=0}^{\infty}(\alpha_i \dagger \beta_i) \circ (\alpha_i^P \dagger \beta_i^P) && \text{(Lemme 2)} \\
&= \bigsqcup_{i=0}^{\infty}(\alpha_i \circ \alpha_i^P) \dagger (\beta_i \circ \beta_i^P) && (\dagger \text{ est un bifoncteur.}) \\
&= (\bigsqcup_{i=0}^{\infty} \alpha_i \circ \alpha_i^P) \dagger (\bigsqcup_{i=0}^{\infty} \beta_i \circ \beta_i^P) && (\dagger \text{ est continu}) \\
&= id_D \dagger id_E && \text{(Lemme 4 appliqué à } \mathbb{D} \text{ et } \mathbb{E}) \\
&= id_{D \dagger E} && (\dagger \text{ est un foncteur}).
\end{aligned}
$$

Résultat:

Soit F un foncteur construit par composition à partir de foncteurs constants, du foncteur identité, et des bifoncteurs $\times$, $+$, $\otimes$, $\oplus$. Alors F conserve les ω-colimites d' ω-diagrammes de **PLG**.

Preuve:

Application immédiate des trois lemmes précédents (lemmes 5, 6 et 7).

3.2.6 Application du théorème de Plotkin et Smyth dans CPOS

On se place dans la catégorie **CPOS**. Cette catégorie a un objet initial: $\{\bot\}$, c'est-à-dire pour tout CPO B, il existe une unique fonction stricte $j : \{\bot\} \to B$. Cette fonction j est un plongement, et j^P est la fonction constante égale à $\bot$.

Soit F un endofoncteur sur **CPOS**, construit à partir des foncteurs constants, du foncteur identité et des bifoncteurs $\times$, $+$, $\otimes$, et $\oplus$.

Soit l_0 l'unique fonction stricte de $\{\bot\}$ dans $F(\{\bot\})$.
Le diagramme $\{\bot\} \stackrel{l_0}{\to} F(\{\bot\}) \stackrel{F(l_0)}{\to} F^2(\{\bot\}) \stackrel{F^2(l_0)}{\to} F^3(\{\bot\}) \cdots$ est donc un diagramme dans **PLG**. Par conséquent, il a un cocône colimite.

Les hypothèses du théorème de Plotkin et Smyth sont vérifiées, donc il existe un objet L et un isomorphisme $\psi : F(L) \to L$. L'algèbre (L, ψ) est initiale dans la catégorie $(F : \mathbf{CPOS})$ des F-algèbres sur **CPOS**.

Cette algèbre donne une sémantique au type abstrait représenté par l'"équation aux domaines" $X \cong F(X)$.

Remarque:

On peut se demander si cette algèbre est initiale dans la catégorie des F-agèbres sur **CPO**. Ce n'est pas le cas. Si (A, a) est une F-algèbre, où a est non strict, il peut exister plusieurs homomorphismes de (L, ψ) dans (A, a).
(Un tel exemple est donné en 3.2.7, dans le paragraphe sur les séquences infinies).

Test d'égalité de deux fonctions:

Soit (L, ϕ) la F-algèbre initiale de $(F : \mathbf{CPOS})$. Soit (A, a) une autre F-algèbre de $(F : \mathbf{CPOS})$. Soit h, l'unique homomorphisme de (L, ϕ) dans (A, a).
Soit une fonction $f : L \to A$. Pour montrer que $f = h$, il suffit de montrer que f est un homomorphisme, c'est-à-dire que: $f \circ \psi = a \circ F(f)$.

Remarque: exponentielle partielle

Il est d'usage de présenter, en même temps que les bifoncteurs $\times$, $+$, $\otimes$, $\oplus$, l'opérateur d'exponentielle partielle $\to$. $[A \to B]$ est le CPO des fonctions continues de A dans B.
Soit $f : B \to A$, et $f' : A' \to B'$.

$$\begin{array}{rccc} (f \to f') : & [A \to A'] & \to & [B \to B'] \\ & h & \mapsto & f' \circ h \circ f \end{array}$$

Le problème est que $\to$ ainsi défini n'est pas un bifoncteur, car f est une fonction de B dans A.

Plotkin, Smyth, Wand ... résolvent ce problème en se plaçant dans la catégorie **PLG**, et en considérent le bifoncteur $\stackrel{E}{\to}$, défini par:
Pour tout CPO A, B: $A \stackrel{E}{\to} B = [A \to B]$. Soit $f : A \to B$, et $f' : A' \to B'$.

$$\begin{array}{rccc} (f \to f') : & [A \to A'] & \to & [B \to B'] \\ & h & \mapsto & f' \circ h \circ f^P \end{array}$$

$\stackrel{E}{\to}$ est bien un foncteur.
Cela ne résout pas tous les problèmes: lorsqu'on applique le théorème de Plotkin et Smyth, on obtient une algèbre initiale dans la catégorie $(F : \mathbf{PLG})$, alors qu'on cherche une algèbre initiale dans $(F : \mathbf{CPOS})$.

3.2.7 Exemples de constructions de types abstraits

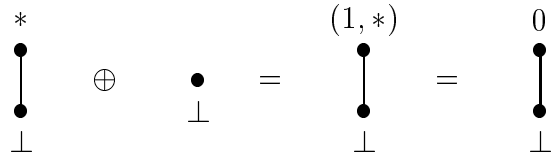
Dans ce paragraphe, on note $1 = \{\bot\}$ le CPO comportant un seul élément, et $1_\bot = \{\bot, *\}$ le CPO comportant deux éléments.

Entiers naturels stricts

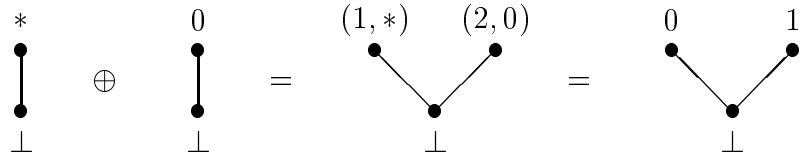
Soit le foncteur F, défini par:
Pour tout objet A, $F(A) = 1_\bot \oplus A$.
Pour toute flèche f, $F(f) = id_{1_\bot} \oplus f$.

Calculons les premiers termes de la suite $F^j(\{\bot\})$:

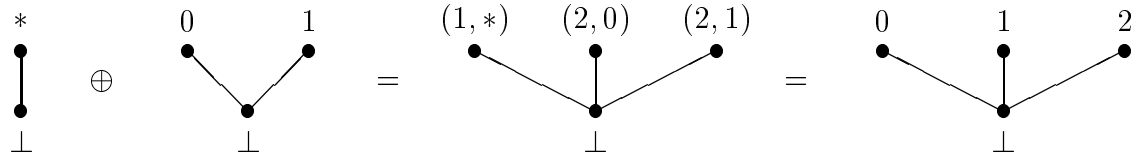
$1 = \{\bot\} =$ 

$F(1) = 1_\bot \oplus 1 =$ 

$F^2(1) = 1_\bot \oplus (1_\bot \oplus 1) =$

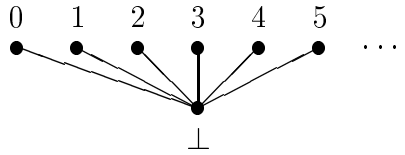


$F^3(1) = 1_\bot \oplus (1_\bot \oplus (1_\bot \oplus 1)) =$



...

On construit donc ainsi l'ensemble des entiers naturels stricts $\mathbb{N}_\bot$.

$\mathbb{N}_\bot =$ 

L'isomorphisme $\psi : F(\mathbb{N}_\bot) \to \mathbb{N}_\bot$ est donné par:

$$\begin{array}{rlll} \psi : & 1_\bot \oplus \mathbb{N}_\bot & \to & \mathbb{N}_\bot \\ & \bot & \mapsto & \bot \\ & (1, *) & \mapsto & 0 \\ & (2, n) & \mapsto & n + 1 \end{array}$$

On a les injections $i : 1_\bot \to 1_\bot \oplus \mathbb{N}_\bot$, et $i' : \mathbb{N}_\bot \to 1_\bot \oplus \mathbb{N}_\bot$.

On pose:

$$\begin{array}{lrcl} z = \psi \circ i : & 1_\perp & \to & I\!N_\perp \\ & \perp & \mapsto & \perp \\ & * & \mapsto & 0 \\ s = \psi \circ i' : & I\!N_\perp & \to & I\!N_\perp \\ & \perp & \mapsto & \perp \\ & n & \mapsto & n+1 \end{array}$$

On retrouve ici les constructeurs classiques z et s: z est la fonction constante égale à 0, et s est la fonction *successeur*. Ces deux fonctions permettent d'obtenir tous les entiers naturels:
z est la flèche qui à $*$ associe 0.
$s \circ z$ est la flèche qui à $*$ associe 1.
$s \circ s \circ z$ est la flèche qui à $*$ associe 2...

Nous allons définir l'addition par homomorphisme.
En utilisant les notations de programmation fonctionnelle, on définit l'addition par:

```
add   0  y = y
add s(x) y = s(add x y)
```

Dans cette définition, $add \in [I\!N_\perp \times I\!N_\perp \to I\!N_\perp]$. On peut également considérer que $add \in [I\!N_\perp \to [I\!N_\perp \to I\!N_\perp]]$:

```
add 0    = id
add s(x) = s o (add x)
```

Considérons maintenant l'algèbre $([I\!N_\perp \to I\!N_\perp], \eta)$:

$$\begin{array}{lrcl} \eta : & 1_\perp + [I\!N_\perp \to I\!N_\perp] & \to & [I\!N_\perp \to I\!N_\perp] \\ & (1, *) & \mapsto & id_{I\!N_\perp} \\ & (2, f) & \mapsto & s \circ f \end{array}$$

On vérifie que l'on a:

$$add \ \circ \psi = \eta \circ (id_{1_\perp} \oplus add)$$

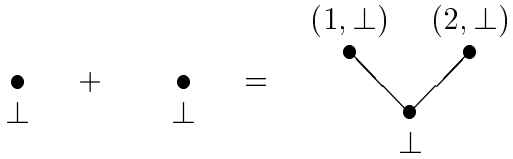
$$\Leftrightarrow \begin{cases} add & \perp_{I\!N_\perp} & = & \perp_{[I\!N_\perp \to I\!N_\perp]} \\ add & 0 & = & id_{I\!N_\perp} \\ add & s(x) & = & s \circ (add\ x) \end{cases}$$

On définit donc l'addition comme l'unique homomorphisme entre $(I\!N_\perp, \psi)$ et $([I\!N_\perp \to I\!N_\perp], \eta)$. add est l'unique fonction telle que $add \circ \psi = \eta \circ (id_{1_\perp} \oplus add)$.

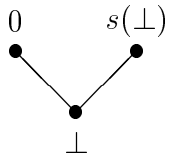
Entiers naturels paresseux:

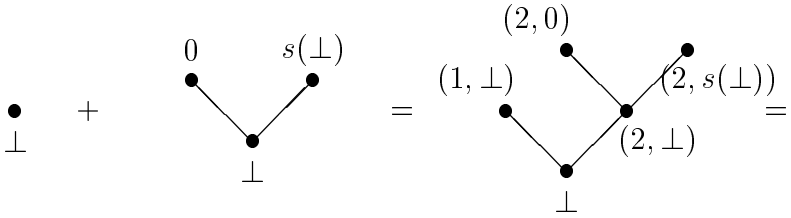
Soit le foncteur F défini par:
Pour tout objet A, $F(A) = 1 + A$
Pour toute flèche f, $F(f) = id_1 + f$.

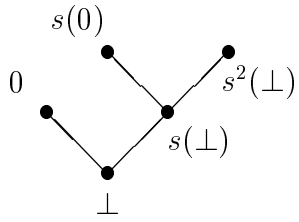
$1 =$ 

$F(1) = 1 + 1 =$ 

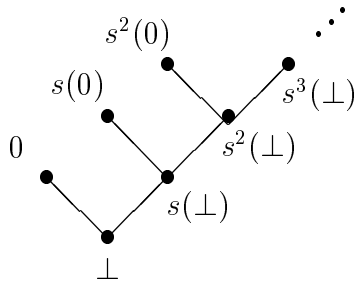
On note $(1, \perp) = 0$, et $(2, x) = s(x)$.
On a donc:

$F(1) =$ 

$F^2(1) = 1 + (1 + 1) =$ 



On construit ainsi l'ensemble des entiers naturels paresseux N_P:

$N_P =$ 

L'isomorphisme $\psi : 1 + N_P \to N_P$ est donné par:

$$\begin{array}{lccl} \psi: & 1+N_P & \to & N_P \\ & \bot & \mapsto & \bot \\ & (1,*) & \mapsto & 0 \\ & (2,n) & \mapsto & s(n) \end{array}$$

Comme précédemment, la fonction $z = \psi \circ i$ est la fonction constante égale à 0, et $s = \psi \circ i'$ est la fonction successeur.

Remarque: On a une chaine croissante: $\bot \sqsubseteq s(\bot) \sqsubseteq s^2(\bot) \sqsubseteq s^3(\bot) \sqsubseteq \cdots$. Toute chaine croissante a une borne supérieure, donc N_P contient un élément infini: $s^\infty(\bot)$. L'ensemble $I\!N_\bot$ des entiers stricts, par contre, ne contenait pas d'élément infini.

Listes strictes

Les listes strictes d'entiers sont obtenues avec le foncteur F:
$F(A) = 1_\bot \oplus (N \otimes A)$
$F(f) = id_{1_\bot} \oplus (id_N \otimes f)$,
où N est un CPO d'entiers naturels (stricts ou paresseux ...)

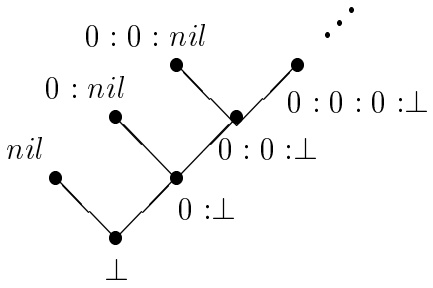
On vérifie que: $F^n(1)$ représente le CPO des listes à $0, 1, \cdots n-1$ éléments.

Soit (L_S, ψ), l'algèbre initiale. L_S est l'ensemble des listes finies.

Soit les injections $i : 1_\bot \to 1_\bot \oplus (N \otimes L_S)$, et $i' : N \otimes L_S \to 1_\bot \oplus (N \otimes L_S)$. $\psi \circ i$ est la fonction constante égale à *nil*. $\psi \circ i'$ est la fonction **stricte** *cons*.

Listes paresseuses

Les listes paresseuses d'entiers sont obtenues avec le foncteur F:
$F(A) = 1 + (N \times A)$
$F(f) = id_1 + (id_N \times f)$.
Si, pour simplifier, on considère que N contient un seul entier 0, on obtient la même structure que celle des entiers naturels paresseux (l'ensemble des listes sur un ensemble à un élément est isomorphe à l'ensemble des entiers).



Soit (L_P, ψ) l'algèbre initiale.
La fonction $\psi \circ i$ est la fonction constante égale à *nil*. La fonction $\psi \circ i'$ est la fonction **non stricte** *cons*.
Par exemple, $cons(0, \perp) = 0 : \perp$ (liste dont le premier élément est 0, et le reste non défini).
Comme pour les entiers naturels paresseux, on a des chaines croissantes de la forme:

$$\perp \sqsubseteq cons(0, \perp) \sqsubseteq cons(0, cons(0, \perp)) \sqsubseteq cons(0, cons(0, cons(0, \perp))) \cdots$$

Toute chaine croissante ayant une borne supérieure, L_P contient toutes les listes *infinies* d'entiers.
Le CPO des listes strictes ne contenait que les liste *finies*.

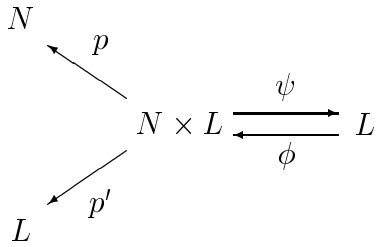
Séquences infinies

Soit le foncteur F défini par:
Pour tout objet A, $F(A) = N \times A$
Pour toute flèche f, $F(f) = id_N \times f$.

Calculons les premiers $F^i(1)$:
$F(1) = N \times 1$
$F^2(1) = N \times (N \times 1))$
$F^3(1) = N \times (N \times (N \times 1)))$

Soit (L, ψ), l'algèbre initiale de la catégorie $(F : \mathbf{CPOS})$. L est l'ensemble des séquences infinies sur N, qui seront notées: $[n_0, n_1, n_2, \cdots]$. Le plus petit élément de L est la séquence infinies de $\perp_N$: $\perp_L = [\perp_N, \perp_N, \perp_N, \cdots]$.

Soit $\phi = \psi^{-1}$.



ψ est la fonction qui ajoute un élément en tête de séquence: $\psi = cons$.
Remarque: ψ est stricte, c'est-à-dire que $\psi(\perp_N, \perp_L) = \perp_L$. Par contre ψ n'est pas *bistricte*, c'est-à-dire que $\psi(n, l) \neq \perp_L$, si $n \neq \perp_N$ ou $l \neq \perp_L$.
$head = p \circ \phi : L \to N$ renvoie le premier élément d'une séquence.
$tail = p' \circ \phi : L \to L$ enlève le premier élément à une séquence.
head et *tail* sont strictes.

Soit une autre F-algèbre (M, η).
D'après le théorème de Plotkin et Smyth, si η est stricte, alors il existe un unique $h : L \to M$ tel que: $h \circ \psi = \eta \circ F(h)$. (Il est facile de voir que h est alors forcément strict).

Le théorème de Plotkin ne nous dit rien si η n'est pas strict. Nous allons donner un exemple où η n'est pas strict, et où il existe plusieurs homomorphismes entre (L, ψ) et (M, η).

Soit $\begin{array}{rccl} \eta : & N \times L & \to & L \\ & (n, l) & \mapsto & cons(0, tail(l)) \end{array}$
En termes purement fonctionnels, η est définie par:

$$\eta = cons \circ < z \circ t_N \circ p, tail \circ p' >,$$

où $z : 1 \to N$ est la fonction constante égale à 0, $t_N : N \to 1$ est l'unique flèche de N dans 1. $z \circ t_N : N \to N$ est la fonction constante égale à 0.

Soit $h : L \to M$ vérifiant $h \circ \psi = \eta \circ F(h)$.

$$h \circ \psi = \eta \circ F(h) \Leftrightarrow$$

Pour toute séquence $[n_1, n_2, n_3, \cdots]$, et $\forall n \in N$:

$$h \circ \psi(n, [n_1, n_2, n_3, \cdots]) = \eta \circ F(h)(n, [n_1, n_2, n_3, \cdots]) \Leftrightarrow$$

$$h([n, n_1, n_2, n_3, \cdots]) = \eta(n, h([n_1, n_2, n_3, \cdots])) \Leftrightarrow$$

$$h([n, n_1, n_2, n_3, \cdots]) = cons(0, tail(h([n_1, n_2, n_3, \cdots])))$$

On remarque que cette relation est satisfaite pour toute fonction h qui renvoie une séquence constante dont le premier élément est 0.
Il n'y a donc pas unicité de l'homomorphisme entre (L, ψ) et (M, η). (L, ψ) est bien initiale dans $(F : \mathbf{CPOS})$, mais pas dans $(F : \mathbf{CPO})$.

3.3 Types abstraits dans SET

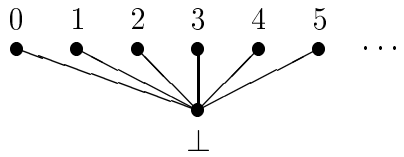
Dans cette partie, on montre que certains résultats démontrés dans **CPOS** peuvent s'appliquer directement dans **SET**. Pour cela, on considère un ensemble comme un CPO *plat*, et on considère une fonction entre deux ensembles comme une fonction *continue très stricte* entre deux CPO plats. On obtient ainsi une sous catégorie de **CPOS**.

Définition: CPO plat

Un CPO A est plat (*flat* en anglais) si

$$\forall a, b \in A : \quad a \sqsubseteq b \Leftrightarrow (a = \perp \quad ou \quad a = b)$$

Exemple de CPO plat:

$\mathbb{N}_\perp =$ 

Définition: fonction très stricte

Une fonction $f : A \to B$ est très stricte (*very strict* en anglais) si:

$$\forall a \in A : \quad f(a) = \perp_B \Leftrightarrow a = \perp_A$$

Une fonction très stricte est donc stricte.

Catéorie PTS, isomorphe à SET

On a une catégorie **PTS**, des CPO plats, avec les fonctions continues très strictes. Cette catégorie est isomorphe à **SET**: à chaque ensemble A, on fait correspondre le CPO plat $A_\perp = A \cup \{\perp\}$. A chaque application $f : A \to B$, on fait correspondre la fonction continue très stricte $f_\perp : A_\perp \to B_\perp$.

Foncteurs dans PTS et SET

En 3.2.2, on a défini les foncteurs $\times$, $+$, $\otimes$, et $\oplus$. On regarde si on peut restreindre ces foncteurs à **PTS**.
Etant donné deux CPO plats A et B, $A \times B$ et $A + B$ ne sont pas des CPO plats. On ne peut donc pas appliquer ces deux foncteurs dans **PTS**. Par contre $A \otimes B$ et $A \oplus B$ sont bien des CPO plats. De plus, si f et g sont deux fonctions très strictes, $f \oplus g$ et $f \otimes g$ sont également très strictes.
Dans **PTS**, on dispose donc des foncteurs $\otimes$ et $\oplus$. On peut vérifier que ces foncteurs correspondent au produit et à la somme dans **SET**, qui sont notés habituellement $\times$ et $+$, et ne doivent pas être confondus avec les foncteurs $\times$ et $+$ de la catégorie **CPO**.

Plongements

On sait déjà que les plongements sont des fonctions strictes. Ce sont également des fonctions très strictes:

$$f(a) = \perp \Rightarrow \ f^P(f(a)) = f^P(\perp) \ \Rightarrow \ a = \perp$$

Les plongements de **PTS** correspondent aux applications *injectives* de **SET**.
Il faut remarquer que étant donné un plongement f, f^P n'est pas une fonction très stricte (c'est seulement une fonction stricte). La flèche f^P n'a donc pas d'équivalent dans **SET** (cette flèche fait partie de la catégorie des ensembles et des fonctions *partielles*).

Construction d'ω-colimites

Soit des CPO plats $L_0, L_1, L_2, \cdots$, et des plongements $f_i : L_i \to L_{i+1}$.
Alors le diagramme $L_0 \stackrel{f_0}{\to} L_1 \stackrel{f_1}{\to} L_2 \cdots$ a un cocône colimite dans **PTS**.

On fait exactement la même construction qu'en 3.2.4. Il suffit de vérifier que:

- Le CPO L est plat.
- Les flèches $\xi_i : L_i \to L$ sont très strictes.
- La flèche $\psi : L \to M$ est très stricte.

Application du théorème de Plotkin et Smyth:

On se place dans la catégorie **SET**. Dans cette catégorie, $\emptyset$ est initial.
On considère un foncteur F *polynomial*, c'est-à-dire construit par composition à partir de foncteurs constants, du foncteur identité, et des bifoncteurs $\times$ et $+$.
On considère le diagramme $\{\emptyset\} \stackrel{\emptyset}{\to} F(\{\emptyset\}) \stackrel{F(\emptyset)}{\to} F^2(\{\emptyset\}) \stackrel{F^2(\emptyset)}{\to} F^3(\{\emptyset\}) \cdots$.
Toutes les flèches sont des injections, donc ce diagramme admet un cocône colimite.
Le foncteur F conserve ce cocône colimite.
Par conséquent, il existe un objet L et un isomorphisme $\psi : F(L) \to L$. L'algèbre (L, ψ) est initiale dans la catégorie $(F : \mathbf{SET})$.

Comparaison avec les types abstraits algébriques:

Dans ce paragraphe, on donne quelques exemples de types abstraits algébriques, et les définitions correspondantes en termes de plus petit point fixe d'endofoncteur.

- Déclaration d'un seul type:
  Soit la signature:

  ```
  0: -> nat
  s: nat -> nat
  ```

  Cette signature correspond au foncteur:
  $F(X) = 1 + X$
  $F(f) = id_1 + f$

  Σ-algèbre et F-algèbre:
  Une Σ-algèbre A est donnée par un ensemble A, la constante $0_A \in A$, et l'opération $s_A : A \to A$.
  Une F-algèbre est donnée par un ensemble L, une application $l : 1 + L \to L$. Donner cette application l revient à donner les deux applications $l \circ i : 1 \Rightarrow L$, et $l \circ i' : L \to L$, qui correspondent à 0_A et à s_A.

Homomorphisme:
Soit A et B deux Σ-algèbres.
Soit $h : A \to B$ un homomorphisme d'algèbre. h vérifie:

$$\begin{cases} h(0_A) = 0_B \\ \forall x \in A : \ h(s_A(x)) = s_B(h(x)) \end{cases}$$

On considère maintenant A et B comme des F-algèbres:

$$a : F(A) \to A$$

$$b : F(B) \to B$$

Les équations vérifiées par h deviennent:

$$\left.\begin{array}{c} h \circ a \circ i = b \circ i \\ h \circ a \circ i' = b \circ i' \circ h \end{array}\right\} \Leftrightarrow$$

$$h \circ a = b \circ (id_1 + h)$$

On voit sur cet exemple que les définitions d'algèbres et d'homomorphismes correspondent, la Σ-algèbre initiale T_Σ et la F-algèbre initiale doivent donc correspondre également.

- Déclarations croisées de plusieurs types:
On définit le type abstrait des arbres non vides et des forêts avec la signature suivante:

```
fvide: -> foret
fcons: (arbre, foret) -> foret
noeud: (nat,foret) -> arbre
```

Cette sinature va être représentée par le foncteur:

$$\begin{array}{rccc} F: & \mathbf{C} \times \mathbf{C} & \to & \mathbf{C} \times \mathbf{C} \\ & (A, f) & \mapsto & (N \times F, 1 + A \times F) \\ & (a, f) & \mapsto & (id_N \times f, id_1 + a \times f) \end{array}$$

Il est facile de voir que si le théorème de Plotkin et Smyth s'applique dans une catégorie $\mathbf{C}$, alors il peut s'appliquer dans la catégorie produit $\mathbf{C} \times \mathbf{C}$.
Soit $((A, F), (a, f))$ la F-algègre initiale de $(F : \mathbf{C} \times \mathbf{C})$.
$a : N \times F \to A$
$f : 1 + A \times F \to F$.
a correspond à l'opérateur *noeud*, $f \circ i$ correspond à *fvide*, et $f \circ i'$ correspond à *fcons*.

3.4 Discussion à propos d'autres travaux

Dans ce paragraphe, nous revenons un instant sur les travaux de de Malcolm ([$Mal89$], [$Mal90$]), et de Meijer, Fokkinga et Paterson ([$MFP91$]).

3.4.1 Dualité

On obtient le dual d'un énoncé concernant une catégorie en retournant toutes les flèches et toutes les compositions. Le principe de dualité dit que le dual de tout théorème est un théorème (car dans la définition d'une catégorie, le dual de tout axiome est un axiome).

Définition: coalgèbre

Etant donné un endofoncteur F sur une catégorie $\mathbf{C}$, on peut définir une F-coalgèbre comme le dual d'une algèbre: c'est un couple (Z, z) où $z : Z \to F(Z)$.

On a vu que l'on peut définir des types abstraits comme algèbre initiale dans une catégorie de F-algèbre. On peut également, par dualité, définir un type abstrait comme coalgèbre terminale d'une catégorie de F-coalgèbres.

Dual du théorème de Plotkin et Smyth:

Soit $\mathbf{C}$ une catégorie ayant un objet *terminal* L_0.

Soit F un endofoncteur sur $\mathbf{C}$.

On suppose que le diagramme

$$L_0 \xleftarrow{\theta_0} F(L_0) \xleftarrow{F(\theta_0)} F^2(L_0) \xleftarrow{F^2(\theta_0)} F^3(L_0) \cdots$$

admet une *limite*.
(θ_0 est l'unique flèche de $F(L_0)$ vers L_0.)
On suppose que F conserve cette *limite*.
Alors il existe un objet L et un isomorphisme $\phi : L \to F(L)$.
De plus la coalgèbre (L, ϕ) est *terminale* dans la catégorie des F-coalgèbres.

Approche de Malcolm:

Malcolm se place dans la catégorie **SET**. Dans cette catégorie, $\emptyset$ est l'objet initial, et $1 = \{*\}$ l'objet terminal. On a donc deux moyens de construire des types abstraits: soit comme *colimite* du diagramme:

$$\emptyset \xrightarrow{\emptyset} F(\emptyset) \xrightarrow{F(\emptyset)} F^2(\emptyset) \xrightarrow{F^2(\emptyset)} F^3(\emptyset) \cdots$$

soit comme *limite* du diagramme:

$$1 \xleftarrow{1} F(1) \xleftarrow{F(1)} F^2(1) \xleftarrow{F^2(1)} F^3(1) \cdots$$

Pour un même foncteur, on n'obtient pas forcément le même type. Par exemple considérons le foncteur F défini par:
On se fixe un objet A.
Pour tout objet W: $F(W) = A \times W$
Pour toute flèche w: $F(w) = id_A \times w$

La définition de type abstrait par colimite donne le type vide, et la définition du type abstrait comme limite donne le type des séquences infinies d'éléments de A.

Approche de Meijer, Fokkinga et Paterson

Meijer, Fokkinga et Paterson se placent dans la catégorie **CPO**. On a vu que pour pouvoir appliquer le théorème de Plotkin et Smyth, on doit en réalité se placer dans la catégorie **CPOS**. Dans cette catégorie, $\{\bot\}$ est à la fois objet initial et objet terminal.

De plus on peut montrer que le type abstrait défini comme *colimite* du diagramme:

$$\{\bot\} \xrightarrow{\bot} F(\{\bot\}) \xrightarrow{F(\bot)} F^2(\{\bot\}) \xrightarrow{F^2(\bot)} F^3(\{\bot\}) \cdots$$

est le même que le type abstrait défini comme *limite* du diagramme:

$$\{\bot\} \xleftarrow{\bot} F(\{\bot\}) \xleftarrow{F(\bot)} F^2(\{\bot\}) \xleftarrow{F^2(\bot)} F^3(\{\bot\}) \cdots$$

Autrement dit, si (L, ψ) est l'algèbre initiale de la catégorie des F-algèbres, alors (L, ψ^{-1}) est l'objet terminal de la catégorie des F-coalgèbres.

Dans cette approche, utiliser le principe de dualité ne définit donc pas de nouveaux types de données. Par contre, on a un résultat de "terminalité", dual de l'initialité:

Initialité:

Soit une algèbre (M, m) de $(F : \mathbf{CPOS})$. Il existe un unique $h : L \to M$ tel que:

$$h \circ \psi = m \circ F(h)$$

"Terminalité":

Soit une coalgèbre (Z, z) de la catégorie des F-coalgèbres sur **CPOS**. Il existe un unique $k : Z \to L$ tel que:

$$\psi \circ k = F(k) \circ z$$

3.4.2 Catamorphisme, anamorphisme

Soit un endofoncteur F sur une catégorie $\mathbf{C}$. Soit (L, ψ), l'algèbre initiale de $(F : \mathbf{C})$. Soit (M, m) une F-algèbre.

Malcolm appelle *catamorphisme* l'homomorphisme initial de L vers M. Cet homomorphisme est noté $(\!|m|\!)$.

Théorème de promotion:

Soit deux algèbres (M, m) et (N, n). Soit $h : M \to N$. On a:

$$h \circ m = n \circ F(h) \;\Rightarrow\; h \circ (\!|m|\!) = (\!|n|\!)$$

Ce résultat provient de l'initialité de (L, ψ): $(\!|n|\!)$ est l'unique homomorphisme de (L, ψ) vers (N, n). Comme h est un homomorphisme, $h \circ (\!|m|\!)$ est un homomorphisme de (L, ψ) vers (N, n), et donc $h \circ (\!|m|\!) = (\!|n|\!)$.

Meijer, Fokkinga et Paterson définissent également un catamorphisme, non pas à partir de l'initialité de (L, ψ), mais en utilisant l'opérateur μ de plus petit point fixe dans les CPO. Etant donné un CPO D, et une fonction continue $H \in [D \to D]$, $\mu(H)$ est le plus petit élément de D tel que:

$$\mu(H) = H(\mu(H))$$

Catamorphisme:

On note $\lambda h.m \circ F(h) \circ \psi^{-1}$ l'opérateur qui à toute flèche $h : L \to M$ associe la flèche $m \circ F(h) \circ \psi^{-1} : L \to M$.
Le catamorphisme associé à l'algèbre (M, m) est le plus petit point fixe de cet opérateur:

$$(\!|m|\!) = \mu(\lambda h.m \circ F(h) \circ \psi^{-1})$$

$(\!|m|\!)$ est bien un homomorphisme puisque:

$$(\!|m|\!) = m \circ F((\!|m|\!)) \circ \psi^{-1} \Rightarrow (\!|m|\!) \circ \psi = m \circ F((\!|m|\!))$$

Anamorphisme:

Un *anamorphisme* est défini de façon duale: L'anamorphisme associé à la coalgèbre (Z, z) est le plus petit point fixe de l'opérateur $\lambda h.\psi \circ F(h) \circ n$, qui associe à toute flèche $h : Z \to L$ la flèche $\psi \circ F(h) \circ z : Z \to L$.

$$[\!(z)\!] = \mu(\lambda h.\psi \circ F(h) \circ z)$$

Lorsque (M, m) est une algèbre dans **CPOS** (c'est-à-dire lorsque m est stricte), $(\!|m|\!)$ est l'homomorphisme initial de (L, ψ) vers (M, m). En effet, il existe alors un unique homomorphisme de (L, ψ) vers (M, m). Par contre, lorsque (M, m) n'est pas stricte, il peut

exister plusieurs homomorphismes de (L, ψ) vers (M, m). La définition de catamorphisme de Meijer, Fokkinga et Paterson est donc plus générale que celle de Malcolm.

Dans [$MFP91$], tous les théorèmes énoncés sont démontrés à partir des propriétés de l'opérateur de plus petit point fixe μ. Certains peuvent être démontrés à partir de l'initialité de (L, ψ).

- La loi (Catafusion'):
  Soit deux algèbres (M, m) et (N, n). Soit une flèche $f : M \to N$. On a:
  f stricte, $f \circ m = n \circ F(f) \;\Rightarrow\; f \circ (\!|m|\!) = (\!|\phi|\!)$
  Ce résultat est identique au théorème de promotion. Si m et n sont strictes, alors cette propriété provient directement de l'initialité de (L, ψ).

- Les catamorphismes conservent la "strictness":
  Si (M, m) est une algèbre telle que m est stricte, alors $(\!|m|\!)$ est stricte.

Chapitre 4

Equations

4.1 Introduction

Nous avons vu comment représenter des types abstraits libres dans le formalisme des catégories. En spécification algébrique, on peut spécifier des types abstraits non libres, c'est-à-dire dont la définition comporte des équations. Par exemple, on peut spécifier l'ensemble des entiers modulo 2 par:

```
0: -> nat
s: nat -> nat
s(s(0)) = 0
```

Le but de ce chapitre est de montrer que l'on peut intégrer les équations dans le formalisme des catégories. Dans ce formalisme, on raisonne toujours sur les flèches, il faut donc commencer à définir les équations en termes de flèches. Ensuite, il faut définir la notion d'algèbre satisfaisant un ensemble d'équations. On propose enfin une construction catégorielle d'algèbre *initiale* dans la catégorie des algèbres satisfaisant ces équations. Cette algèbre correspond à l'algèbre $T_\Sigma/\equiv$ en spécification algébrique.

Dans tout ce chapitre, on considère une catégorie $\mathbf{C}$, et un endofoncteur F sur $\mathbf{C}$ qui satisfont le théorème de Plotkin et Smyth.
On appelle (A, a) l'algèbre initiale de la catégorie $(F : \mathbf{C})$ des F-algèbres.

4.2 Equations

Dans cette partie nous définissons la notion d'équation et d'algèbre satisfaisant un ensemble d'équations dans le cadre catégoriel. Enfin, nous montrons que dans une catégorie qui admet des sommes, un ensemble d'équations peut être considéré comme équivalent à une seule équation plus complexe.

En spécification algébrique, une équation est définie par un ensemble de variables Var, et

deux termes t et u dont les variables appartiennent à Var. Dans le cadre catégoriel, une équation va être représentée par deux flèches f et g, qui correspondent aux termes t et u paramétrés par leurs variables.

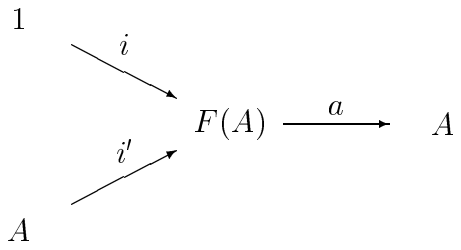
Exemple:

Prenons l'exemple des entiers naturels. En spécification algébrique, on définit la signature Σ:

```
0: -> nat
s: nat -> nat
```

L'ensemble des entiers naturels est donné par l'algèbre initiale T_Σ.
L'équation `s(s(x)) = x` définit l'ensemble des entiers modulo 2. L'ensemble des entiers modulo 2 peut être spécifié de manière équivalente soit par l'équation `s(s(x)) = x`, soit par l'équation `s(s(0)) = 0`.

Dans le cadre catégoriel, on définit le foncteur F par:
$F(W) = 1 + W$
$F(w) = id_1 + w$
Soit (A, a) l'algèbre initiale.



On pose $a \circ i = z$ et $a \circ i' = s$.
L'équation `s(s(x)) = x` correspond à l'équation catégorielle $e_1 = (s \circ s \approx id_A)$.
L'équation `s(s(0)) = 0` correspond à l'équation catégorielle $e_2 = (s \circ s \circ z \approx z)$.
L'équation e_1 consiste en deux flèches $s \circ s,\ id_A : A \to A$.
L'équation e_2 consiste en deux flèches $s \circ s \circ z,\ z : 1 \to A$.
Le domaine des deux flèches de l'équation dépend des variables que l'on a du supprimer pour réaliser l'abstraction. Typiquement, ce domaine sera $1,\ A,\ A \times A, \cdots$. On verra en 4.2.4 que l'on peut considérer des domaines plus compliqués.

4.2.1 Définition: équation

Une *équation* est donnée par un objet X de $\mathbf{C}$, et par deux flèches $f, g : X \to A$.
On note $(f \approx g)$, ou, lorsque l'on veut rappeler les domaines et codomaines de f et g:

$$X \underset{g}{\overset{f}{\rightrightarrows}} A$$

4.2.2 Définition: F-algèbre satisfaisant des équations

On considère un ensemble d'équations:
$E = \{f_0 \approx g_0, f_1 \approx g_1, \cdots, f_n \approx g_n\}$, avec $f_i, g_i : X_i \to A$.

Soit (M, m) une F-algèbre. Soit h_M l'unique homomorphisme de (A, a) dans (M, m). On dit que (M, m) *satisfait les équations de* E, ou que (M, m) *est une F-E-algèbre si:*

$$\forall i \in \{0, 1, \cdots n\} : \; h_M \circ f_i = h_M \circ g_i$$

Cette définition correspond à la définition de Σ-E-algèbre en spécification algébrique lorsque (M, m) est une algèbre engendrée. (Les égalités $h_M \circ f_i = h_M \circ g_i$ n'imposent une contrainte que sur les éléments atteints par h_M).

4.2.3 Catégorie des F-E-algèbres

Soit un ensemble d'équations E.
On a la catégorie des F-E-algèbres, dont les objets sont les F-E-algèbres, et les flèches les homomorphismes de F-algèbre.
La catégorie des F-E-algèbres est une sous-catégorie de la catégorie des F-algèbres.

4.2.4 Représentation de plusieurs équations en une seule

On suppose ici que la catégorie **C** admet des sommes. On considère un ensemble d'équations E. On montre que cet ensemble d'équations est équivalent à une équation unique.

On va montrer que l'on peut se ramener d'un ensemble à deux équations à un ensemble à une seule équation:

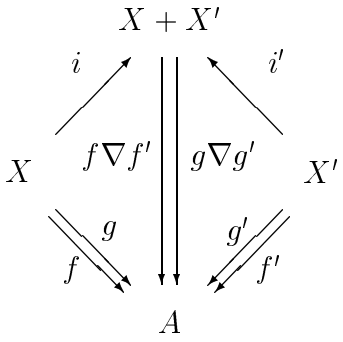
Soit deux équations: $X \underset{g}{\overset{f}{\rightrightarrows}} A$ et $X' \underset{g'}{\overset{f'}{\rightrightarrows}} A$.

Soit (M, m) une F-algèbre.
Soit $h_M : A \to M$ l'homomorphisme initial de (A, a) dans (M, m).

(M, m) satisfait les équations $(f \approx g)$ et $(f' \approx g')$ si et seulement si:

$$\begin{cases} h_M \circ f &= h_M \circ g \\ h_M \circ f' &= h_M \circ g' \end{cases}$$

On considère les flèches $f \nabla f' : X + X' \to A$ et $g \nabla g' : X + X' \to A$.



$$\begin{cases} h_M \circ f &= h_M \circ g \\ h_M \circ f' &= h_M \circ g' \end{cases}$$

$$\Leftrightarrow \begin{cases} h_M \circ (f \nabla f') \circ i &= h_M \circ (g \nabla g') \circ i \\ h_M \circ (f \nabla f') \circ i' &= h_M \circ (g \nabla g') \circ i' \end{cases}$$

$$\Leftrightarrow \quad h_M \circ (f \nabla f') \quad = \quad h_M \circ (g \nabla g')$$

$\Leftrightarrow (M, m)$ satisfait l'équation: $X + X' \underset{g \nabla g'}{\overset{f \nabla f'}{\rightrightarrows}} A$

Par la suite, on considérera toujours une seule équation, sachant que en pratique, plusieurs équations peuvent se ramener à une seule.

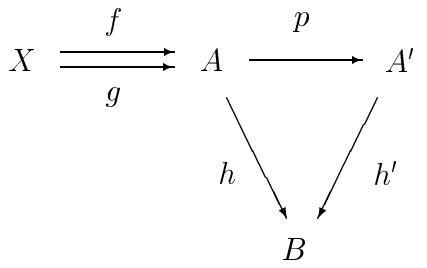
4.3 Coégalisateur

Nous aurons besoin, dans le paragraphe suivant, de la construction catégorielle de coégalisateur.
Soit une catégorie **C**.
Soit deux objets X et A, et deux flèches $f, g : X \to A$.

Un coégalisateur du diagramme $X \underset{g}{\overset{f}{\rightrightarrows}} A$ est un objet A' et une flèche $p : A \to A'$ qui satisfont les propriétés suivantes:

1. $p \circ f = p \circ g$

2. Pour tout objet B et toute flèche $h : A \to B$ telle que $h \circ f = h \circ g$, il existe une unique flèche $h' : A' \to B$ telle que:
   $h' \circ p = h$.



Remarque 1:

Le coégalisateur de $X \underset{g}{\overset{f}{\rightrightarrows}} A$ est tout simplement la colimite de ce diagramme.

Remarque 2:

Dans la catégorie des ensembles **SET**, la construction d'un coégalisateur correspond à un ensemble quotient. Soit $\equiv$ la relation d'équivalence engendrée par $\{(f(x), g(x)),\ x \in X\}$. A' correspond à l'ensemble quotient $A/\equiv$, et p correspond à la *projection* de A sur son quotient $A/\equiv$.

4.4 Algèbre initiale dans la catégorie des F-E-algèbres

4.4.1 Hypothèses

Dans ce paragraphe, on fait les hypothèses suivantes:

1. On se place dans une catégorie **C** ayant un objet initial L_0.
2. On suppose que **C** admet des sommes et des coégalisateurs.
3. On considère un endofoncteur F de **C** qui conserve les ω-colimites, les sommes et les coégalisateurs.
4. On suppose que tout diagramme $D_0 \overset{i_0}{\to} D_1 \overset{i_1}{\to} D_2 \overset{i_2}{\to} D_3 \cdots$, où $D_{i+1} = D_i + E_i$, et $i_1 : D_i \to D_{i+1}$ est l'injection de D_i dans D_{i+1} admet une ω-colimite.

4.4.2 Motivation

Soit (A, a) la F-algèbre initiale dans la catégorie $(F : \mathbf{C})$. On considère une équation

$$E = \{ \quad X \underset{g_0}{\overset{f_0}{\rightrightarrows}} A \quad \}$$

Le but de ce chapitre est de définir une F-E-algèbre (A^*, a^*), qui soit initiale dans la catégorie des F-E-algèbres.

(A^*, a^*) doit donc vérifier:
Pour toute F-E-algèbre (M, m), il existe un unique $\lambda : A^* \to M$ tel que: $\lambda \circ a^* = m \circ F(\lambda)$.

4.4.3 Première idée

La première idée qui vient à l'esprit est de construire un coégalisateur de $X \underset{g_0}{\overset{f_0}{\rightrightarrows}} A$,

défini par un objet A' et une flèche $p : A \to A'$:

$$X \underset{g_0}{\overset{f_0}{\rightrightarrows}} A \xrightarrow{p} A'$$

On prend l'image du diagramme par F:

$$F(X) \underset{F(g_0)}{\overset{F(f_0)}{\rightrightarrows}} F(A) \xrightarrow{F(p)} F(A')$$

Comme F conserve les coégalisateurs, on obtient un coégalisateur du diagramme

$$F(X) \underset{F(g_0)}{\overset{F(f_0)}{\rightrightarrows}} F(A).$$

Considérons le schéma:

$$\begin{array}{ccccc} X & \underset{g_0}{\overset{f_0}{\rightrightarrows}} & A & \xrightarrow{p} & A' \\ & & \uparrow a & & \uparrow a' \\ F(X) & \underset{F(g_0)}{\overset{F(f_0)}{\rightrightarrows}} & F(A) & \xrightarrow{F(p)} & F(A') \end{array}$$

Pour que la flèche $a' : F(A') \to A'$ existe, il faudrait que: $p \circ a \circ F(f) = p \circ a \circ F(g)$. Comme il n'y a aucune raison pour que ce soit le cas, cette définition d'algèbre échoue.

Examinons sur un exemple les raisons de cet échec: un coégalisateur correspond, dans la catégorie **SET**, à un quotient par une relation d'équivalence.

Dans l'approche algébrique des types abstraits, on prend le quotient par la *congruence* engendrée par les équations, et non le quotient par la *relation d'équivalence*.
Considérons par exemple la signature:

```
0: -> nat
s: nat -> nat
```

et l'équation:

```
s(s(0)) = 0
```

Considérer la relation d'équivalence engendrée par cette équation, au lieu de la congruence, consiste à "oublier" toutes les équations:

```
s(s(s(0))) = s(0)
s(s(s(s(0)))) = s(s(0))
...
```

On est donc amené à "simuler" cette fermeture par congruence dans le cadre des catégories. Pour cela, on va "ajouter" toutes les équations qui se déduisent de $X \underset{g}{\overset{f}{\rightrightarrows}} A$ par congruence.

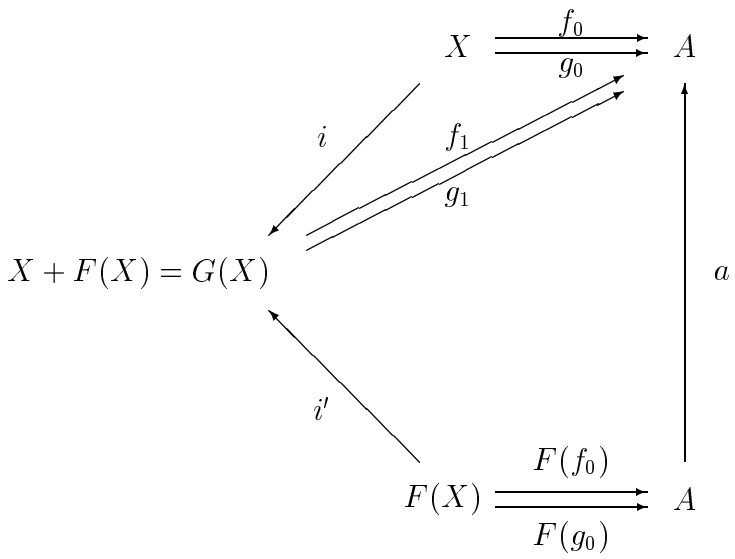
4.4.4 Seconde idée

On considère l'endofoncteur G sur $\mathbf{C}$ défini par:
$G(W) = W + F(W)$
$G(w) = w + F(w)$

Soit $f_1, g_1 : G(X) \to A$ définis par:

$$f_1 = f_0 \nabla (a \circ F(f_0))$$

$$g_1 = g_0 \nabla (a \circ F(g_0))$$



On a vu en 4.2.4 que considérer l'équation $(f_1 \approx g_1)$ revient à considérer la conjonction des deux équations $(f_0 \approx g_0)$ et $(a \circ F(f_0) \approx a \circ F(g_0))$.

Examinons la signification de l'équation $(a \circ F(f_0) \approx a \circ F(g_0))$ dans l'exemple des entiers modulo 2:

On considère toujours le même foncteur F défini par:
$F(W) = 1 + W$
$F(w) = id_1 + w$
Soit (A, a) l'algèbre initiale. On pose $z = a \circ i$, et $s = a \circ i'$. On considère l'équation:

$1 \underset{g_0}{\overset{f_0}{\rightrightarrows}} A$, avec $f_0 = s \circ s \circ z$, et $g_0 = z$.

Soit (M, m) une F-algèbre.
(M, m) satisfait l'équation $(a \circ F(f_0) \approx a \circ F(g_0))$
$\Leftrightarrow$ (M, m) satisfait les deux équations:

$$\begin{cases} a \circ F(f_0) \circ i & \approx & a \circ F(g_0) \circ i \\ a \circ F(f_0) \circ i' & \approx & a \circ F(g_0) \circ i' \end{cases}$$

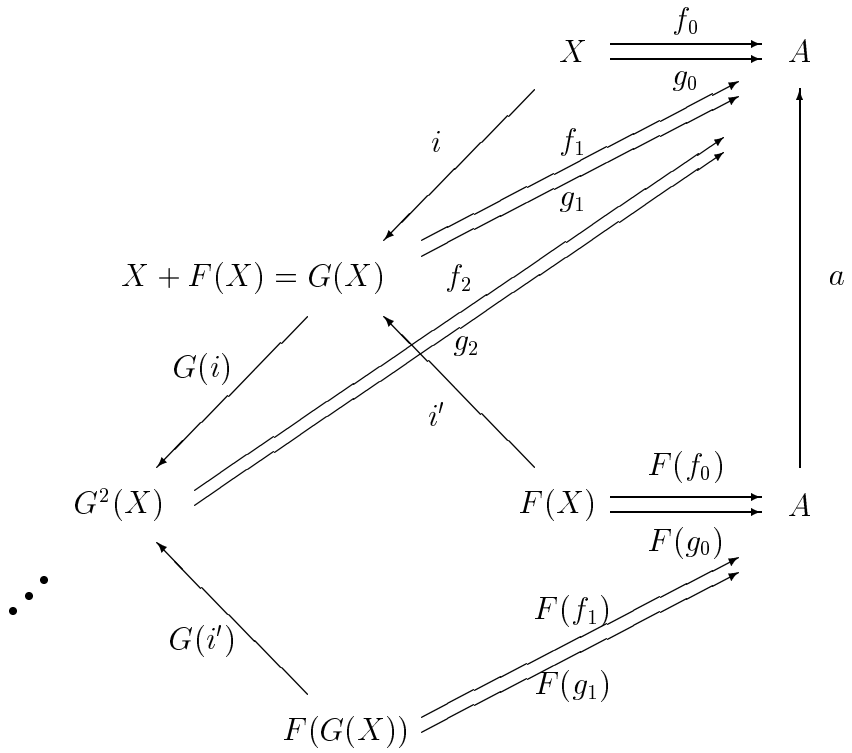
Or, on a:
$a \circ F(f_0) \circ i = z$
$a \circ F(f_0) \circ i' = a \circ i' \circ f_0 = s \circ f_0 = s \circ s \circ s \circ z$
$a \circ F(g_0) \circ i = z$
$a \circ F(g_0) \circ i' = a \circ i' \circ g_0 = s \circ g_0 = s \circ z$

Par conséquent:
(M, m) satisfait l'équation $(a \circ F(f_0) \approx a \circ F(g_0))$
$\Leftrightarrow$ (M, m) satisfait les deux équations:
$\begin{cases} z \approx z \\ s \circ s \circ s \circ z \approx s \circ z \end{cases}$

La première de ces deux équations ne pose aucune contrainte. On remarque que la seconde équation correspond à un "pas" de fermeture par congruence. Pour obtenir toutes les équations

$$(s^2 \circ z \approx z),\ (s^3 \circ z \approx s \circ z),\ \cdots\ (s^{n+2} \circ z \approx s^n \circ z),\ \cdots$$

il va falloir itérer cette opération.
Pour obtenir une équation qui modélise correctement une congruence, on est donc amené à poser:
$f_1 = f_0 \nabla (a \circ F(f_0))$
$g_1 = g_0 \nabla (a \circ F(g_0))$
Puis:
$f_2 = f_1 \nabla (a \circ F(f_1))$
$g_2 = g_1 \nabla (a \circ F(g_1))$
$\cdots$



Il faudra prendre la "limite" de cette suite de fonctions. Pour cela, on va considérer le cocône colimite du diagramme:

$$X \xrightarrow{i} G(X) \xrightarrow{G(i)} G^2(X) \cdots$$

Soit $(Z, \{\xi_k : G^k(X) \to Z\})$ ce cocône. La "limite" des fonctions f_i et g_i va correspondre à deux flèches $f, g : Z \to A$. On prendra enfin le coégalisateur de $Z \underset{g}{\overset{f}{\rightrightarrows}} A$.

4.4.5 Définition de (A^*, a^*)

Dans cette partie, on réalise la construction formelle des idées décrites dans le paragraphe précédent.

Soit une équation $X \underset{g_0}{\overset{f_0}{\rightrightarrows}} A$.

On se place dans les hypothèses décrites en 4.4.1. On considère l'endofoncteur G sur $\mathbf{C}$ défini par:
$G(W) = W + F(W)$
$G(w) = w + F(w)$

Lemme 1:

Le foncteur G conserve les sommes et les ω-colimites.

Preuve:

F conserve les sommes et les ω-colimites. Le bifoncteur $+$ conserve les colimites, donc en particulier les sommes et les ω-colimites. Donc G conserve les sommes et les ω-colimites. □

G conserve les sommes: détaillons ce que cela veut dire:
Soit deux objets B et B'. On considère la somme de B et B':

$$B \xrightarrow{i} B + B' \xleftarrow{i'} B'$$

G conserve les sommes, donc

$$G(B) \xrightarrow{G(i)} G(B + B') \xleftarrow{G(i')} G(B')$$

est la somme de $G(B)$ et $G(B')$, c'est-à-dire:
Pour tout objet C, pour toute flèche $f : G(B) \to C$ et $f' : G(B') \to C$, il existe une unique flèche $f \nabla f' : G(B + B') \to C$ telle que:

$(f \nabla f') \circ G(i) = f$
$(f \nabla f') \circ G(i') = f'$

Soit i_0 et i'_0 les injections de la somme de X et $F(X)$:

$$X \xrightarrow{i_0} G(X) = X + F(X) \xleftarrow{i'_0} F(X)$$

G conserve les sommes, donc pour tout entier k:

$$G^k(X) \xrightarrow{G^k(i_0)} G^{k+1}(X) = G^k(X) + G^k(F(X)) \xleftarrow{G^k(i'_0)} G^k(F(X))$$

est la somme de $G^k(X)$ et $G^k(F(X))$.
Posons $i_k = G^k(i_0)$, et $i'_k = G^k(i'_0)$.
Pour tout objet C, et pour toute flèche $f : G^k(X) \to C$ et $f' : G^k(F(X)) \to C$, il existe une unique flèche $f \nabla f' : G^{k+1}(X) \to C$ telle que:
$(f \nabla f') \circ i_k = f$
$(f \nabla f') \circ i'_k = f'$

Lemme 2:

Le diagramme $X \xrightarrow{i_0} G(X) \xrightarrow{i_1} G^2(X) \xrightarrow{i_2} \cdots$ a une colimite.

Preuve:

Pour tout entier k, la flèche i_k est une injection de $G^k(X)$ dans $G^k(X) + G^k(F(X))$, donc, d'après l'hypothèse 4 de 4.4.1, le diagramme $X \xrightarrow{i_0} G(X) \xrightarrow{i_1} G^2(X) \xrightarrow{i_2} \cdots$ a une colimite.
□

Soit $(Z, \{\xi_k : G^k(X) \to Z,\ k \in I\!N\})$ le cocône colimite.

G conserve les ω-colimites donc le diagramme

$$G(X) \xrightarrow{i_1} G^2(X) \xrightarrow{i_2} G^3(X) \xrightarrow{i_3} \cdots$$

a pour cocône colimite: $(G(Z), \{G(\xi_k : G^{k+1}(X) \to G(Z),\ k \in I\!N\})$.

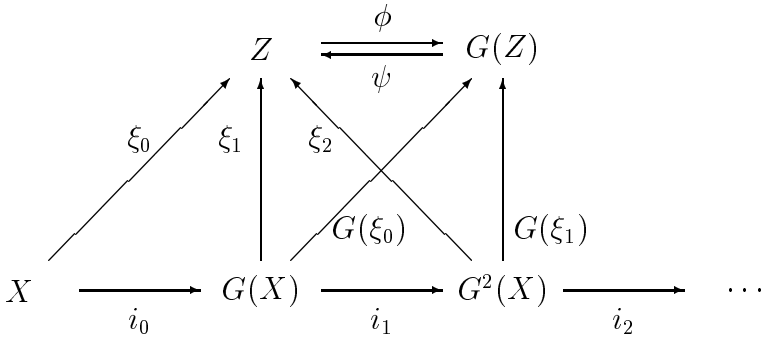
On pose:
$\xi'_0 = G(\xi_0) \circ i_0$
$\forall k > 0 : \xi'_k = G(\xi_{k-1})$.

- $(G(Z), \{\xi'_k : G^k(X) \to G(Z)\})$ est un cocône sur le diagramme $X \xrightarrow{i_0} G(X) \xrightarrow{i_1} G^2(X) \xrightarrow{i_2} \cdots$, donc il existe un unique $\phi : Z \to G(Z)$ tel que: $\forall k \in I\!N : \phi \circ \xi_k = \xi'_k$

- $(Z, \{\xi_k : G^k(X) \to G(Z),\ k > 0\})$ est un cocône sur le diagramme $G(X) \xrightarrow{i_1} G^2(X) \xrightarrow{i_2} G^3(X) \xrightarrow{i_3} \cdots$ donc il existe un unique $\psi : G(Z) \to Z$ tel que: $\forall k > 0 : \psi \circ \xi'_k = \xi_k$.
  De plus, $\psi \circ \xi'_0 = \psi \circ G(\xi_0) \circ i_0 = \psi \circ \xi'_1 \circ i_0 = \psi \circ \xi_0$.

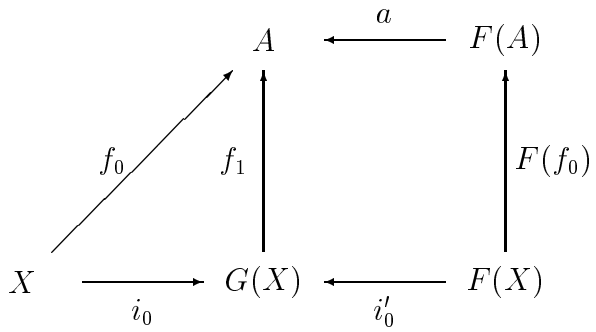
- $\forall i \in I\!N : \psi \circ \phi \circ \xi_i = \psi \circ \xi'_i = \xi_i$
  donc $\psi \circ \phi = id_Z$.
  (Car $(Z, \{\xi_k : G^k(X) \to G(Z),\ k > 0\})$ est le cocône colimite du diagramme $X \xrightarrow{i_0} G(X) \xrightarrow{i_1} G^2(X) \xrightarrow{i_2} \cdots$.)

- $\forall i > 0 : \phi \circ \psi \circ \xi'_i = \phi \circ \xi_i = \xi'_i$ donc $\phi \circ \psi = id_{G(Z)}$.
  (Car $(G(Z), \{\xi'_k : G^k(X) \to G(Z)\})$ est le cocône colimite du diagramme $G(X) \xrightarrow{i_1} G^2(X) \xrightarrow{i_2} G^3(X) \xrightarrow{i_3} \cdots$.)



Rappelons que l'on a l'équation $X \underset{g_0}{\overset{f_0}{\rightrightarrows}} A$.

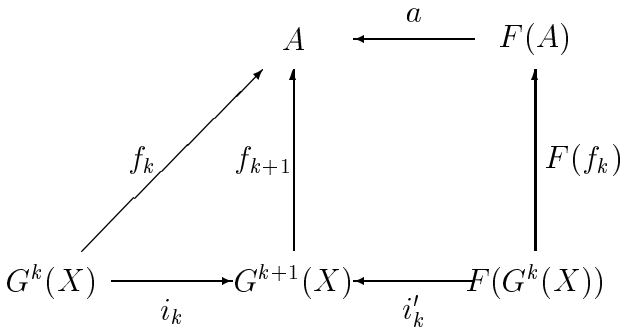
On pose:
$f_1 = f_0 \nabla (a \circ F(f_0))$
$g_1 = g_0 \nabla (a \circ F(g_0))$
f_1 est l'unique flèche telle que: $f_1 \circ i_0 = f_0$ et $f_1 \circ i'_0 = a \circ F(f_0)$.
g_1 est l'unique flèche telle que: $g_1 \circ i_0 = g_0$ et $g_1 \circ i'_0 = a \circ F(g_0)$.



On suppose par récurrence qu'on a construit:
$f_k : G^k(X) \to A$, et $g_k : G^k(X) \to A$.
f_k est l'unique flèche telle que: $f_k \circ i_{k-1} = f_{k-1}$ et $f_k \circ i'_{k-1} = a \circ F(f_{k-1})$.
g_k est l'unique flèche telle que: $g_k \circ i_{k-1} = g_{k-1}$ et $g_k \circ i'_{k-1} = a \circ F(g_{k-1})$.

$$G^k(X) \xrightarrow{i_k} G^{k+1}(X) = G^k(X) + G^k(F(X)) \xleftarrow{i'_k} G^k(F(X))$$

est la somme de $G^k(X)$ et $G^k(F(X))$, donc on peut poser:
$f_{k+1} = f_k \nabla (a \circ F(f_k))$
$g_{k+1} = g_k \nabla (a \circ F(g_k))$
f_{k+1} est l'unique flèche telle que: $f_{k+1} \circ i_k = f_k$ et $f_{k+1} \circ i'_k = a \circ F(f_k)$.
g_{k+1} est l'unique flèche telle que: $g_{k+1} \circ i_k = g_k$ et $g_{k+1} \circ i'_k = a \circ F(g_k)$.



Lemme 3:

Il existe une unique flèche $f : Z \to A$ telle que:
$\forall k \in I\!N : f \circ \xi_k = f_k$.
Il existe une unique flèche $g : Z \to A$ telle que:
$\forall k \in I\!N : g \circ \xi_k = g_k$.

Preuve:

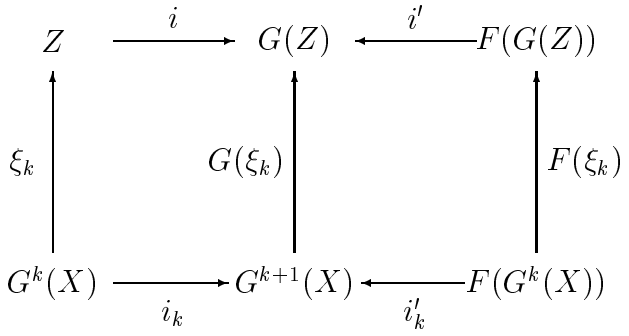
$(A, \{f_k : G^k(X) \to A,\ k \in I\!N\})$ et $(A, \{g_k : G^k(X) \to A,\ k \in I\!N\})$ sont des cocônes sur le diagramme: $X \xrightarrow{i_0} G(X) \xrightarrow{i_1} G^2(X) \xrightarrow{i_2} \cdots$.
En effet, les flèches f_{k+1} et g_{k+1} ont justement été construites pour que:
$f_{k+1} \circ i_k = f_k$ et $g_{k+1} \circ i_k = g_k$.
Par conséquent, comme $(Z, \{\xi_k : G^k(X) \to Z,\ k \in I\!N\})$ est le cocône colimite du diagramme $X \xrightarrow{i_0} G(X) \xrightarrow{i_1} G^2(X) \xrightarrow{i_2} \cdots$:
Il existe une unique flèche $f : Z \to A$ telle que:
$\forall k \in I\!N : f \circ \xi_k = f_k$.
Il existe une unique flèche $g : Z \to A$ telle que:
$\forall k \in I\!N : g \circ \xi_k = g_k$.

Soit $i : Z \to G(Z)$ et $i' : F(Z) \to G(Z)$ les deux injections liées à la somme de Z et $F(Z)$.

Lemme 4:

$G(\xi_k) : G^{k+1}(Z) \to G(Z)$ est l'unique flèche telle que:
$G(\xi_k) \circ i_k = i \circ \xi_k$
$G(\xi_k) \circ i'_k = i' \circ F(\xi_k)$.

Preuve:



Par définition du foncteur G, on a: $G(\xi_k) = \xi_k + F(\xi_k)$.
Par conséquent, $G(\xi_k) : G^{k+1}(Z) \to G(Z)$ est l'unique flèche telle que:
$G(\xi_k) \circ i_k = i \circ \xi_k$
$G(\xi_k) \circ i'_k = i' \circ F(\xi_k)$.

Lemme 5:

$i = \phi$.

Preuve:

Comme $(Z, \{\xi_k : G^k(X) \to G(Z),\ k > 0\})$ est le cocône colimite du diagramme $X \xrightarrow{i_0} G(X) \xrightarrow{i_1} G^2(X) \xrightarrow{i_2} \cdots$, il suffit de montrer que:
$\forall k \in I\!N : i \circ \xi_k = \phi \circ \xi_k$.

$\forall k :$ $i \circ \xi_k = G(\xi_k) \circ i_k$ (Lemme 4)
$\phi \circ \xi_k = \xi'_k = G(\xi_{k-1}) = G(\xi_k) \circ i_k$ (Définition de ϕ)

Lemme 6:

$f \circ \psi \circ i' = a \circ F(f)$
$g \circ \psi \circ i' = a \circ F(g)$.

Preuve:

Soit $t : G(Z) \to A$ l'unique flèche telle que:
$t \circ i = f$
$t \circ i' = a \circ F(f)$
(C'est-à-dire: $t = f \nabla (a \circ f)$)

$$\begin{array}{lll} t \circ i = f & \Rightarrow & t \circ i \circ \psi \circ i' = f \circ \psi \circ i' \\ & \Rightarrow & t \circ \phi \circ \psi \circ i' = f \circ \psi \circ i' \quad \text{(Lemme 5: } i = \phi\text{)} \\ & \Rightarrow & t \circ i' = f \circ \psi \circ i' \quad \text{(Car } \phi \circ i = id_{G(Z)}\text{)} \\ & \Rightarrow & a \circ F(f) = f \circ \psi \circ i' \quad \text{(Définition de } t\text{)} \end{array}$$

La preuve de $g \circ \psi \circ i' = a \circ F(g)$ est exactement identique.

Considérons le diagramme $Z \underset{g}{\overset{f}{\rightrightarrows}} A$. Prenons en le coégalisateur, défini par l'objet A^* et la flèche $p : A \to A^*$

$$Z \underset{g}{\overset{f}{\rightrightarrows}} A \xrightarrow{p} A^*$$

Comme F conserve les coégalisateurs,

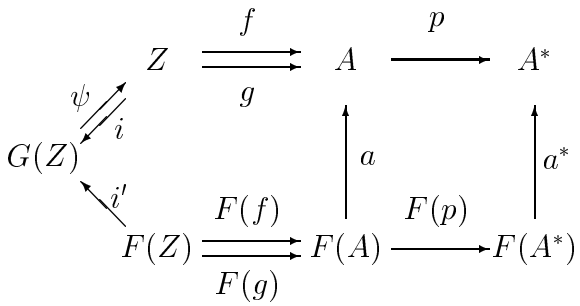
$$F(Z) \underset{F(g)}{\overset{F(f)}{\rightrightarrows}} F(A) \xrightarrow{F(p)} F(A^*)$$

est un coégalisateur.

$$\begin{array}{lll} p \circ a \circ F(f) & = p \circ f \circ \psi \circ i' & (\text{car } a \circ F(f) = f \circ \psi \circ i') \\ & = p \circ g \circ \psi \circ i' & (\text{car } p \circ f = p \circ g) \\ & = p \circ a \circ F(g) & (\text{car } g \circ \psi \circ i' = a \circ F(g)) \end{array}$$

Donc il existe un unique $a^* : F(A^*) \to A^*$ tel que:

$$p \circ a = a^* \circ F(p)$$



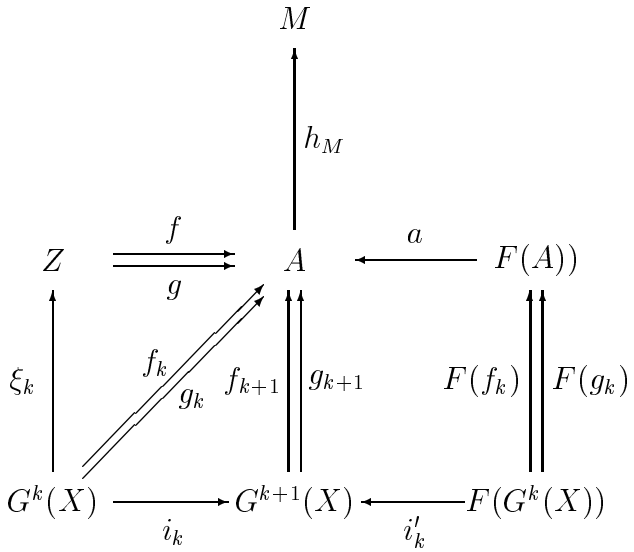
Lemme 7:

Soit une algèbre (M, m).
(1) (M, m) satisfait l'équation $(f_0 \approx g_0)$ $\Leftrightarrow$
(2) (M, m) satisfait les équations $\forall k \in I\!N : (f_k \approx g_k)$ $\Leftrightarrow$
(3) (M, m) satisfait l'équation $(f \approx g)$.

Preuve:

Soit h_M l'unique homomorphisme de (A, a) dans (M, m).
On montre que (1) $\Rightarrow$ (2) $\Rightarrow$ (3).

- (1) $\Rightarrow$ (2)
  Il faut montrer que: $h_M \circ f_0 = h_M \circ g_0 \Rightarrow \forall k \in I\!N : h_M \circ f_k = h_M \circ g_k$
  Par récurrence sur k:
  Pour $k = 0$, c'est évident.
  Supposons que $h_M \circ f_k = h_M \circ g_k$
  Montrons que $h_M \circ f_{k+1} = h_M \circ g_{k+1}$



  Il suffit de montrer que:
  $h_M \circ f_{k+1} \circ i_k = h_M \circ g_{k+1} \circ i_k$ et
  $h_M \circ f_{k+1} \circ i'_k = h_M \circ g_{k+1} \circ i'_k$.

$$\begin{array}{lll} h_M \circ f_{k+1} \circ i_k & = h_M \circ f_k & \text{(définition de } f_{k+1}\text{)} \\ & = h_M \circ g_k & \text{(hypothèse de récurrence)} \\ & = h_M \circ g_{k+1} \circ i_k & \text{(définition de } g_{k+1}\text{)} \end{array}$$

$$\begin{array}{lll} h_M \circ f_{k+1} \circ i'_k & = h_M \circ a \circ F(f_k) & \text{(définition de } f_{k+1}\text{)} \\ & = m \circ F(h_M) \circ F(f_k) & (h_M \text{ homomorphisme}) \\ & = m \circ F(h_M \circ f_k) & (F \text{ foncteur}) \\ & = m \circ F(h_M \circ g_k) & \text{(hypothèse de récurrence)} \\ & = m \circ F(h_M) \circ F(g_k) & (F \text{ foncteur}) \\ & = h_M \circ a \circ F(g_k) & (h_M \text{ homomorphisme}) \\ & = h_M \circ g_{k+1} \circ i'_k & \text{(définition de } g_{k+1}\text{)} \end{array}$$

- (2) $\Rightarrow$ (3)
  On montre que:
  $$(\forall k \in I\!N : \ h_M \circ f_k = h_M \circ g_k) \ \Rightarrow \ h_M \circ f = h_M \circ g$$
  Comme $(Z, \{\xi_k : G^k(X) \to Z, \ k \in I\!N\})$ est un cocône colimite, il suffit de montrer que:
  $$\forall k \in I\!N : \ h_M \circ f \circ \xi_k = h_M \circ g \circ \xi_k$$
  Cela est évident, puisque:
  $\forall k \in I\!N : \quad f \circ \xi_k = f_k$
  $\qquad\qquad\quad g \circ \xi_k = g_k$

- (3) $\Rightarrow$ (1)
  Supposons que: $h_M \circ f = h_M \circ g$.
  On a donc: $h_M \circ f \circ \xi_0 = h_M \circ g \circ \xi_0$
  d'où: $h_M \circ f_0 = h_M \circ g_0$.

Lemme 8:

Soit (M, m) une algèbre satisfaisant l'équation $(f_0 \approx g_0)$. Alors il existe un unique homomorphisme λ de (A^*, a^*) dans (M, m).

Preuve:

Il faut montrer qu'il existe un unique $\lambda : A^* \to M$ tel que:

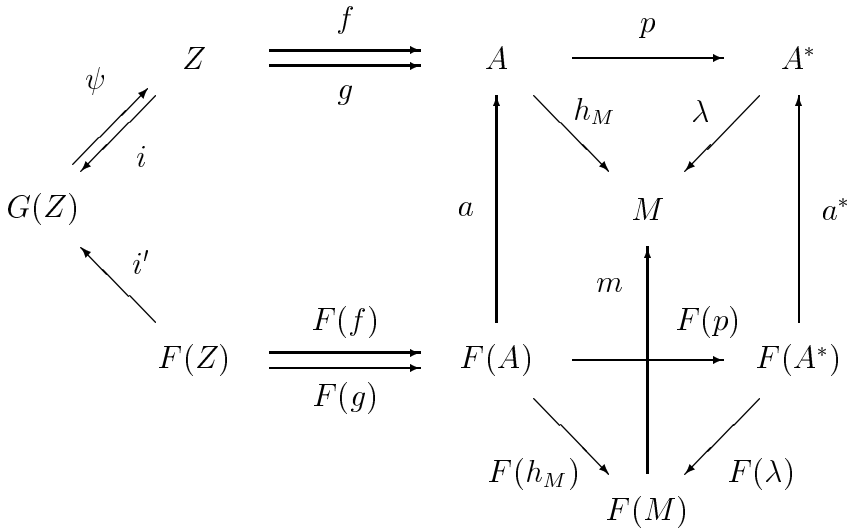
$$\lambda \circ a^* = m \circ F(\lambda)$$

Existence:

D'après le lemme 7, $h_M \circ f = h_M \circ g$. Donc, comme $Z \underset{g}{\overset{f}{\rightrightarrows}} A \xrightarrow{p} A^*$ est un coégalisateur, il existe un unique $\lambda : A^* \to M$ tel que: $\lambda \circ p = h_M$.

Montrons que λ est un homomorphisme, c'est-à-dire que

$$\lambda \circ a^* = m \circ F(\lambda)$$



Comme $F(Z) \underset{F(g)}{\overset{F(f)}{\rightrightarrows}} F(A) \xrightarrow{F(p)} F(A^*)$ est un coégalisateur, pour tout $\gamma : F(A) \to M$ tel que $\gamma \circ F(f) = \gamma \circ F(g)$, il existe un unique $\gamma' : F(A^*) \to M$ tel que $\gamma' \circ F(p) = \gamma$.

Nous allons prendre pour γ la flèche: $m \circ F(h_M) : F(A^*) \to M$.

$$\begin{aligned} h_M \circ f = h_M \circ g &\Rightarrow F(h_M \circ f) = F(h_M \circ g) \\ &\Rightarrow F(h_M) \circ F(f) = F(h_M) \circ F(g) \qquad (F \text{ foncteur}) \\ &\Rightarrow m \circ F(h_M) \circ F(f) = m \circ F(f_M) \circ F(g) \end{aligned}$$

Montrons que:

$$\lambda \circ a^* \circ F(p) = m \circ F(h_M)$$
$$m \circ F(\lambda) \circ F(p) = m \circ F(h_M)$$

Comme il existe un unique $\gamma' : F(A^*) \to M$ tel que $\gamma' \circ F(p) = m \circ F(h_M)$, on aura donc:

$$\lambda \circ a^* = m \circ F(\lambda)$$

$$\begin{aligned} \lambda \circ a^* \circ F(p) &= \lambda \circ p \circ a && (\text{car } p \circ a = a^* \circ F(p)) \\ &= h_M \circ a && (\text{définition de } \lambda) \\ &= m \circ F(h_M) && (h_M \text{ homomorphisme}) \end{aligned}$$

$$\begin{aligned} m \circ F(\lambda) \circ F(p) &= m \circ F(\lambda \circ p) && (F \text{ foncteur}) \\ &= m \circ F(h_M) && (\text{définition de } \lambda) \end{aligned}$$

Unicité:

On montre que:

$$\lambda \circ a^* = m \circ F(\lambda) \;\Rightarrow\; \lambda \circ p = h_M$$

Comme il existe un unique λ tel que $\lambda \circ p = h_M$, λ est unique.

$$\begin{array}{lll} \lambda \circ a^* = m \circ F(\lambda) & \Rightarrow & \lambda \circ a^* \circ F(p) = m \circ F(\lambda) \circ F(p) \\ & \Rightarrow & \lambda \circ a^* \circ F(p) = m \circ F(\lambda \circ p) \qquad (F \text{ foncteur}) \\ & \Rightarrow & \lambda \circ p \circ a = m \circ F(\lambda \circ p) \qquad (\text{définition de } a^*) \end{array}$$

Comme (A, a) est initiale, il existe un unique $h_M : A \to M$ tel que $h_M \circ a = m \circ F(h_M)$. Par conséquent: $h_M = \lambda \circ p$.

4.5 Comparaison avec les esquisses

La théorie des esquisses ([$BW91$], [$DR91$]) permet également de spécifier des types abstraits non libres. Dans ce paragraphe, nous tentons de comparer les deux approches.

Dans la théorie des esquisses, on définit la signature

```
0: -> nat
s: nat -> nat
```

par un graphe **G**, qui contient deux objets 1 et N, et deux flèches $0 : 1 \to N$ et $s : N \to N$.

G: $\quad 1 \overset{0}{\to} N \overset{s}{\to} N$

Soit **S** la catégorie engendrée par **G**.
Une algèbre sur cette signature est une interprétation dans une catégorie "concrète", par exemple **SET**, donnée par un foncteur $I : \mathbf{S} \to \mathbf{SET}$.

La catégorie **S** décrit la syntaxe du type abstrait, et la catégorie **SET** sa sémantique.

On a une définition *inductive* des termes: ce sont des flèches de la catégorie **S**, de source 1 et de destination N:

$$0,\; s \circ 0,\; s \circ s \circ 0,\; \cdots$$

Notre approche est basée sur des définitions récursives entre domaines:
$A \cong 1 + A$. On sait que l'on a une algèbre initiale (A, a), avec $a : 1 + A \to A$. On peut construire des termes *a posteriori* dans cette algèbre initiale, en posant: $0 = a \circ i_1$ et $s = a \circ i_2$. Ici, les termes ne sont pas construits inductivement.

Dans la théorie des esquisses, une équation est une relation entre deux flèches. Par

exemple: $s \circ s \circ 0 \sim 0$.
On considère la congruence engendrée par cette relation, (c'est-à-dire la plus petite relation d'équivalence contenant cette relation, et compatible avec la composition). On considère la catégorie quotient $\mathbf{S}/\sim$. Une algèbre qui satisfait les équations est alors donnée par un foncteur $I' : \mathbf{S}/\sim \to \mathbf{SET}$.
Les équations que l'on écrit sont purement syntaxiques, ce sont des équations *logiques* (au sens de *logique équationnelle*).

Dans notre approche basée sur la construction de domaines, on adopte directement un point de vue sémantique, en se plaçant immédiatement dans une catégorie "concrète" (**SET**, **CPO** ...). On ne peut donc pas imposer d'égalité entre flèches comme dans la catégorie syntaxique **S**.
On contourne le problème en définissant une équation comme deux flèches $f, g : X \to A$. On dit ensuite qu'une algèbre (B, b) satisfait l'équation $(f \approx g)$ si $h_B \circ f = h_B \circ g$, où $h_B : A \to B$ est l'homomorphisme initial de (A, a) dans (B, b).
Les équations que l'on écrit ne sont donc pas des équations au sens logique, il s'agit seulement d'égalités sémantiques vérifiées a posteriori.

En résumé, la théorie des esquisses permet de distinguer syntaxe et sémantique, de construire des termes inductivement et d'écrire des équations d'un point de vue syntaxique. Notre approche décrit des domaines sémantiques de façon récursive, et les "équations" que nous considérons ne sont pas des équations au sens de la logique. En ce sens, on est plus proche de la programmation fonctionnelle que de la spécification algébrique.

4.6 Exemple

Avant de donner un exemple de F-E-algèbre, rappelons quelques propriétés des produits: pour tout objet A et A' on note $p_1 : A \times A' \to A$ et $p_2 : A \times A' \to A'$ les deux projections. On a les égalités:

$$p_1 \circ < f_1, f_2 > = \ f_1$$

$$p_2 \circ < f_1, f_2 > = \ f_2$$

$$< f_1, f_2 > \circ h \ = < f_1 \circ h, f_2 \circ h >$$

$$< f_1 \circ p_1, f_2 \circ p_2 > = \ f_1 \times f_2$$

$$(f_1 \times f_2) \circ < g_1, g_2 > = < f_1 \circ g_1, f_2 \circ g_2 >$$

On peut généraliser le produit binaire à un produit n-aire. Pour $n = 3$, on note les trois projections p_1, p_2, et p_3. On obtient les égalités:

$$p_i \circ < f_1, f_2, f_3 > = \ f_i, \quad i = 1, 2, 3$$

$$< f_1, f_2, f_3 > \circ h \ = < f_1 \circ h, f_2 \circ h, f_3 \circ h >$$

Venons en maintenant à notre exemple. On note N l'ensemble des entiers naturels. On cherche à définir le type abstrait des listes d'entiers naturels de la façon suivante: on a la liste vide, la liste à un élément, et on peut concaténer des listes. En spécification algébrique on considère la signature:

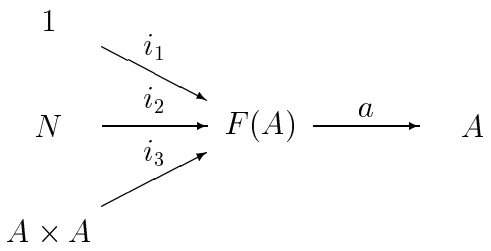
```
nil: -> liste
one: N -> liste
app: (liste,liste) -> liste
```

`nil` est la liste vide, `one(n)`, pour `n`∈`N` est la liste à un élément, et `app(l1,l2)` est la concaténation des listes `l1` et `l2`. On doit en plus spécifier que `nil` est un élément neutre pour la concaténation, et que la concaténation est associative. On ajoute donc les équations:

```
app(nil,x) = x                          (1)
app(x,nil) = x                          (2)
app(app(x,y),z) = app(x,app(y,z))       (3)
```

On va maintenant définir la même chose dans le cadre catégoriel. Pour cela, on définit le foncteur F:
Pour tout objet W: $F(W) = 1 + N + W \times W$
Pour toute flèche w: $F(w) = id_1 + id_N + w \times w$.

On appelle (A, a) l'algèbre initiale de la catégorie des F-algèbres.



Pour des questions de lisibilité, on pose:
$a \circ i_1 = Nil_A : 1 \to A$
$a \circ i_2 = One_A : N \to A$
$a \circ i_3 = App_A : A \times A \to A$

1 est l'objet terminal. Pour tout objet W, soit t_W l'unique flèche de W vers 1. La flèche t_W représente la fonction qui prend un argument de type W et rend l'unique élément de 1. Par exemple $Nil_A \circ t_A : A \to A$ est la fonction constante qui renvoie toujours la liste vide.

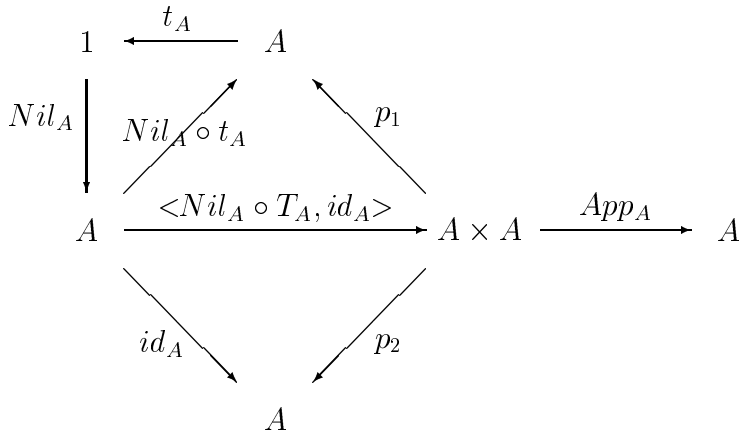
On écrit les trois équations sous forme fonctionnelle:

E : $App_A \circ < Nil_A \circ t_A, id_A > \approx\ id_A$ (1)

$App_A \circ < id_A, Nil_A \circ t_A > \approx\ id_A$ (2)

$App_A \circ < App_A \circ < p_1, p_2 >, p_3 > \approx\ App_A \circ < p_1, App_A \circ < p_2, p_3 >>$ (3)

On peut visualiser l'équation (1) par le diagramme:

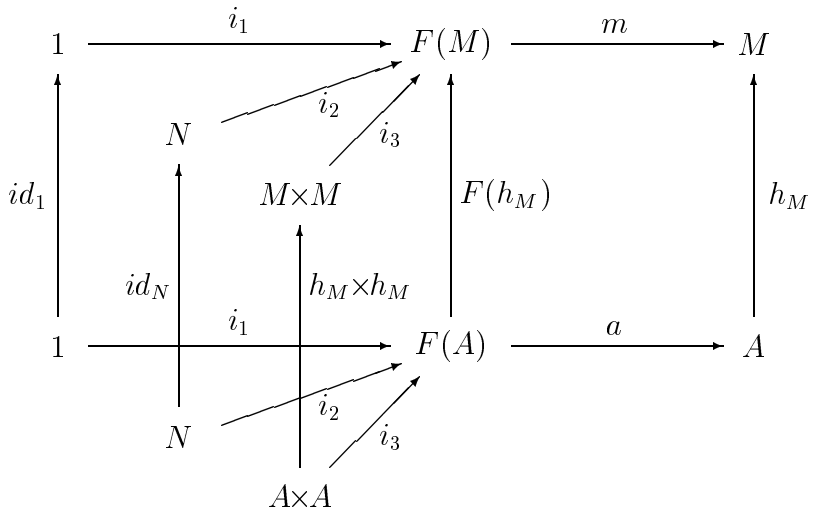


Soit (M, m) une F-algèbre. Posons, comme on l'a fait précédemment:
$m \circ i_1 = Nil_M : 1 \to M$
$m \circ i_2 = One_M : N \to M$
$m \circ i_3 = App_M : M \times M \to M$

Soit h_M l'unique homomorphisme de (A, a) dans (M, m).
h_M vérifie les égalités:

$$h_M \circ a = m \circ F(h_M)$$
$$F(h_M) \circ i_1 = i_1$$
$$F(h_M) \circ i_2 = i_2$$
$$F(h_M) \circ i_3 = i_3 \circ (h_M \times h_M)$$

Les trois dernières égalités proviennent des propriétés du foncteur somme. Tout cela est résumé sur le diagramme:



On en déduit les égalités suivantes:

$$\begin{array}{ll} h_M \circ Nil_A = Nil_M & (M1) \\ h_M \circ One_A = One_M & (M2) \\ h_M \circ App_A = App_M \circ (h_M \times h_M) & (M3) \end{array}$$

En effet:
$h_M \circ Nil_A = h_M \circ a \circ i_1 = m \circ F(h_M) \circ i_1 = m \circ i_1 = Nil_M$
$h_M \circ One_A = h_M \circ a \circ i_2 = m \circ F(h_M) \circ i_2 = m \circ i_2 = One_M$
$h_M \circ App_A = h_M \circ a \circ i_3 = m \circ F(h_M) \circ i_3 = m \circ i_3 \circ (h_M \times h_M) = App_M \circ (h_M \times h_M)$

Supposons que (M, m) satisfait les équations, c'est-à-dire: pour toute équation $(f \approx g)$, on a: $h_M \circ f = h_M \circ g$.

Si on suppose de plus que h_M est un homomorphisme surjectif ((M, m) algèbre engendrée), on a:

$$\begin{array}{ll} App_M \circ < Nil_M \circ t_M, id_M > = \ id_M & (1') \\ App_M \circ < id_M, Nil_M \circ t_M > = \ id_M & (2') \\ App_M \circ < App_M \circ < p_1, p_2 >, p_3 > = \ App_M \circ < p_1, App_M \circ < p_2, p_3 >> & (3') \end{array}$$

Montrons seulement l'égalité $(1')$:

	$h_M \circ App_A \circ < Nil_A \circ t_A, id_A > = \ h_M$	$((M,m)$ vérifie l'équation $(1))$
$\Rightarrow$	$App_M \circ (h_M \times h_M) \circ < Nil_A \circ t_A, id_A > = \ h_M$	$(M3)$
$\Rightarrow$	$App_M \circ < h_M \circ Nil_A \circ t_A, h_M \circ id_A > = \ h_M$	(Propriété du produit)
$\Rightarrow$	$App_M \circ < Nil_M \circ t_A, h_M > = \ h_M$	$(M1)$
$\Rightarrow$	$App_M \circ < Nil_M \circ t_M \circ h_M, id_M \circ h_M > = \ h_M$	$(t_A = t_M \circ h_M$ car la flèche $A \to 1$ est unique)
$\Rightarrow$	$App_M \circ < Nil_M \circ t_M, id_M > \circ h_M \ = \ h_M$	(Propriété du produit)
$\Rightarrow$	$App_M \circ < Nil_M \circ t_M, id_M > = \ id_M$	(h_M surjectif)

Remarquons que si les trois égalités $(1')$, $(2')$, et $(3')$ sont vérifiée, alors (M,m) satisfait les trois équations (même si h_M n'est pas surjectif).

On a vu dans ce chapitre qu'il y a une algèbre initiale dans la catégorie des F-algèbres qui satisfont les trois équations: (A^*, a^*). Cette algèbre modélise le type des listes d'entiers.

Définissons une autre algèbre (R,r) par:

- $R = A^*$
- $r : 1 + N + A^* \times A^* \to A^*$
  $r \circ i_1 = Nil_{A^*} = Nil_R$
  $r \circ i_2 = One_{A^*} = One_R$
  $r \circ i_3 = App_{A^*} \circ < p_2, p_1 >= App_R$

On montre que (R,r) satisfait les équations.
Il suffit de montrer que:

1. $App_R \circ < Nil_R \circ t_R, id_R > = \ id_R$
2. $App_R \circ < id_R, Nil_R \circ t_R > = \ id_R$
3. $App_R \circ < App_R \circ < p_1, p_2 >, p_3 > = \ App_R \circ < p_1, App_R \circ < p_2, p_3 >>$

1. $App_R \circ < Nil_R \circ t_R, id_R > =$
   $App_{A^*} \circ < p_2, p_1 > \circ < Nil_{A^*} \circ t_{A^*}, id_{A^*} > =$
   $App_{A^*} \circ < p_2 \circ < Nil_{A^*} \circ t_{A^*}, id_{A^*} >, p_1 \circ < Nil_{A^*} \circ t_{A^*}, id_{A^*} >> =$
   $App_{A^*} \circ < id_{A^*}, Nil_{A^*} \circ t_{A^*} > = \ id_{A^*}$

2. De même:
   $App_R \circ < id_R, Nil_R \circ t_R > =$
   $App_{A^*} \circ < Nil_{A^*} \circ t_{A^*}, id_{A^*} > = \ id_{A^*}$

3. Transformons la partie gauche:
   $App_R \circ < App_R \circ < p_1, p_2 >, p_3 > =$
   $App_{A^*} \circ < p_2, p_1 > \circ < App_R \circ < p_1, p_2 >, p_3 > =$
   $App_{A^*} \circ < p_3, App_R \circ < p_1, p_2 >> =$
   $App_{A^*} \circ < p_3, App_{A^*} \circ < p_2, p_1 > \circ < p_1, p_2 >> =$

$App_{A^*} \circ < p_3, App_{A^*} \circ < p_2, p_1 >>$

Transformons de même la partie droite:
$App_R \circ < p_1, App_R \circ < p_2, p_3 >> =$
$App_{A^*} \circ < p_2, p_1 > \circ < p_1, App_R \circ < p_2, p_3 >>$
$App_{A^*} \circ < App_R \circ < p_2, p_3 >, p_1 > =$
$App_{A^*} \circ < App_{A^*} \circ < p_2, p_1 > \circ < p_2, p_3 >, p_1 > =$
$App_{A^*} \circ < App_{A^*} \circ < p_3, p_2 >, p_1 >$

De plus, (A^*, a^*) satisfait les équations donc:
$App_{A^*} \circ < App_{A^*} \circ < p_1, p_2 >, p_3 > = App_{A^*} \circ < p_1, App_{A^*} \circ < p_2, p_3 >> \Rightarrow$
$App_{A^*} \circ < App_{A^*} \circ < p_1, p_2 >, p_3 > \circ < p_3, p_2, p_1 > =$
$\quad App_{A^*} \circ < p_1, App_{A^*} \circ < p_2, p_3 >> \circ < p_3, p_2, p_1 > \Rightarrow$
$App_{A^*} \circ < App_{A^*} \circ < p_1, p_2 > \circ < p_3, p_2, p_1 >, p_3 \circ < p_3, p_2, p_1 >> =$
$\quad App_{A^*} \circ < p_1 \circ < p_3, p_2, p_1 >, App_{A^*} \circ < p_2, p_3 > \circ < p_3, p_2, p_1 >> \Rightarrow$
$App_{A^*} \circ < App_{A^*} \circ < p_3, p_2 >, p_1 > = App_{A^*} \circ < p_3, App_{A^*} \circ < p_2, p_1 >>$

d'où le résultat:
$App_R \circ < App_R \circ < p_1, p_2 >, p_3 > = App_R \circ < p_1, App_R \circ < p_2, p_3 >>.$

Par conséquent, (R, r) est une F-E-algèbre. Comme (A^*, a^*) est initiale dans la catégorie des F-E-algèbres, il existe un unique homomorphisme h de (A^*, a^*) dans (R, r), c'est-à-dire tel que:

$$h \circ a^* = r \circ F(h)$$

Transformons un peu cette égalité:

$$h \circ a^* = r \circ F(h) \quad \Leftrightarrow$$

$$k = 1, 2, 3 : \quad h \circ a^* \circ i_k = r \circ F(h) \circ i_k \quad \Leftrightarrow$$

$$\begin{cases} h \circ Nil_{A^*} = Nil_R \ = Nil_{A^*} & (H1) \\ h \circ One_{A^*} = One_R \ = One_{A^*} & (H2) \\ h \circ App_{A^*} = App_R \circ (h \times h) & (H3) \end{cases}$$

Transformons la dernière égalité:

$$\begin{aligned} h \circ App_{A^*} &= App_R \circ (h \times h) \\ &= App_{A^*} \circ < p_2, p_1 > \circ (h \times h) \\ &= App_{A^*} \circ < p_2 \circ (h \times h), p_1 \circ (h \times h) > \\ &= App_{A^*} \circ < h \circ p_2, h \circ p_1 > \qquad (H3') \end{aligned}$$

C'est homomorphisme h est la fonction $reverse$, qui "retourne" une liste. En style fonctionnel, cette fonction se programme par:

```
reverse nil        = nil
reverse one(n)     = one(n)
reverse app(l1,l2) = app(reverse(l1),reverse(l2))
```

Cela correspond bien aux trois égalités que l'on a obtenues.

On peut montrer que l'on a bien:

$$h \circ h = id_{A^*}$$

Pour cela, il suffit de montrer que $h \circ h : A^* \to A^*$ est un homomorphisme. En effet, id_{A^*} est également un homomorphisme, et comme (A^*, a^*) est initiale, cet homomorphisme est unique. Il suffit donc de montrer que:

$$h \circ h \circ a^* = a^* \circ F(h \circ h)$$

Cette égalité est équivalente aux trois égalités suivantes:

1. $h \circ h \circ Nil_{A^*} = Nil_{A^*}$
2. $h \circ h \circ One_{A^*} = One_{A^*}$
3. $h \circ h \circ App_{A^*} = App_{A^*} \circ ((h \circ h) \times (h \circ h))$

1. $h \circ h \circ Nil_{A^*} = h \circ Nil_{A^*}$ (d'après l'égalité $H1$)
   $= Nil_{A^*}$ (d'après l'égalité $H1$)

2. $h \circ h \circ One_{A^*} = h \circ One_{A^*}$ (d'après l'égalité $H2$)
   $= One_{A^*}$ (d'après l'égalité $H2$)

3. $h \circ h \circ App_{A^*} = h \circ App_{A^*} \circ < h \circ p_2, h \circ p_1 >$ (égalité $H3'$)
   $= App_{A^*} \circ < h \circ p_2, h \circ p_1 > \circ < h \circ p_2, h \circ p_1 >$ (égalité $H3'$)

Calculons $< h \circ p_2, h \circ p_1 > \circ < h \circ p_2, h \circ p_1 >$:
$< h \circ p_2, h \circ p_1 > \circ < h \circ p_2, h \circ p_1 > =$
$< h \circ p_2 \circ < h \circ p_2, h \circ p_1 >, h \circ p_1 \circ < h \circ p_2, h \circ p_1 > =$
$< h \circ h \circ p_1, h \circ h \circ p_2 > =$
$(h \circ h) \times (h \circ h)$

Par conséquent:
$h \circ h \circ App_{A^*} = App_{A^*} \circ ((h \circ h) \times (h \circ h))$

$h \circ h$ est donc bien un homomorphisme, et par conséquent:

$$h \circ h = id_{A^*}$$

Conclusion

Il existe différentes approches pour construire des types abstraits: en particulier la spécification algébrique et la théorie des domaines, qui offrent des fondements théoriques pour l'élaboration et la preuve de programmes. Notre travail a consisté à étudier la spécification de types abstraits dans le cadre catégoriel de Plotkin, Smyth et Wand, en essayant d'appliquer certaines idées issues de la spécification algébrique. Nous avons notamment introduit la notion d'équation et d'algèbre satisfaisant un ensemble d'équations dans ce formalisme. Nous avons montré qu'il est possible de définir une algèbre initiale dans la catégorie des algèbres satisfaisant des équations, ce qui permet de donner une sémantique à des types abstraits non libres.

De nombreux points restent à étudier. D'abord nous n'avons pas montré que cette définition d'algèbre initiale est possible dans les catégories **SET** et **CPOS**. Dans **SET**, la construction ne devrait pas poser de problème, puisqu'un coégalisateur est simplement un quotient par une relation d'équivalence. Par contre, il faudrait montrer que la catégorie **CPOS** a des coégalisateurs. Cela doit être possible en construisant un quotient, puis en complétant en ajoutant toutes les bornes supérieures de chaines croissantes, ceci afin d'obtenir un CPO. Il faudrait aussi montrer que tous les foncteurs "utiles" conservent les sommes et les coégalisateurs.

D'autre part, nous n'avons donné pratiquement aucune application. Nous avons uniquement examiné des types abstraits très simples et des structures de contrôle triviales. Il reste maintenant à construire des exemples plus convaincants. On pourrait également étudier des types abstraits paramétrés, très utiles en programmation.

Enfin nous avons travaillé dans deux formalismes: spécification algébrique et théorie des catégories. Nous avons fait quelques comparaisons informelles entre ces deux démarches, et expliqué certaines idées de l'approche catégorielle dans le cadre algébrique. Il serait intéressant d'analyser de façon plus systématique le rapport entre ces deux approches.

Bibliographie

[*AL*91] A. Asperti et G. Longo, Categories, Types and Structures, An Introduction to Category Theory for the Working Computer Scientist, *Foundations of Computing Science, MIT Press*, 1991.

[*AM*75] M. A. Arbib et E.G. Manes, Arrows, Structures and Functors, *Academic Press*, 1975.

[*BP*91] D. Bert et M.-L. Potet, Construction and Transformation of Programs based on an Algebraic Approach of the Data Types, 1991.

[*BW*90] M. Barr et C. Wells, Category Theory for Computing Science, *Prentice Hall*, 1990.

[*DR*91] D. Duval et J.-C. Reynaud, Sketches and Computation, *Rapport de recherche, RR 871-I-Imag-123 LIFIA*, 1991.

[*EM*85] H. Ehrig et B. Mahr, Fundamentals of Algebraic Specification 1, *Springer-Verlag*, 1985.

[*GS*90] C. A. Gunter et D. S. Scott, Semantic domains, in *Handbook of theoretical computer science*, 1990, pages 635-674.

[*Hen*75] F. W. von Henke, On generating programs from datatypes: an approach to automatic programming, in *Proving and improving programs, Proc., Arc et Senans, Ed: INRIA*, pages 57-69, 1975.

[*Jeu*91] J. Jeuring, The derivation of hierarchies of algorithms on matrices, in *Constructing programs from specifications, Möller Ed., IFIP*, pages 9-22, 1991.

[*LS*77] D. J. Lehmann et M. B. Smyth, Data Types, in *Proc. 18th IEEE Symp. on Foundations of computer Science*, 1977.

[*Mal*89] G. Malcolm, Homomorphisms and Promotability, in *Proc. of Mathematics of Program Construction, LNCS 375*, page 335-347, 1989.

[*Mal*90] G. Malcolm, Data Structures and Program Transformation, in *Science of Computer Programming* 14 pages 255-279, 1990.

[*MFP*91] E. Meijer, M. Fokkinga et R. Paterson, Functional Programming with Bananas, Lenses, Envelopes and Barbed Wire, in *Proc. 5th ACM Conf. on Functional Programming Languages and Computer Architecture, LNCS 523*, 1991.

[*MG*85] J. Meseguer et J. A. Goguen, Initiality, induction and computability, in: M. Nivat, J. C. Reynolds, eds., *Algebraic methodes in Semantics* (Cambridge University Press), 1985.

[*ML*71] S. Mac Lane, Categories for the working mathematician, *Graduate Texts in Mathematics, Springer-Verlag*, 1971.

[*SP*77] M. B. Smyth et G.D. Plotkin, The category-theoretic solution of recursive domain equations, in *Proc. 18th IEEE Symp. on Foundations of computer Science*, pages 13-17, 1977.

[*Ten*91] R. D. Tennent, Semantics of Programming Languages, *Prentice Hall, International Series in Computer Science*, 1991.

[*Wan*79] M. Wand, Fixed-point construction in order-enriched categories, in *Theoretical Computer Science*, 8, pages 13-30, 1979.

[*Wec*92] W. Wechler, Universal Algebra for computer scientists, *Springer Verlag*, 1992.