Notes de cours de L3

Prises par Hugo Salou

29 janvier 2026

Table des matières

Préface.

Dans ce document, vous trouverez les notes de cours que j'ai pu prendre en LaTeX durant cette année de L3. Ce document rassemble donc les notes des cours suivants :

- *Théorie de la Programmation* (THPROG), Daniel Hirschkoff ;
- *Logique* (LOG), Natacha Portier ;
- *Projet fonctionnel* (PROFON), Daniel Hirschkoff.

Sur mon site (http://167.99.84.84/[1]), il y a plusieurs documents :

- les notes des cours ci-dessus en PDF individuel (et même en PDF « par chapitre ») ;
- les notes de cours d'*Algorithmique 1* (écrites avec *Typst*, donc pas inclue dans ce document) ;
- les notes de cours de *Mesure et intégration* (écrites à la main et scannées, donc pas inclue dans ce document) ;
- les TDs (Algorithmique 2, Projet fonctionnel, Probabilités, Logique, Algèbre 1, et Théorie des catégories) ;
- les DMs (Algorithmique 2, Probabilités, Logique, Algèbre 1, et Fondements de l'informatique).

Si vous voyez des erreurs dans les notes de cours (formatage, orthographe, grammaire, *etc*), prévenez-moi ! Ces notes sont disponibles sur GitLab et sur GitHub si vous voulez corriger directement. Sinon, prévenez-moi via Discord (`hugos29`) ou par email (y.hugo.s29@gmail.com).

1. Accès direct à la page web pour l'année « L3 ENS » : http://167.99.84.84/ens1/

Amusez-vous bien avec ce document de 267 pages !

Hugo Salou

Première partie

Théorie de la programmation.

Introduction.

Dans ce cours, on étudie la *sémantique des langages de programmation*. On présente des approches pour

- ▷ définir rigoureusement ce qu'est/ce que fait un programme ;
- ▷ établir mathématiquement des propriétés sur des programmes.

Par exemple,

- ▷ démontrer l'absence de *bug* dans un programme ;
- ▷ démontrer des propriétés sur des programmes de transformation de programme ;
- ▷ l'étude des nouveaux langages de programmation.

Dans ce cours, les langages fonctionnels (OCaml, Haskell, Scheme, ...) auront un rôle central.

1. Induction.

Sommaire.

1.1 Définitions inductives d'ensembles.

Dans ce cours, les ensembles définis par induction représenteront les données utilisées par les programmes. De plus, les notions d'ensembles et de types seront identiques : on identifiera :

$$\underbrace{n \in \mathsf{nat}}_{\text{ensemble}} \quad \longleftrightarrow \quad \underbrace{n : \mathsf{nat}}_{\text{type}}.$$

Exemple 1.1 (Types définis inductivement). Dans le code ci-dessous, on définit trois types : le type `nat` représentant les entiers naturels (construction de Peano) ; le type `nlist` représentant les listes d'entiers naturels ; et le type `t` représentant les arbres binaires étiquetés par des entiers aux nœuds.

```
type nat = Z | S of nat
type nlist = Nil | Const of nat * nlist
type t1 = F | N of t1 * nat * t1
```

Code 1.1 | *Trois types définis inductivement*

Définition 1.1. La *définition inductive d'un ensemble* `t` est la donnée de k constructeurs $\mathtt{C}_1, \ldots, \mathtt{C}_k$, où chaque $\mathtt{C}_i$ a pour argument un n_i-uplet dont le type est $\mathtt{u}_1^i * \mathtt{u}_2^i * \cdots * \mathtt{u}_{n_i}^i$. L'opération « $*$ » représente le produit cartésien, avec une notation « à la OCaml ». De plus, chaque $\mathtt{u}_j^i$ est, soit `t`, soit un type pré-existant.

Exemple 1.2.

```
type t2 =
| F
| N2 of (t * nlist * t)
| N3 of (t * nat * t * nat * t)
```

Code 1.2 | *Un exemple de type*

Définition 1.2. Les *types algébriques* sont définis en se limitant à deux opérations :

- ▷ le produit cartésien « $*$ » ;
- ▷ le « ou », noté « | » ou « $+$ », qui correspond à la somme disjointe ;

et un type :

- ▷ le type `unit`, noté 1.

Exemple 1.3 (Quelques types algébriques...).

- ▷ Le type bool est alors défini par $1 + 1$.
- ▷ Le type « jour de la semaine » est alors défini par l'expression $1 + 1 + 1 + 1 + 1 + 1 + 1$.
- ▷ Le type nat vérifie l'équation $X = 1 + X$.
- ▷ Le type nlist vérifie l'équation $X = 1 + \mathsf{nat} * X$.
- ▷ Le type t option est alors défini par $1 + \mathsf{t}$.

Ces ensembles définis inductivement nous intéresse pour deux raisons :

- ▷ pour pouvoir calculer, c'est à dire définir des fonctions de `t` vers `t'` en faisant du *filtrage* (*i.e.* avec `match ... with`)
- ▷ raisonner / prouver des propriétés sur les éléments de `t` : « des preuves par induction ».

1.2 Preuves par induction sur un ensemble inductif.

Exemple 1.4. Intéressons nous à `nat`. Pour prouver $\forall x \in \texttt{nat}, \mathscr{P}(x)$, il suffit de prouver *deux* choses (parce que l'on a deux constructeurs à l'ensemble `nat`) :

1. on doit montrer $\mathscr{P}(0)$;
2. et on doit montrer $\mathscr{P}(\texttt{S}\ n)$ en supposant l'*hypothèse d'induction* $\mathscr{P}(n)$.

Remarque 1.1. Dans le cas général, pour prouver $\forall x \in \texttt{t}, \mathscr{P}(x)$, il suffit de prouver les n propriétés (n est le nombre de constructeurs de l'ensemble `t`), où la i-ème propriété s'écrit :

> On montre $\mathscr{P}(\texttt{C}_i(x_1, \ldots, x_n))$ avec les hypothèses d'inductions $\mathscr{P}(x_j)$ lorsque $\texttt{u}_j^i = \texttt{t}$.

Exemple 1.5. Avec le type $\texttt{t}_2$ défini dans l'exemple 1.2, on a trois constructeurs, donc trois cas à traiter dans une preuve par induction. Le second cas s'écrit :

> On suppose $\mathscr{P}(x_1)$ et $\mathscr{P}(x_3)$ comme hypothèses d'induction, et on montre $\mathscr{P}(\texttt{N2}(x_1, k, x_3))$, où l'on se donne $k \in \texttt{nat}$.

Exemple 1.6. On pose la fonction `red` définie par le code ci-dessous.

```
let rec red k ℓ = match ℓ with
| Nil -> Nil
| Cons(x, ℓ) -> let ℓ'' = red k ℓ'' in
                 if x = k then ℓ''
                 else Cons(x, ℓ'')
```

Code 1.3 | *Fonction de filtrage d'une liste*

Cette fonction permet de supprimer toutes les occurrences de k dans une liste ℓ.

Démontrons ainsi la propriété

$$\forall \ell \in \mathsf{nlist}, \underbrace{\forall k \in \mathsf{nat}, \mathtt{size}(\mathtt{red}\ k\ \ell) \leq \mathtt{size}\ \ell}_{\mathscr{P}(\ell)}.$$

Pour cela, on procède par induction. On a *deux* cas.

1. Cas `Nil` : $\forall k \in \mathsf{nat}, \mathtt{size}(\mathtt{red}\ k\ \mathtt{Nil}) \leq \mathtt{size}\ \mathtt{Nil}$;
2. Cas $\mathtt{Cons}(x, \ell')$: on suppose

$$\forall k \in \mathsf{nat}, \mathtt{size}(\mathtt{red}\ k\ \ell) \leq \mathtt{size}\ \ell,$$

et on veut montrer que

$$\forall k \in \mathsf{nat}, \mathtt{size}(\mathtt{red}\ k\ \mathtt{Cons}(x, \ell')) \leq \mathtt{size}\ \mathtt{Cons}(x, \ell'),$$

ce qui demandera deux sous-cas : si $x = k$ et si $x \neq k$.

1.3 Définitions inductives de relations.

Dans ce cours, les relations définies par inductions représenterons des propriétés sur des programmes.

Un premier exemple : notations et terminologies.

Une relation est un sous-ensemble d'un produit cartésien. Par exemple, la relation $\mathsf{le} \subseteq \mathsf{nat} * \mathsf{nat}$ est une relation binaire. Cette relation re-

présente $\leq$, « *lesser than or equal to* » en anglais.

Notation. On note $\mathsf{le}(n,k)$ dès lors que l'on a $(n,k) \in \mathsf{le}$.

Pour définir cette relation, on peut écrire :

> Soit $\mathsf{le} \subseteq \mathsf{nat} * \mathsf{nat}$ la relation qui vérifie :
>
> 1. $\forall n \in \mathsf{nat}, \mathsf{le}(n,n)$;
> 2. $\forall (n,k) \in \mathsf{nat} * \mathsf{nat}$, si $\mathsf{le}(n,k)$ alors $\mathsf{le}(n, \mathtt{S}\ k)$.

mais, on écrira plutôt :

> Soit $\mathsf{le} \subseteq \mathsf{nat} * \mathsf{nat}$ la relation définie (inductivement) à partir des règles d'inférence suivantes :
>
> $$\frac{}{\mathsf{le}(n,n)}\ \mathscr{L}_1 \qquad \frac{\mathsf{le}(n,k)}{\mathsf{le}(n, \mathtt{S}\ k)}\ \mathscr{L}_2.$$

Remarque 1.2.
- ▷ Dans la définition par règle d'inférence, chaque règle a *une* conclusion de la forme $\mathsf{le}(\cdot,\cdot)$.
- ▷ Les *métavariables* n et k sont quantifiées universellement de façon implicite.

Définition 1.3. On appelle *dérivation* ou *preuve* un arbre construit en appliquant les règles d'inférence (ce qui fait intervenir l'*instantiation des métavariables*) avec des axiomes aux feuilles.

Exemple 1.7. Pour démontrer $\mathsf{le}(2,4)$, on réalise la dérivation ci-dessous.

$$\dfrac{\dfrac{\dfrac{}{\mathsf{le}(2,2)}\ \mathscr{L}_1}{\mathsf{le}(2,3)}\ \mathscr{L}_2}{\mathsf{le}(2,4)}\ \mathscr{L}_2$$

Exemple 1.8. On souhaite définir une relation triée sur nlist. Pour

cela, on pose les trois règles ci-dessous :

$$\frac{}{\mathsf{triée}\ \mathtt{Nil}}\ \mathscr{T}_1 \qquad \frac{}{\mathsf{triée}\ \mathtt{Cons}(x, \mathtt{Nil})}\ \mathscr{T}_2,$$

$$\frac{\mathsf{le}(x, y) \qquad \mathsf{triée}\ \mathtt{Cons}(x, \mathtt{Nil})}{\mathsf{triée}\ \mathtt{Cons}(x, \mathtt{Cons}(y, \ell))}\ \mathscr{T}_3.$$

Ceci permet de dériver, modulo quelques abus de notations, que la liste `[1;3;4]` est triée :

$$\dfrac{\dfrac{\dfrac{}{1 \leq 1}\ \mathscr{L}_1}{1 \leq 2}\ \mathscr{L}_2}{1 \leq 3}\ \mathscr{L}_2 \qquad \dfrac{\dfrac{\dfrac{}{3 \leq 3}\ \mathscr{L}_1}{3 \leq 4}\ \mathscr{L}_2 \qquad \dfrac{}{\mathsf{triée}\ \mathtt{[4]}}\ \mathscr{R}_2}{\mathsf{triée}\ \mathtt{[3;4]}}\ \mathscr{R}_3}{\mathsf{triée}\ \mathtt{[1;3;4]}}\ \mathscr{R}_3.$$

Les parties en bleu de l'arbre ne concernent pas la relation triée, mais la relation le.

Exemple 1.9. On définit la relation mem d'*appartenance* à une liste. Pour cela, on définit $\mathsf{mem} \subseteq \mathsf{nat} * \mathsf{nlist}$ par les règles d'inférences :

$$\frac{}{\mathsf{mem}(k, \mathtt{Cons}(k, \ell))}\ \mathscr{M}_1 \qquad \frac{\mathsf{mem}(k, \ell)}{\mathsf{mem}(k, \mathtt{Cons}(x, \ell))}\ \mathscr{M}_2.$$

On peut constater qu'il y a plusieurs manières de démontrer

$$\mathsf{mem}(0, \mathtt{[0;1;0]}).$$

Ceci est notamment dû au fait qu'il y a deux '0' dans la liste.

Remarque 1.3. Attention ! Dans les prémisses d'une règle, on ne peut pas avoir « $\neg\mathsf{r}(\ldots)$ ». Les règles ne peuvent qu'être « constructive », donc pas de négation.

Exemple 1.10. On définit la relation $\mathsf{ne} \subseteq \mathsf{nat} * \mathsf{nat}$ de non égalité entre deux entiers.

On pourrait imaginer créer une relation d'égalité et de définir ne comme sa négation. Mais non, c'est ce que nous dit la remarque 1.3.

On peut cependant définir la relation ne par :

$$\frac{}{\mathsf{ne}(\mathtt{Z}, \mathtt{S}\ k)}\ \mathscr{N}_1 \qquad \frac{}{\mathsf{ne}(\mathtt{S}\ n, \mathtt{Z})}\ \mathscr{N}_2 \qquad \frac{\mathsf{ne}(n,k)}{\mathsf{ne}(\mathtt{S}\ n, \mathtt{S}\ k)}\ \mathscr{N}_3.$$

Il est également possible de définir ne à partir de la relation le.

Exemple 1.11. En utilisant la relation ne (définie dans l'exemple 1.10), on peut revenir sur la relation d'appartenance et définir une relation alternative à celle de l'exemple 1.9. En effet, soit la relation mem' définie par les règles d'inférences ci-dessous :

$$\frac{}{\mathsf{mem}'(n, \mathtt{Cons}(n, \ell))}\ \mathscr{M}'_1 \qquad \frac{\mathsf{mem}'(n,\ell) \qquad \mathsf{ne}(k,n)}{\mathsf{mem}'(n, \mathtt{Cons}(k, \ell))}\ \mathscr{M}'_2.$$

Il est (*sans doute ?*) possible de montrer que :

$$\forall (n, \ell) \in \mathsf{nat} * \mathsf{nlist}, \mathsf{mem}(n,\ell) \iff \mathsf{mem}'(n,\ell).$$

Remarque 1.4. Dans le cas général, une définition inductive d'une relation Rel, c'est k règles d'inférences de la forme :

$$\frac{H_1 \quad \cdots \quad H_n}{\mathsf{Rel}(x_1, \ldots, x_m)}\ \mathscr{R}_i,$$

où chaque H_j est :

- ▷ soit $\mathsf{Rel}(\ldots)$;

▷ soit une autre relation pré-existante (*c.f.* la définition de triée dans l'exemple 1.8).

On appelle les H_j les *prémisses*, et $\mathsf{Rel}(x_1, \dots, x_m)$ la *conclusion*. Elles peuvent faire intervenir des *métavariables*.

1.4 Preuves par induction sur une relation inductive.

On souhaite établir une propriété de la forme

$$\forall (x_1, \dots, x_m),\ \mathsf{Rel}(x_1, \dots, x_m) \implies \mathscr{P}(x_1, \dots, x_m).$$

Pour cela, on établit autant de propriétés qu'il y a de règles d'inférences sur la relation Rel. Pour chacune de ces propriétés, on a une hypothèse d'induction pour chaque prémisse de la forme $\mathsf{Rel}(\dots)$.

Exemple 1.12 (Induction sur la relation le.)**.** Pour prouver une propriété

$$\forall (n, k) \in \mathsf{nat} * \mathsf{nat}, \quad \mathsf{le}(n, k) \implies \mathscr{P}(n, k),$$

il suffit d'établir *deux* propriétés :

1. $\forall n, \mathscr{P}(n, n)$;
2. pour tout (n, k), montrer $\mathscr{P}(n, \mathtt{S}\ k)$ en supposant $\mathscr{P}(n, k)$.

Exemple 1.13. Supposons que l'on ait une fonction ayant pour signature $\mathtt{sort} : \mathsf{nlist} \to \mathsf{nlist}$ qui trie une nlist. On souhaite démontrer la propriété :

$$\forall \ell \in \mathsf{nlist}, \mathsf{triée}(\ell) \implies \mathtt{sort}(\ell) = \ell.$$

On considère deux approches pour la démonstration : par induction sur ℓ et par induction sur la relation $\mathsf{triée}$.

1. par induction sur la liste ℓ, il y a *deux* cas à traiter :

   - ▷ montrer que $\mathsf{triée}(\mathtt{Nil}) \implies \mathtt{sort}(\mathtt{Nil}) = \mathtt{Nil}$,
   - ▷ montrer que :

$$\mathsf{triée}(\mathtt{Cons}(n, \ell)) \implies \mathtt{sort}(\mathtt{Cons}(n, \ell)) = \mathtt{Cons}(n, \ell);$$

2. par induction sur la relation $\mathsf{triée}(\ell)$, il y a *trois* cas à traiter :
   - ▷ montrer $\mathtt{sort}(\mathtt{Nil}) = \mathtt{Nil}$,
   - ▷ montrer $\mathtt{sort}(\mathtt{Cons}(n, \mathtt{Nil})) = \mathtt{Cons}(n, \mathtt{Nil})$,
   - ▷ montrer $\mathtt{sort}(\mathtt{Cons}(x, \mathtt{Cons}(y, \ell)) = \mathtt{Cons}(x, \mathtt{Cons}(y, \ell)))$, en supposant :
     - $\mathsf{triée}(\mathtt{Cons}(y, \ell))$ et $\mathscr{P}(\mathtt{Cons}(y, \ell))$, pour la première prémisse ;
     - $\mathsf{le}(x, y)$, pour la seconde prémisse.

2. Théorèmes de point fixe.

Sommaire.

Dans cette section, on va formaliser les raisonnements que l'on a réalisé en section 1 à l'aide du théorème de Knaster-Tarksi.

Définition 2.1. Soit E un ensemble, une relation $\mathscr{R} \subseteq E^2$ est un *ordre partiel* si $\mathscr{R}$ est :

- ▷ réflexive : $\forall x \in E,\ x\ \mathscr{R}\ x$;
- ▷ transitive : $\forall x, y, z \in E,\ (x\ \mathscr{R}\ y \text{ et } y\ \mathscr{R}\ z) \implies x\ \mathscr{R}\ z$;
- ▷ antisymétrique : $\forall x, y \in E,\ (x\ \mathscr{R}\ y \text{ et } y\ \mathscr{R}\ x) \implies x = y$.

Exemple 2.1. Dans l'ensemble $E = \mathbb{N}$, les relations $\leq$ et $\mid$ (division) sont des ordres partiels.

Définition 2.2. Soit $(E, \sqsubseteq)$ un ordre partiel.

▷ Un *minorant* d'une partie $A \subseteq E$ est un $m \in E$ tel que

$$\forall x \in A,\ m \sqsubseteq x.$$

▷ Un *majorant* d'une partie $A \subseteq E$ est un $m' \in E$ tel que

$$\forall x \in A,\ x \sqsubseteq m'.$$

▷ Un *treillis complet* est un ordre partiel $(E, \sqsubseteq)$ tel que toute partie $A \subseteq E$ admet un *plus petit majorant*, noté $\bigsqcup A$, et un *plus grand minorant*, noté $\bigsqcap A$.

Remarque 2.1.
▷ Pour tout minorant m de A, on a $m \sqsubseteq \bigsqcap A$.
▷ Pour tout majorant m' de A, on a $\bigsqcup A \sqsubseteq m'$.
▷ Un minorant/majorant de A n'est pas nécessairement dans l'ensemble A. Ceci est notamment vrai pour $\bigsqcap A$ et $\bigsqcup A$.

Notation. On note généralement $\bot = \bigsqcap E$, et $\top = \bigsqcup E$.

Exemple 2.2.
▷ L'ensemble $(\mathbb{N}, \leq)$ n'est pas un treillis complet : si A est infini, il n'admet pas de plus petit majorant.
▷ L'ensemble $(\mathbb{N} \cup \{\infty\}, \leq)$ est un treillis complet avec la convention $\forall n \in \mathbb{N}, n \leq \infty$.
▷ L'ensemble $(\mathbb{N}, |)$ est un treillis complet :
– pour $A \subseteq \mathbb{N}$ fini, on a

$$\bigsqcup A = \operatorname{ppcm} A \quad \text{et} \quad \bigsqcap A = \operatorname{pgcd} A;$$

– pour $A \subseteq \mathbb{N}$ infini, les relations ci-dessus restent valables avec la convention :

$$\forall n \in \mathbb{N},\quad n \mid 0.$$

Exemple 2.3 (Exemple *très* important de treillis complet). Soit E_0 un ensemble. Alors l'ensemble $(\wp(E_0), \subseteq)$ des parties de E_0 est un treillis complet. En particulier, on a :

$$\bigsqcap = \bigcap, \quad \bigsqcup = \bigcup, \quad \bot = \emptyset \quad \text{et} \quad \top = E_0.$$

Théorème 2.1 (Knaster-Tarski)**.** Soit $(E, \sqsubseteq)$ un treillis complet. Soit f une fonction croissante de E dans E.[1] On considère l'ensemble

$$F_f = \{x \in E \mid f(x) \sqsubseteq x\},$$

l'ensemble des *prépoints fixes* de f. Posons $m = \bigsqcap F_f$. Alors, m est un point fixe de f, *i.e.* $f(m) = m$.

Preuve. Soit $y \in F_f$, alors $m \sqsubseteq y$, et par croissance de f, on a ainsi $f(m) \sqsubseteq f(y)$, ce qui implique $f(m) \sqsubseteq y$ par transitivité (et car $y \in F_f$). D'où, $f(m)$ est un minorant de F_f.

Or, par définition, $f(m) \sqsubseteq m$, et par croissance $f(f(m)) \sqsubseteq f(m)$, ce qui signifie que $f(m) \in F_f$. On en déduit $m \sqsubseteq f(m)$.

Par antisymétrie, on en conclut que $f(m) = m$. □

À la suite de ce théorème, on peut formaliser les raisonnements que l'on a réalisé en section 1. Pour cela, il nous suffit d'appliquer le théorème 2.1 de Knaster-Tarksi (abrégé en « théorème K-T »).

2.1 Définitions inductives de relations.

Remarque 2.2. Pour justifier la définition des relations, on applique le théorème K-T. En effet, on part de $E = E_1 * \cdots * E_n$. Les relations sont des sous-ensembles de E, on travaille donc dans le treillis complet $(\wp(E), \subseteq)$. On se donné une définition inductive d'une relation $\mathsf{Rel} \subseteq E$. Pour cela, on s'appuie sur les règles

1. Ceci signifie que $\forall a, b \in E, \quad a \sqsubseteq b \implies f(a) \sqsubseteq f(b)$.

d'inférences et on associe à chaque $\mathscr{R}_i$ une fonction

$$f_i : \wp(E) \to \wp(E).$$

On montre (constate) que les f_i définies sont croissantes. Enfin, on pose pour $A \subseteq E$,

$$f(A) = f_1(A) \cup \cdots \cup f_k(A).$$

La fonction $f \mapsto f(A)$ est croissante.

Par définition, Rel est défini comme le plus petit (pré)-point fixe de la fonction f, qui existe par le théorème K-T (théorème 2.1).

Exemple 2.4. Définissons $\mathsf{le} \subseteq \mathsf{nat} * \mathsf{nat}$. On rappelle les règles d'inférences pour cette relation :

$$\frac{}{\mathsf{le}(n,n)}\ \mathscr{L}_1 \qquad \frac{\mathsf{le}(n,k)}{\mathsf{le}(n,\mathtt{S}\ k)}\ \mathscr{L}_2.$$

Avec un ensemble $A \subseteq \mathsf{nat} * \mathsf{nat}$, on définit

$$f_1(A) = \{(n,n) \mid n \in \mathsf{nat}\},$$

$$f_2(A) = \{(n, \mathtt{S}\ k) \mid (n,k) \in A\};$$

et on pose enfin

$$f(A) = f_1(A) \cup f_2(A).$$

La définition formelle de la relation le est le plus petit point fixe de f.

Exemple 2.5 (Suite de l'exemple 1.8)**.** Définissons $\mathsf{triée} \subseteq \mathsf{nlist}$. On rappelle les règles d'inférences pour cette relation :

$$\frac{}{\mathsf{triée}\ \mathtt{Nil}}\ \mathscr{T}_1 \qquad \frac{}{\mathsf{triée}\ \mathtt{Cons}(x, \mathtt{Nil})}\ \mathscr{T}_2,$$

$$\frac{\mathsf{le}(x,y) \qquad \text{triée } \mathtt{Cons}(x,\mathtt{Nil})}{\text{triée } \mathtt{Cons}(x,\mathtt{Cons}(y,\ell))}\ \mathcal{T}_3.$$

Avec un ensemble $A \subseteq \mathsf{nlist}$, on définit

$$\begin{aligned} f_1(A) &= \{\mathtt{Nil}\},\\ f_2(A) &= \{\mathtt{Cons}(k,\mathtt{Nil}) \mid k \in \mathsf{nat}\},\\ f_3(A) &= \left\{ \mathtt{Cons}(x,\mathtt{Cons}(y,\ell)) \;\middle|\; \begin{array}{l} \mathtt{Cons}(y,\ell) \in A \\ \mathsf{le}(x,y) \end{array} \right\}, \end{aligned}$$

et on pose enfin

$$f(A) = f_1(A) \cup f_2(A).$$

La définition formelle de la relation le est le plus petit point fixe de f.

Remarque 2.3. Dans les exemples ci-avant, même si l'on ne l'a pas précisé, les fonctions f_i sont bien croissantes pour l'inclusion $\subseteq$. C'est ceci qui assure l'application du théorème K-T (théorème 2.1).

Comme dit dans la remarque 1.3, on ne définit pas de règles d'induction de la forme

$$\frac{\neg\mathsf{Rel}(x'_1,\ldots,x'_{n'})}{\mathsf{Rel}(x_1,\ldots,x_n)} \qquad \longrightarrow \quad \text{C'est interdit !}$$

En effet, la fonction f définie n'est donc plus croissante.

Remarque 2.4. Une relation R définie comme le plus petit point fixe d'une fonction f vérifie, mais on ne demande en rien que l'on ait $A \subseteq f(A)$ quel que soit $A \subseteq E$. En effet, pour

$$f(\{(3,2)\}) = \{(n,n) \mid n \in \mathsf{nat}\} \cup \{(3,1)\}$$

ne vérifie pas cette propriété.

2.2 Définitions inductives d'ensembles.

Exemple 2.6. On reprend le type $\mathtt{t}_2$ défini à l'exemple 1.2 :

```
type t₂ =
| F
| N2 of (t ∗ nlist ∗ t)
| N3 of (t ∗ nat ∗ t ∗ nat ∗ t)
```

Code 2.1 | *Un exemple de type*

On le définit en utilisant le théorème K-T (théorème 2.1) en posant :

$$f_1(A) = \{\mathtt{F}\}$$
$$f_2(A) = \{(x, \ell, y) \mid \ell \in \mathsf{nlist} \text{ et } (x, y) \in A^2\}$$
$$f_3(A) = \left\{ (x, k_1, y, k_2, z) \;\middle|\; \begin{array}{l} (x, y, z) \in A^3 \\ (k_1, k_2) \in \mathsf{nat}^2 \end{array} \right\},$$

puis, quel que soit A,

$$f(A) = f_1(A) \cup f_2(A) \cup f_3(A).$$

On pose ensuite $\mathtt{t}_2$ comme le plus petit point fixe de f.

Exemple 2.7. Avec $\mathsf{nat} = \{\mathtt{Z}, \mathtt{S}\ \mathtt{Z}, \mathtt{S}\ \mathtt{S}\ \mathtt{Z}, \ldots\}$, on utilise

$$f(A) = \{\mathtt{Z}\} \cup \{\mathtt{S}\ n \mid n \in A\},$$

et on pose nat le plus petit point fixe de f.

Et si on retire le cas de base ? Que se passe-t-il ? On pose la fonction

$$f'(A) = \{\mathtt{S}\ n \mid n \in A\}.$$

Le plus petit point fixe de f est l'ensemble vide $\emptyset$. On ne définit donc pas les entiers naturels.

Remarque 2.5. Après quelques exemples, il est important de se demander comment f est définie. C'est une fonction de la forme

$$f : \wp(\boxed{?}) \to \wp(\boxed{?}).$$

Quel est l'ensemble noté « $\boxed{?}$ » ? Quel est l'*ensemble ambiant* ?

La réponse est : c'est l'ensemble des arbres étiquetés par des chaînes de caractères.

Remarque 2.6. Pour définir inductivement un relation, on peut considérer qu'on construit un ensemble de dérivation.

Par exemple, pour le, on aurait

$$f_2(A) = \left\{ \frac{\delta}{\mathsf{le}(n, \mathtt{S}\ k)} \;\middle|\; \begin{array}{l} \delta \text{ est une dérivation de } \mathsf{le}(n,k) \textit{ i.e.,} \\ \delta \text{ est une dérivation dont } \mathsf{le}(n,k) \text{ est} \\ \text{à la racine} \end{array} \right\}.$$

2.3 Preuves par induction sur un ensemble inductif.

Remarque 2.7. Soit $\mathtt{t}$ un ensemble défini par induction par les constructeurs $\mathtt{C}_1, \ldots, \mathtt{C}_n$. On pose f tel que $\mathtt{t}$ est le plus petit pré-point fixe de f.

On veut montrer $\forall x \in \mathtt{t}, \mathscr{P}(x)$. Pour cela, on pose

$$A = \{x \in \mathtt{t} \mid \mathscr{P}(x)\},$$

et on montre que $f(A) \subseteq A$, *i.e.* A est un pré-point fixe de f. Ceci implique, par définition de $\mathtt{t}$, que $\mathtt{t} \subseteq A$, d'où

$$\forall x, x \in \mathtt{t} \implies \mathscr{P}(x).$$

Exemple 2.8. Expliquons ce que veut dire « montrer $f(A) \subseteq A$ » sur un exemple.

Pour nlist, on pose deux fonctions

$$f_1(A) = \{\texttt{Nil}\}$$
$$f_2(A) = \{\texttt{Cons}(k, \ell) \mid \ell \in A\}$$
$$.$$

Pour montrer $f(A) \subseteq A$, il y a *deux* cas :

- ▷ (pour f_1) montrer $\mathscr{P}(\texttt{Nil})$;
- ▷ (pour f_2) avec l'hypothèse d'induction $\mathscr{P}(\ell)$, et $k \in \mathsf{nat}$, montrer $\mathscr{P}(\texttt{Cons}(n, \ell))$.

2.4 Preuves par induction sur une relation inductive.

2.4.1 Une première approche...

Remarque 2.8. Soit Rel une relation définie comme le plus petit (pré)point fixe d'une fonction f, associée aux k règles d'inférences $\mathscr{R}_1, \ldots, \mathscr{R}_k$. On veut montrer que

$$\forall (x_1, \ldots, x_m) \in E, \quad \mathsf{Rel}(x_1, \ldots, x_m) \implies \mathscr{P}(x_1, \ldots, x_m).$$

Pour cela, on pose $A = \{(x_1, \ldots, x_m) \in E \mid \mathscr{P}(x_1, \ldots, x_m)\}$, et on montre que $f(A) \subseteq A$, *i.e.* que A est un prépoint fixe de f. Ainsi, on aura $\mathsf{Rel} \subseteq A$ et on aura donc montré

$$\forall (x_1, \ldots, x_m) \in E, \quad \mathsf{Rel}(x_1, \ldots, x_m) \implies \mathscr{P}(x_1, \ldots, x_m).$$

Exemple 2.9. Pour le, prouver $f(A) \subseteq A$ signifie prouver deux propriétés :

1. $\forall n \in \mathsf{nat}, \mathscr{P}(n)$;
2. $\forall (n,k) \in \mathsf{nat}^2, \underbrace{\mathscr{P}(n,k)}_{\text{hyp. ind.}} \implies \mathscr{P}(n, \mathtt{S}\ k)$

Exemple 2.10. Pour triée, on a *trois* propriétés à prouver :

1. $\mathscr{P}(\mathtt{Nil})$;
2. $\forall k \in \mathsf{nat}, \mathscr{P}(\mathtt{Cons}(k, \mathtt{Nil}))$;
3. $\forall (x,y) \in \mathsf{nat}^2, \forall \ell \in \mathsf{nlist}$,

$$\underbrace{\mathscr{P}(\mathtt{Cons}(y,\ell))}_{\text{hyp.ind}} \wedge \mathsf{le}(x,y) \implies \mathscr{P}(\mathtt{Cons}(x, \mathtt{Cons}(y,\ell))).$$

Remarque 2.9. Remarquons que dans l'exemple 2.10 ci-dessus, dans le 3ème cas, on n'a pas d'hypothèse $\mathsf{triée}(\mathtt{Cons}(y,\ell))$. Ceci vient du fait que, dans la remarque 2.7, l'ensemble A ne contient pas que des listes triées. La contrainte de la relation n'a pas été appliquée, on n'a donc pas accès à cette hypothèse.

2.4.2 Une approche plus astucieuse...

Remarque 2.10. On modifie légèrement le raisonnement présenté en remarque 2.7. On pose

$$A' = \{(x_1, \dots, x_m) \in E \mid \mathsf{Rel}(x_1, \dots, x_m) \wedge \mathscr{P}(x_1, \dots, x_m)\}.$$

On montre $f(A') \subseteq A'$ et donc, par définition de Rel, on aura l'inclusion $\mathsf{Rel} \subseteq A'$. Avec ce raisonnement, on peut utiliser des hypothèses, comme montré dans les exemples 2.11 et 2.12. Le but de la preuve n'est donc plus $\mathscr{P}(\dots)$ mais $\mathsf{Rel}(\dots) \wedge \mathscr{P}(\dots)$.

En rouge sont écrits les différences avec le raisonnement précédent.

Exemple 2.11 (Version améliorée de l'exemple 2.9)**.** Pour le, prouver $f(A) \subseteq A$ signifie prouver deux propriétés :

1. $\forall n \in \mathsf{nat}, \mathsf{le}(n, n) \wedge \mathscr{P}(n)$;
2. $\forall (n, k) \in \mathsf{nat}^2, \underbrace{\mathsf{le}(n, k) \wedge \mathscr{P}(n, k)}_{\text{hyp. ind.}} \implies \mathsf{le}(n, \mathtt{S}\ k) \wedge \mathscr{P}(n, \mathtt{S}\ k)$

Exemple 2.12 (Version améliorée de l'exemple 2.10)**.** Pour $\mathsf{triée}$, on a *trois* propriétés à prouver :

1. $\mathsf{triée}(\mathtt{Nil}) \wedge \mathscr{P}(\mathtt{Nil})$;
2. $\forall k \in \mathsf{nat}, \mathsf{triée}(\mathtt{Cons}(k, \mathtt{Nil})) \wedge \mathscr{P}(\mathtt{Cons}(k, \mathtt{Nil}))$;
3. $\forall (x, y) \in \mathsf{nat}^2, \forall \ell \in \mathsf{nlist}$,

$$\overbrace{\mathsf{triée}(\mathtt{Cons}(y, \ell)) \wedge \mathscr{P}(\mathtt{Cons}(y, \ell))}^{\text{hyp.ind}} \wedge \mathsf{le}(x, y)$$
$$\Downarrow$$
$$\mathsf{triée}(\mathtt{Cons}(x, \mathtt{Cons}(y, \ell))) \wedge \mathscr{P}(\mathtt{Cons}(x, \mathtt{Cons}(y, \ell)))$$

2.5 Domaines et points fixes.

Définition 2.3. Soit $(E, \sqsubseteq)$ un ordre partiel. Une *chaîne infinie* dans l'ensemble ordonné $(E, \sqsubseteq)$ est une suite $(e_n)_{n \geq 0}$ telle que

$$e_0 \sqsubseteq e_1 \sqsubseteq e_2 \sqsubseteq \cdots .$$

On dit que $(E, \sqsubseteq)$ est *complet* si pour toute chaîne infinie, il existe $\bigsqcup_{n \geq 0} e_n \in E$, un plus petit majorant dans E.

Si, de plus, E a un plus petit élément $\bot$, alors $(E, \sqsubseteq)$ est un *domaine*.

Remarque 2.11. Un treillis complet est un domaine.

Théorème 2.2. Soit $(E, \sqsubseteq)$ un domaine. Soit $f : E \to E$ *continue* :

- ▷ f est croissante ;
- ▷ pour toute chaîne infinie $(e_n)_{n\geq 0}$,

$$f\Big(\bigsqcup_{n\geq 0} e_n\Big) = \bigsqcup_{n\geq 0} f(e_n).$$

Les $(f(e_n))_{n\geq 0}$ forment une chaîne infinie par croissance de la fonction f.

On pose, quel que soit $x \in E$, $f^0(x) = x$, et pour tout entier $i \geq 0$, on définit $f^{i+1}(x) = f(f^i(x))$.

On pose enfin

$$\begin{aligned}\mathrm{fix}(f) &= \bigsqcup_{n\geq 0} f^n(\bot) \\ &= \bot \sqcup f(\bot) \sqcup f^2(\bot) \sqcup \cdots .\end{aligned}$$

Alors, $\mathrm{fix}(f)$ est le plus petit point fixe de f.

Preuve. La preuve viendra plus tard. □

Les définitions inductives par constructeurs ou règles d'inférences peuvent être définis par des fonctions continues. Et, on peut se placer dans le domaine $(\wp(E), \subseteq)$ pour définir les ensembles définis par inductions.

Exemple 2.13. Avec les listes d'entiers, on définit

$$\mathtt{nat} = \underbrace{\emptyset}_{\bot} \cup \underbrace{\{\mathtt{Nil}\}}_{f(\bot)} \cup \underbrace{\{\mathtt{Cons}(k, \mathtt{Nil}) \mid k \in \mathtt{nat}\}}_{f^2(\bot)} \cup \cdots .$$

3. Les bases de Rocq.

3.1 Les définitions par induction : `Inductive`.

En Rocq (anciennement Coq), on peut définir des ensembles par induction. Pour cela, on utilise le mot `Inductive`.

Par exemple, pour définir un type de liste d'entiers, on utilise le code ci-dessous.

```
Inductive nlist : Set:=
| Nil : nlist
| Cons : nat → nlist → nlist.
```

Code 3.1 | *Définition du type* nlist *en Rocq*

En Rocq, au lieu de définir la fonction `Cons` comme une « fonction » de la forme `Cons` : nat∗nlist → nlist, on la *curryfie* en une « fonction » de la forme `Cons` : nat → nlist → nlist. Les types définis par les deux versions sont isomorphes.

Pour définir une relation, on utilise aussi le mot clé `Inductive` :

```
Inductive le : nat → nat → Prop :=
| le_refl : forall n, le n n
| le_S : forall n k, le n k → le (S n) (S k).
```

Code 3.2 | *Définition de la relation* le

Aux types définis par induction, on associe un principe d'induction (qu'on voir avec `Print le_ind.` ou `Print nlist_ind.`). Ce principe d'induction permet de démontrer une propriété $\mathcal{P}$ sur un ensemble/une relation définie par induction.

3.2 Quelques preuves avec Rocq.

On décide de prouver le lemme suivant avec Rocq.

> **"Lemme" 3.1.** Soit ℓ une liste triée, et soient a et b deux entiers tels que $a \leq b$. Alors la liste $a :: b :: \ell$ est triée.

Pour cela, on écrit en Rocq :

```
Lemma exemple_triee :
  forall l, triée l →
    forall a b, le a b →
      triée (Cons a (Cons b l)).
```

Il ne reste plus qu'à prouver ce lemme. On commence la démonstration par introduire les variables et hypothèses : les variables `l`, `a`, `b`, et les hypothèses (`H1`) : triée `l`, et (`H2`) : le `a` `b`. On commence par introduire la liste `l` et l'hypothèse `H1` et on s'occupera des autres un peu après.

```
Proof.
  intros l H1.
```

On décide de réaliser une preuve par induction sur la relation triée, qui est en hypothèse (`H1`).

```
  induction H1.
```

Dans le cas d'une preuve par induction sur triée, on a *trois* cas.

- ▷ *Cas 1.* On se trouve dans le cas `l` = `Nil`. Pas trop de problèmes pour prouver que `[a;b]` est triée avec l'hypothèse `a` ≤ `b`. On introduit les variables et hypothèses `a`, `b` et `H2`.

  ```
  - intros a b H2.
  ```

  À ce moment de la preuve, l'objectif est de montrer :

  $$\text{triée } \texttt{Cons}(\texttt{a}, \texttt{Cons}(\texttt{b}, \texttt{Nil})).$$

  Pour cela, on utilise deux fois les propriétés de la relation triée :

```
apply t_cons.
apply t_singl.
```

Notre objectif a changé, on doit maintenant démontrer `le a b`. C'est une de nos hypothèses, on peut donc utiliser :

```
assumption.
```

Ceci termine le cas 1.

▷ *Cas 2.* On se trouve dans le cas `l = [k]`. On doit de démontrer que la liste `[a;b;k]` est triée. On a l'hypothèse `a` $\leq$ `b`, mais aucune hypothèse de la forme `b` $\leq$ `k`. On est un peu coincé pour ce cas...

(Un jour je finirai d'écrire cette partie... Malheureusement, ce n'est pas aujourd'hui...)

4. Sémantique opérationnelle pour les expressions arithmétiques simples (EA).

Sommaire.

Depuis le début du cours, on s'est intéressé à la *méthode inductive*. On essaie d'appliquer cette méthode à « l'exécution » des « programmes ».

On définira un programme comme un ensemble inductif : un programme est donc une structure de donnée. L'exécution d'un programme sera décrit comme des relations inductives (essentiellement binaires) sur les programmes. Définir ces relations, cela s'appelle la *sémantique opérationnelle*.

On considèrera deux sémantiques opérationnelles

- ▷ la sémantique à grands pas, où l'on associe un résultat à un programme ;
- ▷ la sémantique à petits pas, où l'on associe un programme « un peu plus tard » à un programme.

Notre objectif, dans un premier temps, est de définir OCaml, ou plutôt un plus petit langage fonctionnel inclus dans OCaml.

On se donne l'ensemble $\mathbb{Z}$ (on le prend comme un postulat). On définit l'ensemble EA en Rocq par :

```
Inductive EA : Set :=
|Cst : ℤ → EA
|Add : EA → EA → EA.
```

Code 4.1 | *Définition des expressions arithmétiques simples*

Note 4.1. On se donne $\mathbb{Z}$ et on note $k \in \mathbb{Z}$ (vu comme une métavariable). On définit (inductivement) l'ensemble EA des expressions arithmétiques, notées $a, a', a_1, \ldots$ par la grammaire

$$a ::= \underline{k} \mid a_1 \oplus a_2.$$

Exemple 4.1. L'expression $\underline{1} \oplus (\underline{3} \oplus \underline{7})$ représente l'expression Rocq

$$\texttt{Add}(\texttt{Cst}\ 1, \texttt{Add}\ (\texttt{Cst}\ 3)\ (\texttt{Cst}\ 7)),$$

que l'on peut représenter comme l'arbre de syntaxe...

Remarque 4.1. Dans le but de définir un langage minimal, il n'y a donc pas d'intérêt à ajoute $\ominus$ et $\otimes$, représentant la soustraction et la multiplication.

4.1 Sémantique à grands pas sur EA.

On définit la sémantique opérationnelle à grands pas pour EA. L'intuition est d'associer l'exécution d'un programme avec le résultat. On définit la relation d'évaluation $\Downarrow \subseteq \mathsf{EA} * \mathbb{Z}$, avec une notation infixe, définie par les règles d'inférences suivantes :

$$\frac{}{\underline{k} \Downarrow k} \quad \text{et} \quad \frac{a_1 \Downarrow k_1 \qquad a_2 \Downarrow k_2}{a_1 \oplus a_2 \Downarrow k},$$

où, dans la seconde règle d'inférence, $k = k_1 + k_2$. Attention, le $+$ est la somme dans $\mathbb{Z}$, c'est une opération *externalisée*. Vu qu'on ne sait pas comment la somme a été définie dans $\mathbb{Z}$ (on ne sait pas si elle est définie par induction/point fixe, ou pas du tout), on ne l'écrit pas dans la règle d'inférence.

La forme générale des règles d'inférences est la suivante :

$$\text{Cond. App.}\ \frac{P_1 \quad \dots \quad P_m}{C}\ \mathscr{R}_i$$

où l'on donne les conditions d'application (ou *side condition* en anglais). Les $P_1, \dots, P_m, C$ sont des relations inductives, mais les conditions d'applications **ne sont pas** forcément inductives.

Exemple 4.2.

$$3+7=10\ \frac{\underline{3} \Downarrow 3 \qquad 2+5=7\ \dfrac{\underline{2} \Downarrow 2 \quad \underline{5} \Downarrow 5}{(\underline{2} \oplus \underline{5}) \Downarrow 7}}{\underline{3} \oplus (\underline{2} \oplus \underline{5}) \Downarrow 10}\ .$$

4.2 Sémantique à petits pas sur EA.

On définit ensuite la sémantique opérationnelle à *petits pas* pour EA. L'intuition est de faire un pas exactement (la relation n'est donc pas réflexive) dans l'exécution d'un programme et, si possible, qu'elle soit déterministe.

Une relation *déterministe* (ou *fonctionnelle*) est une relation $\mathscr{R}$ telle que, si $a\ \mathscr{R}\ b$ et $a\ \mathscr{R}\ c$ alors $b = c$.

La relation de réduction $\to \subseteq \mathsf{EA} * \mathsf{EA}$, notée infixe, par les règles d'inférences suivantes

$$k = k_1 + k_2\ \frac{}{\underline{k_1} \oplus \underline{k_2} \to \underline{k}}\ \mathscr{A},$$

$$\frac{a_2 \to a_2'}{a_1 \oplus a_2 \to a_1 \oplus a_2'}\ \mathscr{C}_{\mathrm{d}} \quad \text{et} \quad \frac{a_1 \to a_1'}{a_1 \oplus \underline{k} \to a_1' \oplus \underline{k}}\ \mathscr{C}_{\mathrm{g}}.$$

Il faut le comprendre par « quand c'est fini à droite, on passe à gauche ».

Les règles $\mathscr{C}_{\mathrm{g}}$ et $\mathscr{C}_{\mathrm{d}}$ sont nommées respectivement *règle contextuelle droite* et *règle contextuelle gauche*. Quand $a \to a'$, on dit que a se *réduit* à a'.

Remarque 4.2. La notation $\underline{k} \not\to$ indique que, quelle que soit l'expression $a \in \mathsf{EA}$,on n'a pas $\underline{k} \to a$. Les constantes ne peuvent pas être exécutées.

Exercice 4.1. Et si on ajoute la règle

$$\frac{a_1 \to a_1' \qquad a_2 \to a_2'}{a_1 \oplus a_2 \to a_1' \oplus a_2'}\ ,$$

appelée *réduction parallèle*, que se passe-t-il ?

Remarque 4.3. Il n'est pas possible de démontrer $\underline{2} \oplus (\underline{3} \oplus \underline{4}) \to \underline{9}$. En effet, on réalise *deux* pas.

4.3 Coïncidence entre grands pas et petits pas.

On définit la clôture réflexive et transitive d'une relation binaire $\mathscr{R}$ sur un ensemble E, notée $\mathscr{R}^\star$. On la définit par les règles d'inférences suivantes :

$$\frac{}{x\ \mathscr{R}^\star\ x} \quad \text{et} \quad \frac{x\ \mathscr{R}\ y \qquad y\ \mathscr{R}^\star\ z}{x\ \mathscr{R}^\star\ z}\ .$$

Lemme 4.1. La relation $\mathscr{R}^\star$ est transitive.

Preuve. On démontre

$$\forall x, y \in E, \quad \text{si } x \mathrel{\mathscr{R}^\star} y \text{ alors } \underbrace{\forall z,\ y \mathrel{\mathscr{R}^\star} z \implies x \mathrel{\mathscr{R}^\star} z}_{\mathscr{P}(x,y)}$$

par induction sur $x \mathrel{\mathscr{R}^\star} y$. Il y a *deux* cas.

- ▷ *Réflexivité.* On a donc $x = y$ et, par hypothèse, $y \mathrel{\mathscr{R}^\star} z$.
- ▷ *Transitivité.* On sait que $x \mathrel{\mathscr{R}} a$ et $a \mathrel{\mathscr{R}^\star} y$. De plus, on a l'hypothèse d'induction

$$\mathscr{P}(a, y) : \forall z, y \mathrel{\mathscr{R}^\star} z \implies a \mathrel{\mathscr{R}^\star} z.$$

Montrons $\mathscr{P}(x, y)$. Soit z tel que $y \mathrel{\mathscr{R}^\star} z$. Il faut donc montrer $x \mathrel{\mathscr{R}^\star} z$. On sait que $x \mathrel{\mathscr{R}} a$ et, par hypothèse d'induction, $a \mathrel{\mathscr{R}^\star} z$. Ceci nous donne $x \mathrel{\mathscr{R}^\star} z$ en appliquant la seconde règle d'inférence.

□

Lemme 4.2. Quelles que soient a_2 et a_2', si $a_2 \to^\star a_2'$, alors pour tout a_1, on a $a_1 \oplus a_2 \to^\star a_1 \oplus a_2'$.

Preuve. On procède par induction sur $a_2 \to^\star a_2'$. Il y a *deux* cas.

1. On a $a_2' = a_2$. Il suffit donc de montrer que l'on a

$$a_1 \oplus a_2 \to^\star a_1 \oplus a_2,$$

ce qui est vrai par réflexivité.

2. On sait que $a_2 \to a$ et $a \to^\star a_2'$. On sait de plus que

$$\forall a_1, \quad a_1 \oplus a \to^\star a_1 \oplus a_2'$$

par hypothèse d'induction. On veut montrer que

$$\forall a_1, \quad a_1 \oplus a_2 \to^\star a_1 \oplus a_2'.$$

On se donne a_1. On déduit de $a_2 \to a$ que $a_1 \oplus a_2 \to a_1 \oplus a$ par $\mathscr{C}_{\mathrm{d}}$. Par hypothèse d'induction, on a $a_1 \oplus a \to^\star a_1 \oplus a_2'$. Par la seconde règle d'inférence, on conclut.

□

Lemme 4.3. Quelles que soient les expressions a_1 et a_1', si $a_1 \to^\star a_1'$ alors, pour tout k, $a_1 \oplus \underline{k} \to^\star a_1' \oplus \underline{k}$. □

Attention, le lemme précédent est faux si l'on remplace $\underline{k}$ par une expression a_2. En effet, a_2 ne peut pas être « spectateur » du calcul de a_1.

Proposition 4.1. Soient a une expression et k un entier. On a l'implication

$$a \Downarrow k \implies a \to^\star \underline{k}.$$

Preuve. On le démontre par induction sur la relation $a \Downarrow k$. Il y a *deux* cas.

1. Dans le cas $a = \underline{k}$, alors on a bien $\underline{k} \to^\star \underline{k}$.
2. On sait que $a_1 \Downarrow k_1$ et $a_2 \Downarrow k_2$, avec $k = k_1 + k_2$. On a également deux hypothèses d'induction :
   - ▷ $(H_1) : a_1 \to^\star \underline{k}_1$;
   - ▷ $(H_2) : a_2 \to^\star \underline{k}_2$.

   On veut montrer $a_1 \oplus a_2 \to^\star \underline{k}$, ce que l'on peut faire par :

$$a_1 \oplus a_2 \xrightarrow{(H_2)+\text{lemme 4.2}}{}^\star a_1 \oplus \underline{k}_2 \xrightarrow{(H_1)+\text{lemme 4.3}}{}^\star \underline{k}_1 \oplus \underline{k}_2 \xrightarrow{\mathscr{A}} \underline{k}.$$

□

Proposition 4.2. Soient a une expression et k un entier. On a l'implication

$$a \to^\star \underline{k} \implies a \Downarrow k.$$

□

4.4 L'ensemble EA avec des erreurs à l'exécution.

On exécute des programmes de EA. On considère que $\underline{k}_1 \oplus \underline{k}_2$ s'évalue comme

$$\frac{(k_1 + k_2) \times k_2}{k_2}.$$

Le cas $k_2 = 0$ est une situation d'erreur, une « ***situation catastrophique*** ». (C'est une convention : quand un ordinateur divise par zéro, il explose !)

4.4.1 Relation à grands pas.

On note encore $\Downarrow$ la relation d'évaluation sur $\mathsf{EA} * \mathbb{Z}_\perp$, où l'on définit l'ensemble $\mathbb{Z}_\perp = \mathbb{Z} \cup \{\perp\}$. Le symbole $\perp$ est utilisé pour représenter un cas d'erreur.

Les règles d'inférences définissant $\Downarrow$ sont :

$$\frac{}{\underline{k} \Downarrow k} \qquad \underset{k \neq 0}{^{k = k_1 + k_2}} \ \frac{a_1 \Downarrow k_1 \qquad a_2 \Downarrow k_2}{a_1 \oplus a_2 \Downarrow k} \qquad \frac{a_1 \Downarrow k_1 \qquad a_2 \Downarrow 0}{a_1 \oplus a_2 \Downarrow \perp},$$

et les règles de propagation du $\perp$:

$$\frac{a_1 \Downarrow \perp \qquad (a_2 \Downarrow r)}{a_1 \oplus a_2 \Downarrow \perp} \qquad \frac{(a_1 \Downarrow r) \qquad a_2 \Downarrow \perp}{a_1 \oplus a_2 \Downarrow \perp}.$$

4.4.2 Relation à petits pas.

On (re)-définit la relation $\to \subseteq \mathsf{EA} * \mathsf{EA}_\perp$, où $\mathsf{EA}_\perp = \mathsf{EA} \cup \{\perp\}$, par les règles d'inférences

$$\underset{k_2 \neq 0}{^{k = k_1 + k_2}} \ \frac{}{\underline{k}_1 \oplus \underline{k}_2 \to \underline{k}} \qquad {}^{a_2 \neq \perp} \ \frac{a_2 \to a_2'}{a_1 \oplus a_2 \to a_1 \oplus a_2'}$$

$$^{a_1 \neq \perp} \ \frac{a_1 \to a_1'}{a_1 \oplus \underline{k} \to a_1' \oplus \underline{k}} \qquad \frac{}{\underline{k}_1 \oplus \underline{0} \to \perp},$$

et les règles de propagation du $\bot$:

$$\frac{a_1 \to \bot}{a_1 \oplus \underline{k} \to \bot} \quad \text{et} \quad \frac{a_2 \to \bot}{a_1 \oplus a_2 \to \bot}.$$

Pour démontrer l'équivalence des relations grand pas et petits pas, ça semble un peu plus compliqué...

4.5 Sémantique contextuelle pour EA.

On définit la relation $\mapsto : \mathsf{EA} \times \mathsf{EA}$ par la règle :

$$\frac{}{E[\underline{k_1} \oplus \underline{k_2}] \mapsto E[\underline{k}],} \; k = k_1 + k_2$$

où E est un *contexte d'évaluation* que l'on peut définir par la grammaire

$$E ::= [\,] \mid \boxed{?}.$$

Le *trou* est une constante, notée $[\,]$ qui n'apparaît qu'une fois par contexte d'évaluation. Pour E un contexte d'évaluation et $a \in \mathsf{EA}$, alors $E[a]$ désigne l'expression arithmétique obtenue en remplaçant le trou par a dans E.

Exemple 4.3. On note $E_0 = \underline{3} \oplus ([\,] \oplus \underline{5})$ et $a_0 = \underline{1} \oplus \underline{2}$. Alors

$$\underline{3} \oplus ((\underline{1} \oplus \underline{2}) \oplus \underline{5}).$$

Que faut-il mettre à la place de ? ?

Exemple 4.4 (Première tentative)**.** On pose

$$E ::= [\,] \mid \underline{k} \mid E_1 \oplus E_2.$$

Mais, ceci peut introduire *plusieurs* trous (voire aucun) dans un même contexte. C'est raté.

Exemple 4.5 (Seconde tentative)**.** On pose

$$E ::= [\,] \mid a \oplus E \mid E \oplus a.$$

Mais, on pourra réduire une expression à droite avant de réduire à gauche. C'est encore raté.

Exemple 4.6 (Troisième (et dernière) tentative)**.** On pose

$$E ::= [\,] \mid a \oplus E \mid E \oplus \underline{k}.$$

Là, c'est réussi !

Lemme 4.4. Pour toute expression arithmétique $a \in \mathsf{EA}$ qui n'est pas une constante, il existe un unique triplet (E, k_1, k_2) tel que

$$a = E[\underline{k}_1 \oplus \underline{k}_2].$$

Ceci permet de justifier la proposition suivante, notamment au niveau des notations.

Proposition 4.3. Pour tout a, a', on a

$$a \to a' \quad \text{si, et seulement si,} \quad a \mapsto a'.$$

Preuve. Pour démontrer cela, on procède par double implication :

- ▷ « $\Longrightarrow$ » par induction sur $a \to a'$;
- ▷ « $\Longleftarrow$ » par induction sur E.

□

5. Sémantique opérationnelle des expressions arithmétiques avec déclarations locales (LEA).

Sommaire.

On suppose donnés $\mathbb{Z}$ les entiers relatifs et $\mathscr{V}$ un ensemble infini de variables (d'identifiants/d'identificateurs/de noms).

On définit LEA par la grammaire suivante :

$$a ::= \underline{k} \mid a_1 \oplus a_2 \mid \texttt{let}\, x \,\texttt{=}\, a_1 \;\texttt{in}\; a_2 \mid x,$$

où $x \in \mathscr{V}$ et $k \in \mathbb{Z}$.

En Rocq, on peut définir :

```
Inductive LEA : Set :=
| Cst : ℤ → LEA
| Add : LEA → LEA → LEA
| Let : 𝒱 → LEA → LEA → LEA
```

| `Var` : $\mathcal{V} \to$ LEA.

Code 5.1 | *Définition inductive de* LEA

Exemple 5.1. Voici quelques exemples d'expressions avec déclarations locales :

1. $\texttt{let}\, x = 3\ \texttt{in}\ x \oplus x$;
2. $\texttt{let}\, x = \underline{2}\ \texttt{in let}\, y = x \oplus \underline{2}\ \texttt{in}\ x \oplus y$;
3. $\texttt{let}\, x = (\texttt{let}\, y = \underline{5}\ \texttt{in}\ y \oplus y)\ \texttt{in}\ (\texttt{let}\, z = \underline{6}\ \texttt{in}\ z \oplus \underline{2}) \oplus x$;
4. $\texttt{let}\, x = \underline{7} \oplus \underline{2}\ \texttt{in}\ (\texttt{let}\, x = 5\ \texttt{in}\ x \oplus x) \oplus x$.

5.1 Sémantique à grands pas sur LEA.

On définit une relation d'*évaluation* $\Downarrow : \text{LEA} * \mathbb{Z}$ [1] définie par :

$$\frac{}{\underline{k} \Downarrow k} \qquad {\scriptstyle k = k_1 + k_2}\ \frac{a_1 \Downarrow k_1 \qquad a_2 \Downarrow k_2}{a_1 \oplus a_2 \Downarrow k}\ ,$$

et on ajoute une règle pour le $\texttt{let}\, x = \ldots\ \texttt{in}\ \ldots$:

$$\frac{a_1 \Downarrow k_1 \qquad a_2\,[\underline{k_1}/x] \Downarrow k_1}{(\texttt{let}\, x = a_1\ \texttt{in}\ a_2) \Downarrow k_2}.$$

On note ici $a[\underline{k}/x]$ la substitution de $\underline{k}$ à la place de x dans l'expression a. Ceci sera défini après.

Attention : on n'a pas de règles de la forme

les variables sont censées disparaitre avant qu'on arrivent à elles.

Définition 5.1. Soit $a \in$ LEA. L'ensemble des *variables libres* d'une expression a noté $\mathcal{VL}(a)$, et est défini par induction sur a de la manière suivante :

- $\mathcal{Vl}(\underline{k}) = \emptyset$;
- $\mathcal{Vl}(x) = \{x\}$;
- $\mathcal{Vl}(a_1 \oplus a_2) = \mathcal{Vl}(a_1) \cup \mathcal{Vl}(a_2)$;
- $\mathcal{Vl}(\texttt{let}\, x = a_1 \;\texttt{in}\; a_2) = \mathcal{Vl}(a_1) \cup (\mathcal{Vl}(a_2) \setminus \{x\})$.

Exemple 5.2.

$$\mathcal{Vl}(\texttt{let}\, x = \underline{3} \;\texttt{in}\; \texttt{let}\, y = x \oplus \underline{2} \;\texttt{in}\; y \oplus (z \oplus \underline{15})) = \{z\}.$$

Définition 5.2. Une expression $a \in \mathsf{LEA}$ est *close* si $\mathcal{Vl}(a) = \emptyset$. On note $\mathsf{LEA}_0 \subseteq \mathsf{LEA}$ l'ensemble des expressions arithmétiques de closes.

Définition 5.3. Soient $a \in \mathsf{LEA}$, $x \in \mathcal{V}$ et $k \in \mathbb{Z}$. On définit par induction sur a (*quatre cas*) le résultat de la *substitution* de x par $\underline{k}$ dans a, noté $a[\underline{k}/x]$ de la manière suivante :

- $\underline{k}'[\underline{k}/x] = \underline{k}'$;
- $(a_1 \oplus a_2)[\underline{k}/x] = (a_1[\underline{k}/x]) \oplus (a_2[\underline{k}/x])$;
- $y[\underline{k}/x] = \begin{cases} \underline{k} & \text{si } x = y \\ y & \text{si } x \neq y \,; \end{cases}$
- $(\texttt{let}\, y = a_1 \;\texttt{in}\; a_2)[\underline{k}/x] = \begin{cases} \texttt{let}\, y = a_1[\underline{k}/x] \;\texttt{in}\; a_2 & \text{si } x = y \\ \texttt{let}\, y = a_1[\underline{k}/x] \;\texttt{in}\; a_2[\underline{k}/x] & \text{si } x \neq y. \end{cases}$

5.2 Sémantique à petits pas sur LEA.

On définit la relation $\to \subseteq \mathsf{LEA} * \mathsf{LEA}$ inductivement par :

$$\text{[}k = k_1 + k_2\text{]}\ \frac{}{\underline{k_1} \oplus \underline{k_2} \to \underline{k}}\ \mathscr{A},$$

1. On surcharge encore les notations.

$$\frac{a_2 \to a_2'}{a_1 \oplus a_2 \to a_1 \oplus a_2'}\ \mathscr{C}_{\text{d}} \quad \text{et} \quad \frac{a_1 \to a_1'}{a_1 \oplus \underline{k} \to a_1' \oplus \underline{k}}\ \mathscr{C}_{\text{g}},$$

puis les nouvelles règles pour le $\texttt{let}\, x = \ldots\ \texttt{in}\ \ldots$:

$$\frac{a_1 \to a_1'}{\texttt{let}\, x = a_1\ \texttt{in}\ a_2 \to \texttt{let}\, x = a_1'\ \texttt{in}\ a_2}\ \mathscr{C}_{\text{l}}$$

$$\frac{}{\texttt{let}\, x = \underline{k}\ \texttt{in}\ a \to a[\underline{k}/x]}.$$

On peut démontrer l'équivalence des sémantiques à grands pas et à petits pas.

5.3 Sémantique contextuelle pour LEA.

On définit les contextes d'évaluations par la grammaire suivante :

$$\begin{aligned} E ::= {} & [\,] \\ & \mid a \oplus E \\ & \mid E \oplus \underline{k} \\ & \mid \texttt{let}\, x = E\ \texttt{in}\ a. \end{aligned}$$

On définit *deux* relations $\mapsto_{\text{a}}$ et $\mapsto$ par les règles :

$$^{k = k_1 + k_2}\ \frac{}{\underline{k}_1 \oplus \underline{k}_2 \mapsto_{\text{a}} k_2} \qquad \frac{}{\texttt{let}\, x = \underline{k}\ \texttt{in}\ a \mapsto_{\text{a}} a[\underline{k}/x]},$$

et

$$\frac{a \mapsto_{\text{a}} a'}{E[a] \mapsto E[a']}.$$

5.4 Sémantique sur LEA avec environnement.

Définition 5.4. Soient A et B deux ensembles. Un *dictionnaire* sur (A, B) est une fonction partielle à domaine fini de A dans B.

Si D est un dictionnaire sur (A, B), on note $D(x) = y$ lorsque D associe $y \in B$ à $x \in A$.

Le domaine d'un dictionnaire D est

$$\mathrm{dom}(D) = \{x \in A \mid \exists y \in B, D(x) = y\}.$$

On note $\emptyset$ le dictionnaire vide.

Pour un dictionnaire D sur (A, B), deux éléments $x \in A$ et $y \in B$, on note $D[x \mapsto y]$ est le dictionnaire D' défini par

- $D'(x) = y$;
- $D'(z) = D(z)$ pour $z \in \mathrm{dom}(D)$ tel que $z \neq x$.

On ne s'intéresse pas à la construction d'un tel type de donné, mais juste son utilisation.

On se donne un ensemble Env d'*environnements* notés $\mathscr{E}, \mathscr{E}', \ldots$ qui sont des dictionnaires sur $(\mathscr{V}, \mathbb{Z})$.

5.4.1 Sémantique à grands pas sur LEA avec environnements.

On définit la relation $\Downarrow \subseteq \mathsf{LEA} * \mathrm{Env} * \mathbb{Z}$, noté $a, \mathscr{E} \Downarrow k$ (« a s'évalue en k dans $\mathscr{E}$ ») défini par

$$\frac{}{\underline{k}, \mathscr{E} \Downarrow k} \qquad \frac{a_1, \mathscr{E} \Downarrow k_1 \qquad a_2, \mathscr{E} \Downarrow k_2}{a_1 \oplus a_2, \mathscr{E} \Downarrow k}\ {\scriptstyle k = k_1 + k_2} \qquad \frac{}{x, \mathscr{E} \Downarrow k,}\ {\scriptstyle \mathscr{E}(x) = k}$$

$$\frac{a_1, \mathscr{E} \Downarrow k_1 \qquad a_2, \mathscr{E}[x \mapsto k_1] \Downarrow k_2}{\texttt{let}\, x = a_1\ \texttt{in}\ a_2 \Downarrow k_2}\ .$$

Remarque 5.1.
- Dans cette définition, on n'a pas de substitutions (c'est donc plus facile à calculer).
- Si $\mathscr{V}\ell(a) \subseteq \mathrm{dom}(\mathscr{E})$, alors il existe $k \in \mathbb{Z}$ tel que $a, \mathscr{E} \Downarrow k$.
- On a $a \Downarrow k$ (sans environnement) si, et seulement si $a, \emptyset \Downarrow k$ (avec environnement).

Pour les petits pas avec environnements, c'est un peu plus compliqué...On verra ça en TD. (Écraser les valeurs dans un dictionnaire, ça peut être problématique avec les petits pas.)

6. Un petit langage fonctionnel : FUN.

Sommaire.

On se rapproche de notre but final en considérant un petit langage fonctionnel, nommé FUN.

On se donne l'ensemble des entiers relatifs $\mathbb{Z}$ et un ensemble infini de variables $\mathscr{V}$. L'ensemble des expressions de FUN, notées e, e' ou e_i, est défini par la grammaire suivante :

$$e ::= k \mid e_1 + e_2 \mid \underbrace{\texttt{fun}\, x \to e}_{\text{Fonction / Abstraction}} \mid \overbrace{e_1\ e_2}^{\text{Application}} \mid x.$$

Note 6.1. On simplifie la notation par rapport à EA ou LEA : on ne souligne plus les entiers, on n'entoure plus les plus.

On notera de plus $e_1\ e_2\ e_3$ pour $(e_1\ e_2)\ e_3$. Aussi, l'expression $\texttt{fun}\, x\ y \to e$ représentera l'expression $\texttt{fun}\, x \to (\texttt{fun}\, y \to e)$. On

n'a pas le droit à plusieurs arguments pour une fonction, mais on applique la curryfication.

6.1 Sémantique opérationnelle « informellement ».

Exemple 6.1. Comment s'évalue $(\texttt{fun}\, x \to x + x)(7 + 7)$?

- ▷ D'une part, $7 + 7$ s'évalue en 14.
- ▷ D'autre part, $(\texttt{fun}\, x \to x + x)$ s'évalue en elle même.
- ▷ On procède à une substitution de $(x + x)[^{14}/_{x}]$ qui s'évalue en 28.

Exemple 6.2. Comment s'évalue l'expression

$$(\overbrace{(\texttt{fun}\, f \to \underbrace{(\texttt{fun}\, x \to x + (f\ x))}_{B})}^{A}\ \underbrace{(\texttt{fun}\, y \to y + y)}_{C})\ 7\ ?$$

On commence par évaluer A et C qui s'évaluent en A et C respectivement. On continue en calculant la substitution

$$(\texttt{fun}\, x \to x + (f\ x))[^{\texttt{fun}\, y \to y + y}/_{f}],$$

ce qui donne

$$(\texttt{fun}\, x \to x + ((\texttt{fun}\, y \to y + y)\ x)).$$

Là, on **ne simplifie pas**, car c'est du code *dans* une fonction. On calcule ensuite la substitution

$$(x + ((\texttt{fun}\, y \to y + y)\ x))[^{7}/_{x}],$$

ce qui donne

$$7 + ((\texttt{fun}\, y \to y + y)\ 7).$$

On termine par la substitution

$$(y + y)[^7/_y] = 7 + 7.$$

On conclut que l'expression originelle s'évalue en 21.

Remarque 6.1. Dans FUN, le résultat d'un calcul (qu'on appellera *valeur*) n'est plus forcément un entier, ça peut aussi être une fonction.

L'ensemble des valeurs, notées v, est défini par la grammaire

$$v ::= k \mid \texttt{fun}\, x \to e.$$

Les fonctions sont des valeurs ! Et, le « contenu » la fonction n'est pas forcément une valeur.

On peut remarquer que l'ensemble des valeurs est un sous-ensemble des expressions de FUN.

6.2 Sémantique opérationnelle de FUN (version 1).

Définition 6.1. On définit l'ensemble des *variables libres* $\mathcal{VL}(e)$ d'une expression e par (on a 5 cas) :

- $\mathcal{VL}(x) = \{x\}$;
- $\mathcal{VL}(k) = \emptyset$;
- $\mathcal{VL}(e_1 + e_2) = \mathcal{VL}(e_1) \cup \mathcal{VL}(e_2)$;
- $\mathcal{VL}(e_1\ e_2) = \mathcal{VL}(e_1) \cup \mathcal{VL}(e_2)$;
- $\mathcal{VL}(\texttt{fun}\, x \to e) = \mathcal{VL}(e) \setminus \{x\}$. [1]

On dit que e est *close* si $\mathcal{VL}(e) = \emptyset$.

1. L'expression $\texttt{fun}\, x \to e$ est un *lieur* : x est liée dans e.

Définition 6.2. Pour $e \in \mathsf{FUN}$, $x \in \mathscr{V}$ et v une valeur **close**, on définit la *substitution* $e[v/x]$ de x par v dans e par :

- $k[v/x] = k$;
- $y[v/x] = \begin{cases} v & \text{si } x = y \\ y & \text{si } x \neq y \end{cases}$;
- $(\texttt{fun}\, y \to e)[v/x] = \begin{cases} \texttt{fun}\, y \to e & \text{si } x = y \\ \texttt{fun}\, y \to e[e/x] & \text{si } x \neq y; \end{cases}$
- $(e_1 + e_2)[v/x] = (e_1[v/x]) + (e_2[v/x])$;
- $(e_1\ e_2)[v/x] = (e_1[v/x])\ (e_2[v/x])$.

6.2.1 Grands pas pour FUN.

On définit la relation $\Downarrow$ sur couples (expression, valeur) par :

$$\text{\scriptsize } k = k_1 + k_2 \ \frac{e_1 \Downarrow k_1 \qquad e_2 \Downarrow k_2}{e_1 + e_2 \Downarrow k} \qquad \frac{}{v \Downarrow v}$$

$$\frac{e_1 \Downarrow \texttt{fun}\, x \to e \qquad e_2 \Downarrow v_2 \qquad e[v_2/x] \Downarrow v}{e_1\ e_2 \Downarrow v.}$$

Remarque 6.2. Certaines expressions ne s'évaluent pas :

$$x \not\Downarrow \qquad \text{et} \qquad z + (\texttt{fun}\, x \to x) \not\Downarrow$$

par exemple.

6.2.2 Petits pas pour FUN.

On définit la relation $\to \subseteq \mathsf{FUN} * \mathsf{FUN}$ par :

$$\text{\scriptsize } k = k_1 + k_2 \ \frac{}{k_1 + k_2 \to k}\ \mathscr{R}_{\text{pk}} \qquad \frac{}{(\texttt{fun}\, x \to e)\ v \to e[v/x]}\ \mathscr{R}_{\beta}$$

$$\frac{e_2 \to e_2'}{e_1 + e_2 \to e_1 + e_2'}\ \mathscr{R}_{\text{pd}} \qquad \frac{e_1 \to e_1'}{e_1 + k \to e_1' + k}\ \mathscr{R}_{\text{pg}}$$

$$\frac{e_2 \to e_2'}{e_1\ e_2 \to e_1\ e_2'}\ \mathscr{R}_{\text{ad}} \qquad \frac{e_1 \to e_1'}{e_1\ v \to e_1'\ v}\ \mathscr{R}_{\text{ag}}.$$

Remarque 6.3. Il existe des expressions que l'on ne peut pas réduire :

1. $k \not\to$;
2. $(\texttt{fun}\, x \texttt{ -> } x) \not\to$;
3. $e_1 + (\texttt{fun}\, x \texttt{ -> } x) \not\to$;
4. $3\ (5+7) \to 3\ 12 \not\to$.

Dans les cas 1. et 2., c'est cohérent : on ne peut pas réduire des valeurs.

Lemme 6.1. On a

$$e \Downarrow v \qquad \text{si, et seulement si,} \qquad e \to^\star v.$$

Remarque 6.4. Soit $e_0 = (\texttt{fun}\, x \texttt{ -> } x\ x)\ (\texttt{fun}\, x \texttt{ -> } x\ x)$. On remarque que $e_0 \to e_0$.

En FUN, il y a des divergences : il existe $(e_n)_{n\in\mathbb{N}}$ telle que l'on ait $e_n \to e_{n+1}$.

La fonction[2] définie par $\Downarrow$ est donc partielle.

Remarque 6.5 (Problème avec la substitution)**.** On a la chaîne de réductions :

$$\begin{aligned} & ((\texttt{fun}\, y \texttt{ -> } (\texttt{fun}\, x \texttt{ -> } x + y))\ (x+7))\ 5 \\ (\star) \qquad & \to (\texttt{fun}\, x \texttt{ -> } x + (x+7))\ 5 \\ & \to 5 + (5+7) \\ & \to^\star 17. \end{aligned}$$

Attention ! Ici, on a triché : on a substitué avec l'expression $x+7$

mais ce n'est pas une valeur (dans la réduction $(\star)$) !

Mais, on a la chaîne de réductions

$$\begin{aligned}&(\texttt{fun}\, f \to (\texttt{fun}\, x \to (f\ 3) + x))\ (\texttt{fun}\, t \to x + 7)\ 5\\ \to\ &(\texttt{fun}\, x \to ((\texttt{fun}\, t \to x + 7)\ 3) + x)\ 5\\ \to\ &(\texttt{fun}\, x \to ((\texttt{fun}\, t \to x + 7)\ 3) + x)\ 5.\end{aligned}$$

Et là, c'est le drame, on a **capturé la variable libre**. D'où l'hypothèse de v close dans la substitution.

Remarque 6.6. Les relations $\Downarrow$ et $\to$ sont définies sur des expressions **closes**. Et on a même $\to \subseteq \mathsf{FUN}_0 * \mathsf{FUN}_0$.[3]

Lemme 6.2. ▷ Si v est close et si $x \notin \mathscr{V}\ell(e)$ alors $e[^v/_x] = e$.
▷ Si v est close, $\mathscr{V}\ell(e[^v/_x]) = \mathscr{V}\ell(e) \setminus \{x\}$. □

Lemme 6.3. Si $e \in \mathsf{FUN}_0$ et $e \to e'$ alors $e' \in \mathsf{FUN}_0$.

Preuve. Montrons que, quelles que soient e et e', on a : si $e \to e'$ alors $(e \in \mathsf{FUN}_0) \implies (e' \in \mathsf{FUN}_0)$ On procède par induction sur la relation $e \to e'$. Il y a 6 cas :

1. Pour $\mathscr{R}_\beta$, on suppose $(\texttt{fun}\, x \to e)\ v$ est close, alors
   ▷ $(\texttt{fun}\, x \to e)$ est close ;
   ▷ v est close.

   On sait donc que $\mathscr{V}\ell(e) \subseteq \{x\}$, d'où par le lemme précédent, $\mathscr{V}\ell(e[^v/_x]) = \emptyset$ et donc $e[^v/_x]$ est close.
2–6. Pour les autres cas, on procède de la même manière.

□

2. Pour indiquer cela, il faudrait démontrer que la relation $\Downarrow$ est déterministe.

3. Il faudrait ici justifier que la réduction d'une formule close est close. C'est ce que nous allons justifier.

Remarque 6.7. De même, si $e \Downarrow v$ où e est close, alors v est close.

Les relations $\Downarrow$ et $\to$ sont définies sur les expressions et les valeurs closes.

Définition 6.3 (Définition informelle de l'α-conversion)**.** On définit l'α-conversion, notée $e =_\alpha e'$: on a $\texttt{fun}\, x \to e =_\alpha \texttt{fun}\, y \to e'$ si, et seulement si, e' s'obtient en replaçant x par y dans e à condition que $y \notin \mathcal{VL}(e)$. [4]

On étend $e =_\alpha e'$ à toutes les expressions : « on peut faire ça partout ».

Exemple 6.3 (*Les variables liées sont muettes.*)**.** On a :

$$\begin{aligned} \texttt{fun}\, x \to x + z &=_\alpha \texttt{fun}\, y \to y + z \\ &=_\alpha \texttt{fun}\, t \to t + z \\ &\neq_\alpha \texttt{fun}\, z \to z + z. \end{aligned}$$

L'intuition est, quand on a $\mathsf{fun}\ x \to e$ et qu'on a besoin de renommer la variable x, pour cela on prend $x' \notin \mathcal{VL}(e)$.

"Lemme" 6.1. Si $E_0 \subseteq \mathcal{V}$ est un ensemble fini de variables, alors il existe $z \notin E_0$ et $e' \in \mathsf{FUN}$ tel que $\texttt{fun}\, x \to e =_\alpha \texttt{fun}\, z \to e'$. □

Remarque 6.8 (**Fondamental**)**.** En fait FUN désigne l'ensemble des expressions décrites par la grammaire initiale *quotientée* par α-conversion.

Remarque 6.9. On remarque que

$$(e =_\alpha e') \implies \mathcal{VL}(e) = \mathcal{VL}(e').$$

4. C'est une « variable fraîche ».

D'après le "lemme", on peut améliorer notre définition de la substitution.

Définition 6.4. Pour $e \in \mathsf{FUN}$, $x \in \mathscr{V}$ et v une valeur **close**, on définit la *substitution* $e[^v/_x]$ de x par v dans e par :

- $k[^v/_x] = k$;
- $y[^v/_x] = \begin{cases} v & \text{si } x = y \\ y & \text{si } x \neq y \end{cases}$;
- $(\texttt{fun}\, x \texttt{ -> } e)[^v/_x] = (\texttt{fun}\, y \texttt{ -> } e)[^v/_x]$ lorsque $x \neq y$;
- $(e_1 + e_2)[^v/_x] = (e_1[^v/_x]) + (e_2[^v/_x])$;
- $(e_1\ e_2)[^v/_x] = (e_1[^v/_x])\ (e_2[^v/_x])$.

6.3 Ajout des déclarations locales (FUN+`let`).

On ajoute les déclarations locales (comme pour $\mathsf{EA} \to \mathsf{LEA}$) à notre petit langage fonctionnel. Dans la grammaire des expressions de FUN, on ajoute :

$$e ::= \cdots \mid \texttt{let}\, x \texttt{ = } e_1 \texttt{ in } e_2.$$

Ceci implique d'ajouter quelques éléments aux différentes opérations sur les expressions définies ci-avant :

- on définit $\mathscr{V}\ell(\texttt{let}\, x \texttt{ = } e_1 \texttt{ in } e_2) = \mathscr{V}\ell(e_1) \cup (\mathscr{V}\ell(e_2) \setminus \{x\})$;
- on ne change pas les valeurs : une déclaration locale n'est pas une valeur ;
- on ajoute $\texttt{let}\, x \texttt{ = } e_1 \texttt{ in } e_2 =_\alpha \texttt{let}\, y \texttt{ = } e_1 \texttt{ in } e_2'$, où l'on remplace x par y dans e_2 pour obtenir e_2' ;
- pour la substitution, on pose lorsque $x \neq y$ (que l'on peut toujours supposer modulo α-conversion)

$$(\texttt{let}\, y \texttt{ = } e_1 \texttt{ in } e_2)[^v/_x] = (\texttt{let}\, y \texttt{ = } e_1[^v/_x] \texttt{ in } e_2[^v/_x]).$$

- pour la sémantique à grands pas, c'est comme pour LEA ;

- pour la sémantique à petits pas, on ajoute les deux règles :

$$\frac{}{\texttt{let}\, x = v \;\texttt{in}\; e_2 \to e_2[v/x]}\ \mathscr{R}_{\text{lv}}$$

et

$$\frac{e_1 \to e_1'}{\texttt{let}\ x = e_1\ \texttt{in}\ e_2 \to \texttt{let}\ x = e_1'\ \texttt{in}\ e_2}\ \mathscr{R}_{\text{lg}}.$$

Attention ! On n'a pas de règle

$$\frac{e_2 \to e_2'}{\texttt{let}\ x = e_1\ \texttt{in}\ e_2 \to \texttt{let}\ x = e_1\ \texttt{in}\ e_2'}\ \mathscr{R}_{\text{ld}},$$

on réduit d'abord l'expression e_1 jusqu'à une valeur, avant de passer à e_2.

Le langage que l'on construit s'appelle FUN + `let`.

6.3.1 Traduction de FUN + `let` vers FUN.

On définit une fonction qui, à toute expression de e dans FUN + `let` associe une expression notée $[\![e]\!]$ dans FUN (on supprime les expressions locales). L'expression $[\![e]\!]$ est définie par induction sur e. Il y a 6 cas :

- $[\![k]\!] = k$;
- $[\![x]\!] = x$;
- $[\![e_1 + e_2]\!] = [\![e]\!]_1 + [\![e]\!]_2$;
- $[\![e_1\ e_2]\!] = [\![e]\!]_1\ [\![e]\!]_2$;
- $[\![\texttt{fun}\, x \to e]\!] = \texttt{fun}\, x \to [\![e]\!]$;
- $[\![\texttt{let}\, x = e_1\ \texttt{in}\ e_2]\!] = (\texttt{fun}\, x \to [\![e_2]\!])\ [\![e_1]\!]$.

Lemme 6.4. Pour tout $e \in$ (FUN + `let`),

- $[\![e]\!]$ est une expression de FUN [5] ;
- on a $\mathscr{V}\ell([\![e]\!]) = \mathscr{V}\ell(e)$;
- $[\![e]\!]$ est une valeur *ssi* e est une valeur ;
- $[\![e[v/x]]\!] = [\![e]\!]\,[[\![v]\!]/x]$ [6]. □

Pour démontrer le lemme 6.4, on procède par induction sur e. C'est long et rébarbatif, mais la proposition ci-dessous est bien plus intéressante.

Proposition 6.1. Pour toutes expressions e, e' de $\mathsf{FUN} + \texttt{let}$, si on a la réduction $e \to_{\mathsf{FUN}+\texttt{let}} e'$ alors $[\![e]\!] \to_{\mathsf{FUN}} [\![e']\!]$.

Preuve. On procède par induction sur $e \to e'$ dans $\mathsf{FUN} + \texttt{let}$. Il y a 8 cas car il y a 8 règles d'inférences pour $\to$ dans $\mathsf{FUN} + \texttt{let}$.

- ▷ *Cas* $\mathscr{R}_{\text{lv}}$. Il faut montrer que $[\![\texttt{let}\, x = v\ \texttt{in}\ e_2]\!] \to_{\mathsf{FUN}} [\![e[v/x]]\!]$. Par définition, l'expression de droite vaut

$$(\texttt{fun}\, x \to [\![e]\!]_2)\ [\![v]\!] \xrightarrow{\mathscr{R}_\beta}_{\mathsf{FUN}} [\![e]\!]_2\, [[\![v]\!]/x],$$

  car $[\![v]\!]$ est une valeur par le lemme 6.4, ce qui justifie $\mathscr{R}_\beta$. De plus, encore par le lemme 6.4, on a l'égalité entre $[\![e]\!]_2\, [[\![v]\!]/x] = [\![e[v/x]]\!]$.
- ▷ *Cas* $\mathscr{R}_{\text{lg}}$. On sait que $e_1 \to e_1'$ et, par hypothèse d'induction, on a $[\![e_1]\!] \to [\![e_1']\!]$. Il faut montrer que

$$[\![\texttt{let}\, x = e_1\ \texttt{in}\ e_2]\!] \to [\![\texttt{let}\, x = e_1'\ \texttt{in}\ e_2]\!]\,.$$

  L'expression de droite vaut

$$(\texttt{fun}\, x \to [\![e_2]\!])\ [\![e_1]\!] \xrightarrow{\mathscr{R}_{\text{ad}}\ \&\ \text{hyp. ind.}} (\texttt{fun}\, x \to [\![e_2]\!])\ [\![e_1']\!]\,.$$

  Et, par définition de $[\![\cdot]\!]$, on a l'égalité :

$$[\![\texttt{let}\, x = e_1'\ \texttt{in}\ e_2]\!] = (\texttt{fun}\, x \to [\![e_2]\!])\ [\![e_1']\!]\,.$$

- ▷ Les autres cas sont laissées en exercice.

□

5. *i.e.* $[\![e]\!]$ n'a pas de déclarations locales
6. On le prouve par induction sur e, c'est une induction à 6 cas

Proposition 6.2. Si $[\![e]\!] \to [\![e']\!]$ alors $e \to e'$.

Preuve. La proposition ci-dessus est mal formulée pour être prouvée par induction, on la ré-écrit. On démontre, par induction sur la relation $f \to f'$ la propriété suivante :

> « quel que soit e, si $f = [\![e]\!]$ alors il existe e' une expression telle que $f' = [\![e']\!]$ et $e \to e'$ (dans FUN + `let`) »,

qu'on notera $\mathcal{P}(f, f')$.

Pour l'induction sur $f \to f'$, il y a 6 cas.

- ▷ *Cas de la règle* $\mathcal{R}_{\text{ad}}$. On suppose $f_2 \to f_2'$ et par hypothèse d'induction $\mathcal{P}(f_2, f_2')$. On doit montrer $\mathcal{P}(f_1\ f_2, f_1\ f_2')$. On suppose donc $[\![e]\!] = f_1\ f_2$. On a deux sous-cas.
  - *1er sous-cas.* On suppose $e = e_1\ e_2$ et $[\![e_1]\!] = f_1 = f_2$. Par hypothèse d'induction et puisque $[\![e_2]\!] = f_2$, il existe e_2' tel que $e_2 \to e_2'$ et $[\![e_2']\!] = f_2'$. De $e_2 \to e_2'$, on en déduit par $\mathcal{R}_{\text{ad}}$ que $e_1\ e_2 \to e_1\ e_2'$. On pose $e' = e_1\ e_2'$ et on a bien $[\![e']\!] = [\![e_1]\!]\ [\![e_2']\!]$.
  - *2ème sous-cas.* On suppose $e = \texttt{let}\ x\ \texttt{=}\ e_1\ \texttt{in}\ e_2$. Alors,
    $$[\![e]\!] = \underbrace{(\texttt{fun}\ x \to [\![e_2]\!])}_{f_1}\ \underbrace{[\![e_1]\!]}_{f_2}.$$
    Par hypothèse d'induction, il existe e_1' tel que $e_1 \to e_1'$ et $[\![e_1']\!] = f_2'$. Posons $e' = (\texttt{let}\ x\ \texttt{=}\ e_1'\ \texttt{in}\ e_2)$. On doit vérifier $[\![e]\!] \to [\![e']\!]$ ce qui est vrai par $\mathcal{R}_{\text{ad}}$ et que $[\![e']\!] = f_1\ f_2'$, ce qui est vrai par définition.
- ▷ *Cas de la règle* $\mathcal{R}_{\text{ag}}$. On suppose $f_1 \to f_1'$ et l'hypothèse d'induction $\mathcal{P}(f_1, f_1')$. On doit vérifier que $\mathcal{P}(f_1\ v, f_1'\ v)$. On suppose $[\![e]\!] = f_1\ v$ et on a deux sous-cas.
  - *1er sous-cas.* On suppose $e = e_1\ e_2$ et alors $[\![e]\!] = [\![e_1]\!]\ [\![e_2]\!]$ par le lemme 6.4 et parce que e_2 est une

valeur (car $[\![e_2]\!] = v$). On raisonne comme pour la règle $\mathscr{R}_{\text{ad}}$ dans le premier sous-cas, en appliquant $\mathscr{R}_{\text{ag}}$.

– *2nd sous-cas.* On suppose $e = (\texttt{let}\, x = e_1 \;\texttt{in}\; e_2)$ alors

$$[\![e]\!] = \underbrace{\texttt{fun}\, x \to [\![e_2]\!]}_{f_1}\ \underbrace{[\![e_1]\!]}_{f_2}.$$

On vérifie aisément ce que l'on doit montrer.

▷ Les autres cas se font de la même manière (attention à $\mathscr{R}_\beta$).

□

7. Typage en FUN.

Sommaire.

7.1 Définition du système de types.

L'ensemble Typ des types, notés $\tau, \tau_1, \tau', \ldots$, est définit par la grammaire suivante :

$$\tau ::= \texttt{int} \mid \tau_1 \to \tau_2.$$

Note 7.1. Attention ! Le type $\tau_1 \to \tau_2 \to \tau_3$ n'est pas égal au type $(\tau_1 \to \tau_2) \to \tau_3$. En effet, dans le premier cas, c'est une fonction qui renvoie une fonction ; et, dans le second cas, c'est une fonction qui prend une fonction.

Définition 7.1. Un *environnent de typage*, noté $\Gamma, \Gamma_1, \Gamma', \ldots$, est un dictionnaire sur $(\mathcal{V}, \mathsf{Typ})$, où Typ est l'ensemble des types.

Une *hypothèse de typage*, notée $x : \tau$, est un couple (x, τ).

On note $\Gamma, x : \tau$ l'extension de Γ avec l'hypothèse de typage $x : \tau$ qui n'est définie que lorsque $x \notin \operatorname{dom} \Gamma$. [1]

Remarque 7.1. On peut voir/implémenter Γ comme des listes finies de couples (x, τ).

Définition 7.2. La *relation de typage*, notée $\Gamma \vdash e : \tau$ (« sous les hypothèses Γ, l'expression e a le type τ ») est définie par les règles d'inférences suivantes.

$$\frac{}{\Gamma \vdash k : \texttt{int}}\ \mathcal{T}_{\text{k}} \qquad \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ \mathcal{T}_{\text{v}} \qquad \frac{\Gamma, x : \tau_1 \text{ [2]} \vdash e_2}{\Gamma \vdash \texttt{fun}\, x \texttt{ -> } e : \tau_1 \to \tau_2}\ \mathcal{T}_{\text{f}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 + e_2 : \texttt{int}}\ \mathcal{T}_{\text{p}} \qquad \frac{\Gamma \vdash e : \tau_1 \to \tau_2 \qquad \Gamma \vdash e' : \tau_1}{\Gamma \vdash e\ e' : \tau_2}\ \mathcal{T}_{\text{a}}.$$

Remarque 7.2. Pour l'instant, on parle uniquement d'expressions et pas du tout de valeurs ou de sémantique opérationnelle.

Remarque 7.3.
1. On dit que e est *typable* s'il existe Γ et τ tel que $\Gamma \vdash e : \tau$.
2. Il y a une règle de typage par construction du langage des expressions.

Exemple 7.1. 1. L'expression $\texttt{fun}\, x \texttt{ -> } x$ est particulière : on

1. La définition de $\Gamma, x : \tau$ est « comme on le pense ».
2. On peut toujours étendre Γ ainsi, modulo α-conversion.

peut la typer avec $\tau \to \tau$ quel que soit τ. Par exemple,

$$\dfrac{\dfrac{}{x : \texttt{int} \vdash x : \texttt{int}}\ \mathcal{T}_{\text{v}}}{\emptyset \vdash \texttt{fun}\, x \to x : \texttt{int} \to \texttt{int}}\ \mathcal{T}_{\text{f}}.$$

On aurait pu faire de même avec le type $(\texttt{int} \to \texttt{int}) \to (\texttt{int} \to \texttt{int})$.

2. Quel est le type de $\texttt{fun}\, g \to g\ (g\ 7)$?

$$\dfrac{\dfrac{\dfrac{}{g : \texttt{int} \to \texttt{int} \vdash g : \texttt{int} \to \texttt{int}}\ \mathcal{T}_{\text{v}} \quad \dfrac{\dfrac{}{\Gamma \vdash g : \texttt{int} \to \texttt{int}}\ \mathcal{T}_{\text{v}} \quad \dfrac{}{\Gamma \vdash 7 : \texttt{int}}\ \mathcal{T}_{\text{k}}}{g : \texttt{int} \to \texttt{int} \vdash g\ 7 : \texttt{int}}\ \mathcal{T}_{\text{p}}}{g : \texttt{int} \to \texttt{int} \vdash g\ (g\ 7)}\ \mathcal{T}_{\text{a}}}{\emptyset \vdash \texttt{fun}\, g \to g\ (g\ 7) : (\texttt{int} \to \texttt{int}) \to \texttt{int}}\ \mathcal{T}_{\text{f}}.$$

7.2 Propriétés du système de types.

Lemme 7.1. ▷ Si $\Gamma \vdash e : \tau$ alors $\mathcal{V}\ell(e) \subseteq \text{dom}(\Gamma)$.

▷ *Affaiblissement.* Si $\Gamma \vdash e : \tau$ alors

$$\forall x \notin \text{dom}(\Gamma),\ \forall \tau_0, \quad \Gamma, x : \tau_0 \vdash e : \tau.$$

▷ *Renforcement.* Si $\Gamma, x : \tau_0 \vdash e : \tau$, et si $x \notin \mathcal{V}\ell(e)$ alors on a le typage $\Gamma \vdash e : \tau$.

Preuve. Par induction sur la relation de typage (5 cas). □

7.2.1 Propriété de progrès.

Lemme 7.2. 1. Si $\emptyset \vdash e : \texttt{int}$ et $e \not\to$ alors, il existe $k \in \mathbb{Z}$ tel que $e = k$.

2. Si $\emptyset \vdash e : \tau_1 \to \tau_2$ et $e \not\to$ alors il existe x et e_0 tels que l'on ait $e = \texttt{fun}\, x \to e_0$.

Preuve. Vu en TD. □

Proposition 7.1 (Propriété de progrès). Si $\emptyset \vdash e : \tau$ alors on a la disjonction :

1. soit e est une valeur ;
2. soit il existe e' telle que $e \to e'$.

Remarque 7.4.

- ▷ Si $\emptyset \vdash e_1\ e_2 : \tau$ alors il existe e' tel que $e_1\ e_2 \to e'$.
- ▷ Si $\emptyset \vdash e_1 + e_2 : \tau$ alors il existe e' tel que $e_1 + e_2 \to e'$.

Remarque 7.5. Par le typage, on a exclu les expressions bloquées car « mal formées » (*e.g.* 3 2 ou $3 + (\texttt{fun}\, x \to x)$).

7.2.2 Propriété de préservation.

Cette propriété a plusieurs noms : préservation du typage, réduction assujettie, *subject reduction*.

Lemme 7.3 (typage et substitution). Si l'on a le typage $\emptyset \vdash v : \tau_0$ et $\Gamma, x : \tau_0 \vdash e : \tau$ alors on a $\Gamma \vdash e[v/x] : \tau$

Preuve. On prouve cette propriété par induction sur e. Il y a 5 cas.

- ▷ *Cas* $e = y$. On a deux sous-cas.
  - *1er sous-cas* $x \neq y$. Dans ce cas, $e[v/x] = y$. Il faut montrer $\Gamma \vdash y : \tau$ sachant que $\Gamma, x : \tau_0 \vdash y : \tau$. On applique le lemme de renforcement.
  - *2nd sous-cas* $x = y$. Dans ce cas, $e[v/x] = v$. Il faut montrer que $\Gamma \vdash v : \tau$. Or, on sait que $\Gamma, x : \tau_0 \vdash x : \tau$ (d'où $\tau = \tau_0$) et $\emptyset \vdash v : \tau_0$. On conclut par affaiblissement.
- ▷ Les autres cas sont en exercice.

□

Proposition 7.2 (Préservation du typage)**.** Si $\emptyset \vdash e : \tau$, et $e \to e'$ alors $\emptyset \vdash e' : \tau$.

Preuve. On montre la propriété par induction sur $\emptyset \vdash e : \tau$. Il y a 5 cas.

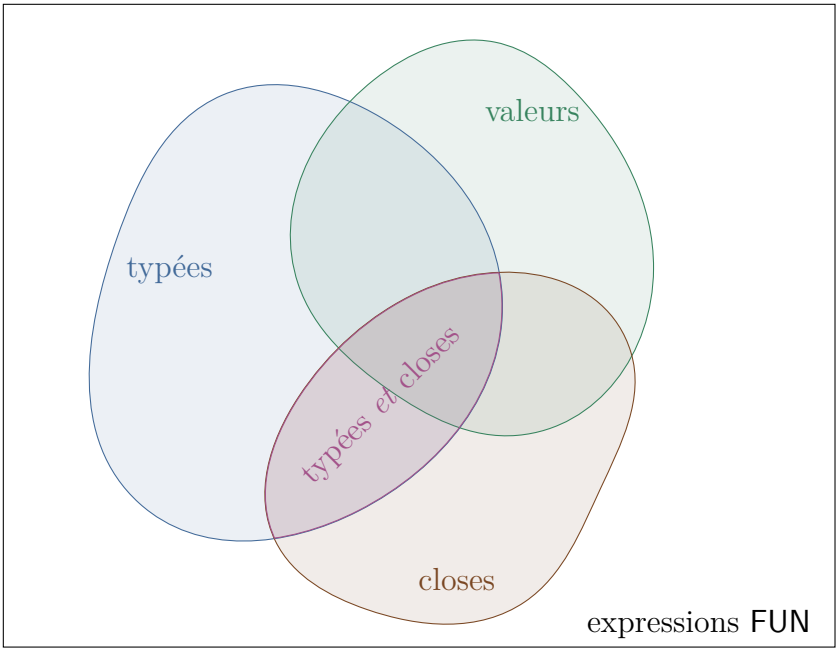
- ▷ *Cas* $\mathcal{T}_{\text{v}}$. C'est absurde ! (On n'a pas $\emptyset \vdash x : \tau$.)
- ▷ *Cas* $\mathcal{T}_{\text{f}}$. Si $(\texttt{fun}\, x \texttt{ → } e) \to e'$ alors ... On peut conclure immédiatement car les fonctions sont des valeurs, elles ne se réduisent donc pas.
- ▷ *Cas* $\mathcal{T}_{\text{k}}$. C'est le même raisonnement.
- ▷ *Cas* $\mathcal{T}_{\text{a}}$. On a $e = e_1\ e_2$. On sait qu'il existe τ_0 un type tel que $\emptyset \vdash e_1 : \tau_0 \to \tau$ (H_1) et $\emptyset \vdash e_2 : \tau_0$ (H_2). On a également les hypothèses d'induction :
  - (H_1') : si $e_1 \to e_1'$ alors $\emptyset \vdash e_1' : \tau_0 \to \tau$;
  - (H_2') : si $e_2 \to e_2'$ alors $\emptyset \vdash e_2' : \tau_0$.

  On doit montrer que si $e_1\ e_2 \to e'$ alors $\emptyset \vdash e' : \tau$. Supposons que $e_1\ e_2 \to e'$, il y a 3 sous-cas.
  - *Sous-cas* $\mathcal{R}_{\text{ad}}$. Cela veut dire que $e_2 \to e_2'$ et $e' = e_1\ e_2'$. On conclut $\emptyset \vdash e_1\ e_2' : \tau$ par (H_2') et (H_1).
  - *Sous-cas* $\mathcal{R}_{\text{ag}}$. Cela veut dire que $e_1 \to e_1'$ et $e' = e_1'\ e_2$. On conclut $\emptyset \vdash e_1'\ e_2 : \tau$ par (H_1') et (H_2).
  - *Sous-cas* $\mathcal{R}_{\beta}$. On a $e_1 = \texttt{fun}\, x \texttt{ → } e_0$, $e_2 = v$ et finalement $e' = e_0[v/x]$. On doit montrer $\emptyset \vdash e_0[v/x] : \tau$. De plus, (H_1) s'énonce par $\emptyset \vdash \texttt{fun}\, x \texttt{ → } e_0 : \tau_0 \to \tau$. Nécessairement (c'est un « `inversion` » en Rocq), cela provient de $x : \tau_0 \vdash e_0 : \tau$. On en conclut par le lemme de substitution.
- ▷ *Cas* $\mathcal{T}_{\text{p}}$. Laissé en exercice.

□

Remarque 7.6. Avec les propriétés de progrès et préservation implique qu'il n'y a pas de « mauvaises surprises » à l'exécution. On

a, en un sens, nettoyé le langage FUN.



C'est la considération d'un langage *statiquement typé*. On aime savoir qu'OCaml ou Rust ont, pour la sémantique et le système de types, une propriété de progrès et de préservation.

Exercice 7.1. Trouver e et e' deux expressions telles que $\emptyset : e' : \tau$ et $e \to e'$ mais que l'on ait pas $\emptyset \vdash e : \tau$.

Solution. Il suffit de trouver une valeur non typable e_1, par exemple $\texttt{fun}\, x \to (x\ x)$ ou $\texttt{fun}\, x \to (19\ 27)$, puis de considérer

$$e = (\texttt{fun}\, x \to 3)\ e_1 \to 3.$$

Or, 3 est typable mais e non.

7.3 Questions en lien avec la relation de typage.

- ▷ *Typabilité.* Pour e donné, existe-t-il Γ, τ tels que $\Gamma \vdash e : \tau$?
- ▷ *Vérification/Inférence de types.* Pour Γ et e donnés, existe-t-il τ tel que l'on ait $\Gamma \vdash e : \tau$? (▷ OCaml)
- ▷ *Habitation.* Pour τ donné, existe-t-il e tel que $\emptyset \vdash e : \tau$? (▷ Rocq[3])

7.4 Inférence de types.

7.4.1 Typage et contraintes.

Exemple 7.2. Dans une version étendue de FUN (on se rapproche plus au OCaml), si l'on considère le programme :

```
let rec f x g=
  ...g x ...
  ...if g f then ... else ...
  ...let h = x 7 in ...
```

On remarque que

- ▷ x et f ont le même type ;
- ▷ g a un type $? \to \texttt{bool}$;
- ▷ x a un type $\texttt{int} \to ?$.

On doit donc lire le programme, et « prendre des notes ». Ces « notes » sont des contraintes que doivent vérifier le programme.

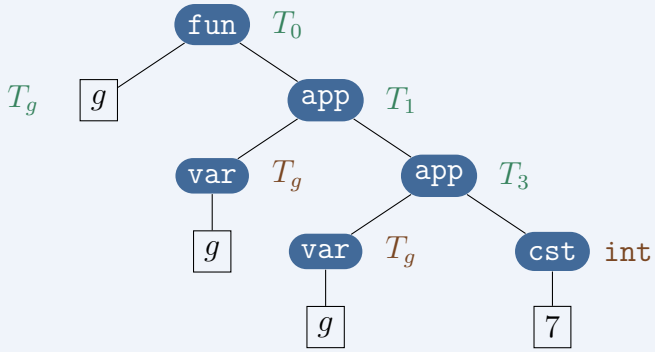
Exemple 7.3. On souhaite déterminer le type τ tel que

$$\emptyset \vdash \texttt{fun}\, g \to g\ (g\ 7) : \tau.$$

3. On peut voir une preuve d'un théorème en Rocq comme fournir une preuve qu'il existe une expression e avec type τ.

(On sait que $\tau = (\texttt{int} \to \texttt{int}) \to \texttt{int}$.)

On construit l'arbre de l'expression (l'AST) :



On procède en plusieurs étapes :

1. On ajoute des inconnues de types T_1, T_2 T_3, *etc* (en vert).
2. On écrit des contraintes faisant intervenir les T_i (en orange/marron).

$$T_0 = T_g \to T_1$$
$$T_g = T_2 \to T_1$$
$$T_g = \texttt{int} \to T_1.$$

3. On résout les contraintes pour obtenir

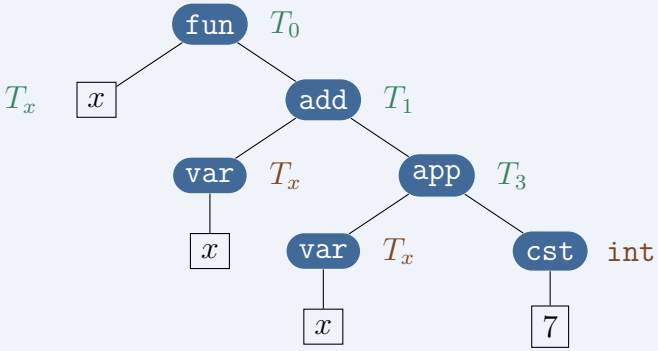
$$T_0 = (\texttt{int} \to \texttt{int}) \to \texttt{int}.$$

Exemple 7.4 (Cas limites). ▷ L'expression $\texttt{fun}\, x \to 7$ admet une infinité de types $(T_x \to \texttt{int})$.
▷ L'expression $(\texttt{fun}\, x \to 7)\ (\texttt{fun}\, z \to z)$ a toujours le type $\texttt{int}$ mais admet une infinité de dérivations.

Exemple 7.5 (Et quand ça ne marche pas ?). On essaie d'inférer

le type de l'expression

$$\texttt{fun}\, x \to x + (x\ 2).$$



Les contraintes sont :

$$T_0 = T_x \to T_1$$
$$T_1 = T_x = T_2 = \texttt{int}$$
$$T_x = \texttt{int} \to T_2.$$

Catastrophe ! On ne peut pas résoudre ce système de contraintes (on ne peut pas avoir $T_x = \texttt{int}$ et $T_x = \texttt{int} \to T_2$ en même temps). L'expression n'est donc pas typable.

Définition 7.3.
- ▷ On se donne un ensemble infini IType d'*inconnues de type*, notées T, T_1, T', *etc.*
- ▷ On définit les *types étendus*, notés $\hat{\tau}$, par la grammaire :

$$\hat{\tau} ::= \texttt{int} \mid \hat{\tau}_1 \to \hat{\tau}_2 \mid T.$$

- ▷ L'ensemble des types (*resp.* étendus) est noté Typ (*resp.* $\widehat{\mathsf{Typ}}$).
- ▷ Les environnement de types étendus sont notés $\widehat{\Gamma}$.
- ▷ Ainsi défini, tout τ est un $\hat{\tau}$, tout Γ est un $\widehat{\Gamma}$.

▷ Un $\hat{\tau}$ est dit *constant* s'il ne contient pas d'inconnue de type (*i.e.* si c'est un τ).

Définition 7.4. Une *contrainte de typage* est une paire de types étendus[4], notée $\hat{\tau}_1 \stackrel{?}{=} \hat{\tau}_2$, ou parfois $\hat{\tau}_1 = \hat{\tau}_2$.

On se donne $e \in \mathsf{FUN}$. On suppose que toutes les variables liées de e sont :

- ▷ distinctes deux à deux ;
- ▷ différentes de toutes les variables libres de e.

On se donne $\widehat{\Gamma}$ tel que $\mathscr{V}\ell(e) \subseteq \mathrm{dom}(\widehat{\Gamma})$. On choisit $T \in \mathrm{IType}$.

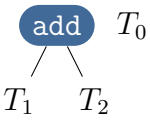
On définit un ensemble de contraintes, notée $\mathsf{CT}(e, \widehat{\Gamma}, T)$ par induction sur e, il y a 5 cas :

- ▷ $\mathsf{CT}(e_1 + e_2, \widehat{\Gamma}, T) = \mathsf{CT}(e_1, \widehat{\Gamma}, T_1) \cup \mathsf{CT}(e_2, \widehat{\Gamma}, T_2) \cup \{T_1 \stackrel{?}{=} \texttt{int}, T_2 \stackrel{?}{=} \texttt{int}, T \stackrel{?}{=} \texttt{int}\}$
- ▷ $\mathsf{CT}(e_1\ e_2, \widehat{\Gamma}, T) = \mathsf{CT}(e_1, \widehat{\Gamma}, T_1) \cup \mathsf{CT}(e_2, \widehat{\Gamma}, T_2) \cup \{T_1 \stackrel{?}{=} T_2 \to T\}$
- ▷ $\mathsf{CT}(x, \widehat{\Gamma}, T) = \{T \stackrel{?}{=} \widehat{\Gamma}(x)\}$
- ▷ $\mathsf{CT}(k, \widehat{\Gamma}, T) = \{T \stackrel{?}{=} \texttt{int}\}$
- ▷ $\mathsf{CT}(\texttt{fun}\, x \to e, \widehat{\Gamma}, T) = \mathsf{CT}(e, (\widehat{\Gamma}, x : T_x), T_2) \cup \{T \stackrel{?}{=} T_1 \to T_2\}$

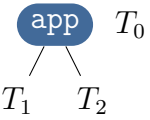
où les variables T_1, T_2, T_x sont *fraîches* (on notera par la suite $\mathit{И}$ T_1, T_2, T_x).

Remarque 7.7. On peut résumer les cas « plus », « application » et « abstraction ».

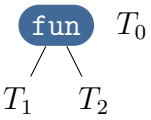
4. ***Attention*** c'est une paire, pas un couple.



$T_0 = T_1 = T_2 = \texttt{int}$



$T_1 = T_2 \to T_0$



$T_0 = T_1 \to T_2$

Définition 7.5. Soit C un ensemble de contraintes de typage. On note $\mathrm{Supp}(C)$, le *support* de C, l'ensemble des inconnues de type mentionnées dans C.

Une *solution* σ de C est un dictionnaire sur $(\mathrm{ITyp}, \widehat{\mathsf{Typ}})$ tel que $\mathrm{dom}(\sigma) \supseteq \mathrm{Supp}(C)$ et que σ égalise toutes les contraintes de C.

Pour $(\hat{\tau}_1 \stackrel{?}{=} \hat{\tau}_2) \in C$, on dit que σ égalise $\hat{\tau}_1 \stackrel{?}{=} \hat{\tau}_2$ signifie que $\sigma(\hat{\tau}_1)$ et $\sigma(\hat{\tau}_2)$ sont le même type étendu.

Il reste à définir $\sigma(\hat{\tau})$, le résultat de l'application de σ à $\hat{\tau}$, par induction sur $\hat{\tau}$, il y a trois cas :

- $\sigma(\hat{\tau}_1 \to \hat{\tau}_2) = \sigma(\hat{\tau}_1) \to \sigma(\hat{\tau}_2)$;
- $\sigma(\texttt{int}) = \texttt{int}$;
- $\sigma(T)$ est le type étendu associé à T dans σ.

Exemple 7.6. Avec $\sigma = [T_1 \mapsto \texttt{int}, T_2 \mapsto (\texttt{int} \to T_3)]$, on a donc

$$\sigma(T_1 \to T_2) = \texttt{int} \to (\texttt{int} \to T_3).$$

Exemple 7.7. La contrainte $T_1 \stackrel{?}{=} T_2 \to T_3$ est égalisée par la solution $\sigma = [T_1 \mapsto T_2 \to \texttt{int}, T_3 \mapsto \texttt{int}]$.

Définition 7.6. Une *solution constante* de C est un dictionnaire sur $(\mathrm{ITyp}, \mathsf{Typ})$ (et pas $(\mathrm{ITyp}, \widehat{\mathsf{Typ}})$) qui est une solution de C.

Proposition 7.3. Soit $e \in \mathsf{FUN}$ et soit Γ tel que $\mathscr{V}\ell(e) \subseteq \mathrm{dom}(\Gamma)$. Soit $T \in \mathrm{ITyp}$. Si σ est une solution constante de $\mathsf{CT}(e, \Gamma, T)$, alors $\Gamma \vdash e : \tau$ où $\tau = \sigma(T)$.

Preuve. On procède par induction sur e ; il y a 5 cas.

- ▷ Dans le cas $e = e_1\ e_2$, on écrit

  $$\mathsf{CT}(e, \Gamma, T) = \mathsf{CT}(e_1, \Gamma, T_1) \cup \mathsf{CT}(e_2, \Gamma, T_2) \cup \{T_1 \stackrel{?}{=} T_2 \to T\},$$

  où $\mathit{И}\ T_1$, T_2. Soit σ une solution constante de $\mathsf{CT}(e, \Gamma, T)$. Alors,
  - σ est une solution constante de $\mathsf{CT}(e_1, \Gamma, T_1)$;
  - σ est une solution constante de $\mathsf{CT}(e_2, \Gamma, T_1)$.

  Et, par induction, on sait que
  - $\Gamma \vdash e_1 : \sigma(T_1)$;
  - $\Gamma \vdash e_2 : \sigma(T_2)$.

  Par ailleurs, $\sigma(T_1) = \sigma(T_2) \to \sigma(T)$. On en conclut en appliquant $\mathscr{T}_{\mathrm{a}}$.
- ▷ Les autres cas se traitent similairement.

□

Proposition 7.4. Supposons $\Gamma \vdash e : \tau$. Alors, pour tout $T \in \mathrm{ITyp}$, il existe σ une solution constante de $\mathsf{CT}(e, \Gamma, T)$ telle que l'on ait l'égalité $\sigma(T) = \tau$.

Preuve. On procède par induction sur e. Il y a 5 cas.

- ▷ Dans le cas $e = e_1\ e_2$, supposons $\Gamma \vdash e_1\ e_2 : \tau$. Nécessairement, cette dérivation provient de $\Gamma \vdash e_1 : \tau_2 \to \tau$ et aussi $\Gamma \vdash e_2 : \tau_2$.

  Soit $T_0 \in \mathrm{ITyp}$, on a

  $$\mathsf{CT}(e, \Gamma, T_0) = \mathsf{CT}(e_1, \Gamma, T_1) \cup \mathsf{CT}(e_2, \Gamma, T_2) \cup \{T_1 \stackrel{?}{=} T_2 \to T_0\}.$$

Et, par induction, on a σ_1 et σ_2 des solutions constantes de $\mathsf{CT}(e_1, \Gamma, T_1)$ et $\mathsf{CT}(e_2, \Gamma, T_2)$ avec $\sigma_1(T_1) = \tau_2 \to \tau$ et $\sigma_2(T_2) = \tau_2$.

On définit σ en posant :

- $\sigma(T) = \sigma_1(T)$ si $T \in \mathrm{Supp}(\mathsf{CT}(e_1, \Gamma, T_1))$;
- $\sigma(T) = \sigma_2(T)$ si $T \in \mathrm{Supp}(\mathsf{CT}(e_2, \Gamma, T_2))$;
- $\sigma(T_0) = \tau$.

On vérifie bien que σ est solution constante de $\mathsf{CT}(e, \Gamma, T_0)$.

▷ Les autres cas se traitent similairement.

□

Théorème 7.1. On a $\Gamma \vdash e : \tau$ si, et seulement si $\forall T \in \mathrm{ITyp}$, l'ensemble de contraintes $\mathsf{CT}(e, \Gamma, T)$ admet une solution constante σ tel que $\sigma(T) = \tau$. □

Remarque 7.8. On a caractérisé l'ensemble des dérivations de $\Gamma \vdash e : \tau$ avec l'ensemble des solutions constantes de $\mathsf{CT}(e, \Gamma, T)$.

7.4.2 Termes et unification.

On va momentanément oublier FUN, pour généraliser à tout ensemble d'expressions. Ceci permet d'appliquer cet algorithme à une grande variété de « langages ».

Définition 7.7. On se donne

- ▷ un ensemble fini Σ de *constantes*, notées f, g, a, b où chaque constante $f \in \Sigma$ a un entier naturel nommé *arité* ;
- ▷ un ensemble infini V d'*inconnues*/de *variables*/de *variables d'unification* ; notées X, Y, Z (mais parfois x, y, z).

L'ensemble $\mathsf{T}(\Sigma, \mathrm{V})$ des *termes* sur (Σ, V), notés t, u, *etc*, est

défini de manière inductive, ce qui est décrit par la grammaire :

$$t ::= f^k(t_1, \dots, t_k) \mid X,$$

où f est une constante d'arité k.

Remarque 7.9. L'intuition est que l'on étend, comme lors du passage de Typ à $\widehat{\mathsf{Typ}}$, un langage de départ pour ajouter des inconnues. La définition inductive a $|\Sigma| + 1$ constructeurs.

Intuitivement, les $X \in \mathrm{V}$ ne fait pas partie du langage de départ. Il n'y a pas de liens pour X.

Exemple 7.8. Avec $\Sigma = \{f^2, g^1, a^0, b^0\}$,

$$t_0 := f(g(a), f(X, f(Y, g(X)))) \in \mathsf{T}(\Sigma, V)$$

est un terme.

Définition 7.8. On définit $\mathsf{Vars}(t)$ l'ensemble des inconnues/variables de t par induction sur t. Il y a deux familles de cas :

- $\mathsf{Vars}(f(t_1, \dots, t_k)) = \mathsf{Vars}(t_1) \cup \dots \cup \mathsf{Vars}(t_k)$;
- $\mathsf{Vars}(X) = \{X\}$.

Exemple 7.9. Avec l'expression t_0 précédente, on a

$$\mathsf{Vars}(t_0) = \{X, Y\}.$$

Définition 7.9. Une *substitution*, notée $\sigma, \sigma_1, \sigma'$, *etc*, est un dictionnaire sur $(\mathrm{V}, \mathsf{T}(\Sigma, \mathrm{V}))$.

Si $X \in \mathrm{dom}(\sigma)$, on dit que σ est *définie* en X.

Soit σ une substitution et $t \in \mathsf{T}(\Sigma, \mathrm{V})$. Le résultat de l'application

de σ à t, noté $\sigma(t)$, est défini par induction sur t, il y a deux familles de cas :

- ▷ $\sigma(f(t_1, \dots, t_k)) = f(\sigma(t_1), \dots, \sigma(t_k))$;
- ▷ $\sigma(X) = X$ si $X \notin \mathrm{dom}(\sigma)$;
- ▷ $\sigma(X)$ est le terme associé à X dans σ si $X \in \mathrm{dom}(\sigma)$.

Exemple 7.10. Avec $\sigma = [X \mapsto g(Y), Y \mapsto b]$, on a

$$\sigma(t_0) = f(g(a), f(\underbrace{g(Y)}, f(b, g(\underbrace{g(Y)}))))).$$

Attention ! On n'a pas de terme en $g(b)$: c'est une substitution *simultanée*.

Note 7.2. On rappelle qu'un dictionnaire peut être vu comme un ensemble fini de couples (X, t) avec $X \in \mathrm{V}$ et $t \in \mathsf{T}(\Sigma, \mathrm{V})$ tel que, pour toute variable $X \in \mathrm{V}$, il y a au plus un couple de la forme (X, t) dans la liste.

On utilise la notation $[t/X]$ pour représenter la notation $[X \mapsto t]$. Ceci est utiliser que lorsqu'on ne change qu'*une* variable.

Définition 7.10. Un *problème d'unification* est la donnée d'un ensemble fini de paires de termes (les contraintes) dans $\mathsf{T}(\Sigma, \mathrm{V})$. On note un tel problème $\mathscr{P} = \{t_1 \stackrel{?}{=} u_1, \dots, t_k \stackrel{?}{=} u_k\}$.

Une *solution*, un *unificateur*, d'un tel $\mathscr{P}$ est une substitution σ telle que, pour toute contrainte $t \stackrel{?}{=} u$ dans $\mathscr{P}$, $\sigma(t)$ et $\sigma(u)$ sont le même terme, ce que l'on note $\sigma(t) = \sigma(u)$.

On note $\mathrm{U}(\mathscr{P})$ l'ensemble des unificateurs de P.

Exemple 7.11. Avec le problème d'unification

$$\mathscr{P}_1 = \{f(a, g(X)) \stackrel{?}{=} f(Z, Y), g(T) \stackrel{?}{=} g(Z)\},$$

les substitutions

- $\sigma_1 = [Z \mapsto a, Y \mapsto g(X), T \mapsto a]$;
- $\sigma_2 = [Z \mapsto a, Y \mapsto g(b), T \mapsto a, X \mapsto b]$;

sont des solutions de $\mathscr{P}_1$. Mais,

$$\sigma_3 = [Z \mapsto f(b,b), T \mapsto f(b,b), Y \mapsto g(b), X \mapsto b]$$

n'est pas une solution.

Laquelle des solutions σ_1 et σ_2 est meilleure ? On remarque que $\sigma_2 = [{}^{b}/_{X}] \circ \sigma_1$ (où la composition est définie « comme on le pense »[5]). Ainsi, σ_1 est « plus général » que σ_2 ; σ_2 est un « cas particulier » de σ_1.

Exemple 7.12 (Aucune solution)**.** Les problèmes

- $\mathscr{P}_2 = \{f(X,Y) \stackrel{?}{=} g(Z)\}$;
- $\mathscr{P}_3 = \{f(X,Y) \stackrel{?}{=} X\}$

n'ont aucune solution : $\mathrm{U}(\mathscr{P}_2) = \mathrm{U}(\mathscr{P}_3) = \emptyset$.

7.4.3 Algorithme d'unification (du premier ordre).

Définition 7.11. Un *état* est soit un couple $(\mathscr{P}, \sigma)$, soit $\bot$ (*l'état d'échec*).

Un état de la forme $(\emptyset, \sigma)$ est appelé *état de succès*.

Un état qui n'est, ni échec, ni succès, peut s'écrire sous la forme $(\{t \stackrel{?}{=} t'\} \sqcup \mathscr{P}, \sigma)$, la contrainte $t \stackrel{?}{=} t'$ étant choisie de manière non-déterministe.

On définit une relation binaire $\to$ entre états par :

- $\bot \not\to$;
- $(\emptyset, \sigma) \not\to$;

5. Elle sera définie formellement ci-après.

▷ Il ne reste que les cas ni succès, ni échec, que l'on traite par la disjonction de cas :

1. $(\{f(t_1,\dots,t_k) \stackrel{?}{=} f(u_1,\dots,u_n) \sqcup \mathscr{P}, \sigma\}) \to (\{t_1 \stackrel{?}{=} u_1,\dots,t_k \stackrel{?}{=} u_k\} \cup \mathscr{P}, \sigma)$;
2. $(\{f(t_1,\dots,t_k) \stackrel{?}{=} g(u_1,\dots,u_n) \sqcup \mathscr{P}, \sigma\}) \to \bot$ si $f \neq g$;
3. $(\{X \stackrel{?}{=} t\} \sqcup \mathscr{P}, \sigma) \to (\mathscr{P}[t/X], [t/X] \circ \sigma)$ où
   - $X \notin \mathsf{Vars}(t)$,
   - $\mathscr{P}[t/X] = \{u[t/X] \stackrel{?}{=} u'[t/X] \mid (u \stackrel{?}{=} u') \in \mathscr{P}\}$,
   - et $[t/X] \circ \sigma$ est la substitution telle que, quel que soit $Y \in \mathrm{V}$, $([t/X] \circ \sigma)(Y) = (\sigma(Y))[t/X]$;
4. $(\{X \stackrel{?}{=} t\} \sqcup \mathscr{P}, \sigma) \to \bot$ si $X \in \mathsf{Vars}(t)$ et $t \neq X$;
5. $(\{X \stackrel{?}{=} X\} \sqcup \mathscr{P}, \sigma) \to (\mathscr{P}, \sigma)$.

L'*état initial* de l'algorithme correspond à $(\mathscr{P}, \emptyset)$: le problème $\mathscr{P}$ muni de la substitution vide $\emptyset$.

Exemple 7.13. On applique l'algorithme d'unification comme

montré ci-dessous :

$$
\begin{aligned}
& \{\underbrace{f(a,X) \stackrel{?}{=} f(Y,a)}_{\text{choix}}, g(X) \stackrel{?}{=} g(Y)\}, \emptyset \\
\to \quad & \{\underbrace{a \stackrel{?}{=} Y}_{\text{choix}}, X \stackrel{?}{=} a, g(X) \stackrel{?}{=} g(Y)\}, \emptyset \\
\to \quad & \{\underbrace{X \stackrel{?}{=} a}_{\text{choix}}, g(X) \stackrel{?}{=} g(a)\}, [Y \mapsto a] \\
\to \quad & \{\underbrace{g(a) \stackrel{?}{=} g(a)}_{\text{choix}}\}, [Y \mapsto a, X \mapsto a] \\
\to \quad & \{\underbrace{a \stackrel{?}{=} a}_{\text{choix}}\}, [Y \mapsto a, X \mapsto a] \\
\to \quad & \emptyset, [Y \mapsto a, X \mapsto a] \\
& .
\end{aligned}
$$

On peut remarquer que l'ensemble des clés de σ n'apparaît pas dans le problème ni dans les autres termes de la substitution : lorsqu'on ajoute une clé, elle disparaît du problème.

Définition 7.12. Un état $(\mathscr{P}, \sigma)$ est en *forme résolue* si, pour toute clé $X \in \mathrm{dom}(\sigma)$, alors X n'apparaît pas dans $\mathscr{P}$ et, quel que soit la clé $Y \in \mathrm{dom}(\sigma)$ alors $X \notin \mathsf{Vars}(\sigma(Y))$.

Remarque 7.10 (Notation)**.** Une substitution σ peut être vue comme un problème d'unification, que l'on note $\overset{?}{\sigma}$. (On passe d'un ensemble de couples à un ensemble de paires.)

Proposition 7.5. Si $(\mathscr{P}_0, \sigma_0)$ est en forme résolue et $(\mathscr{P}_0, \sigma_0) \to (\mathscr{P}_1, \sigma_1)$ alors $(\mathscr{P}_1, \sigma_1)$ est en forme résolue et

$$\mathrm{U}(\mathscr{P}_0 \cup \overset{?}{\sigma_0}) = \mathrm{U}(\mathscr{P}_1 \cup \overset{?}{\sigma_1}).$$

Preuve. La vraie difficulté se trouve dans le 3ème cas (les cas 1 et 5 sont immédiats). Pour cela, on utilise le lemme « technique » ci-dessous.

Lemme 7.4. Si $X \notin \mathrm{dom}(\sigma)$ alors

$$[t/X] \circ \sigma = \Big[X \mapsto t, Y_1 \mapsto (\sigma(Y_1))[t/X], \dots, Y_l \mapsto (\sigma(Y_k))[t/X]\Big],$$

où $\mathrm{dom}(\sigma) = \{Y_1, \dots, Y_k\}$. □

□

Proposition 7.6. On note $\to^\star$ la clôture réflexive et transitive de la relation $\to$.

1. Un *unificateur le plus général* (*mgu*[6] dans la littérature anglaise) est une solution $\sigma \in \mathrm{U}(\mathscr{P})$ telle que, quelle que soit $\sigma' \in \mathrm{U}(\mathscr{P})$, il existe σ'' telle que $\sigma' = \sigma'' \circ \sigma$.

   Si $(\mathscr{P}, \emptyset) \to^\star (\emptyset, \sigma)$ alors σ est un unificateur le plus général de $\mathscr{P}$.
2. Si $(\mathscr{P}, \emptyset) \to^\star \bot$ alors $\mathrm{U}(\mathscr{P}) = \emptyset$.

Preuve. 1. On montre par induction sur $(\mathscr{P}, \emptyset) \to^\star (\emptyset, \sigma)$ l'égalité $\mathrm{U}(\mathscr{P}) = \mathrm{U}(\overset{?}{\sigma})$ à l'aide de la proposition précédente. Puis, on conclut avec le lemme suivant.

Lemme 7.5. Pour toute substitution σ, alors σ est un unificateur le plus général de $\overset{?}{\sigma}$.

6. Pour *Most Général Unifier*

Preuve. Soit $\sigma' \in \mathrm{U}(\overset{?}{\sigma})$ et soit $X \in \mathrm{V}$. On montre que $\sigma' \circ \sigma = \sigma'$.

- ▷ Si $X \in \mathrm{dom}(\sigma)$, alors $\sigma'(\sigma(X)) = \sigma'(X)$ car σ' satisfait la contrainte $X \overset{?}{=} \sigma(X)$.
- ▷ Si $X \notin \mathrm{dom}(\sigma)$ alors $\sigma'(\sigma(X)) = \sigma'(X)$.

Ainsi $\sigma' \circ \sigma = \sigma'$. □

2. On montre que si $(\mathscr{P}, \emptyset) \to \bot$ alors $\mathrm{U}(\mathscr{P} \cup \overset{?}{\sigma})$. Pour le 2nd cas, c'est immédiat. Pour le 4ème cas, on procède par l'absurde. Soit σ_0 qui satisfait $X \overset{?}{=} t$ avec $X \in \mathsf{Vars}(t)$ et $X \neq t$. Alors $\sigma_0(X) = \sigma_0(t)$, qui contient $\sigma_0(X)$ et c'est un sous-ensemble strict. Absurde.

   On raisonne ensuite par induction sur $\to^\star$ pour conclure que $(\mathscr{P}, \emptyset) \to^\star (\mathscr{P}_0, \sigma_0) \to \bot$.

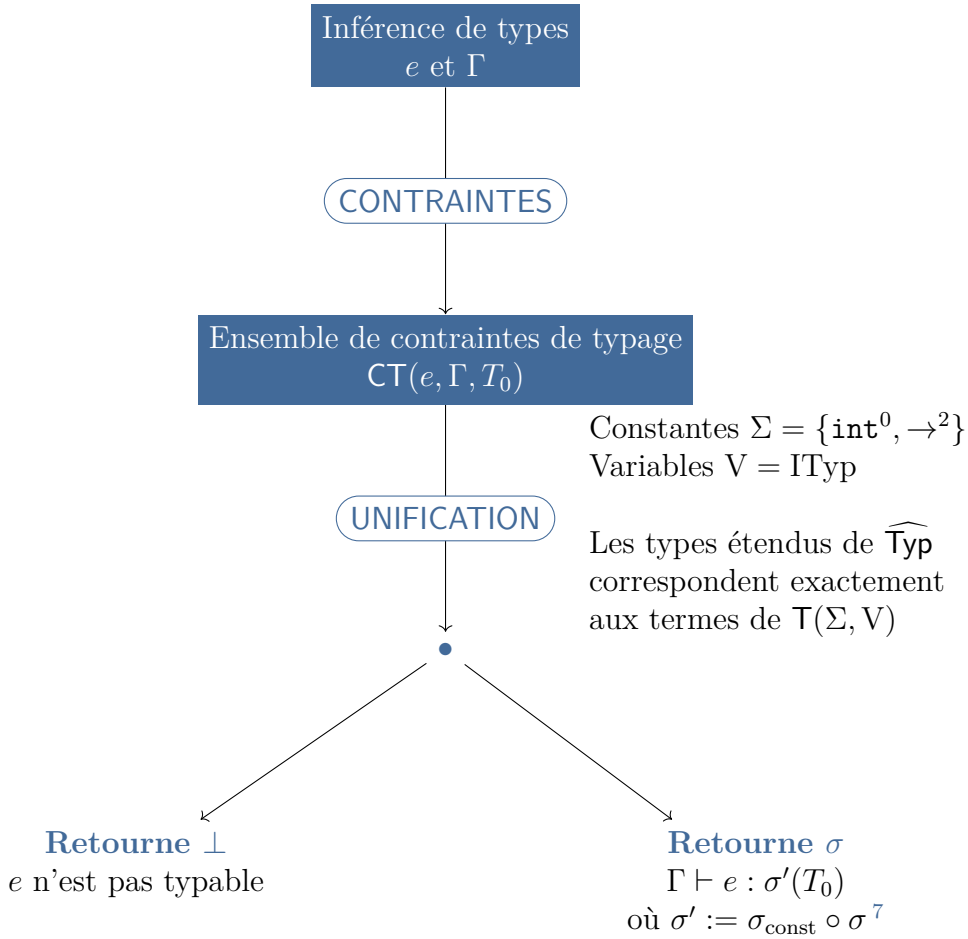
□

Lemme 7.6. La relation $\to$ est terminante (il n'y a pas de chaîne infinie avec cette relation).

Preuve. Vue plus tard. □

Théorème 7.2. L'algorithme d'unification calcule un unificateur le plus général si, et seulement si le problème initial a une solution. □

7.4.4 Retour sur l'inférence de types pour FUN.



Ceci conclut notre étude du petit langage fonctionnel FUN.

7. L'unificateur le plus général peut contenir des variables dans ses valeurs qui ne sont pas des clés (par exemple lors du typage de $\texttt{fun}\, x \to x$). Il faut donc composer σ avec une substitution « constante » pour effacer ces variables inutilisée.

8. Un petit langage impératif, IMP.

Sommaire.

8.1 Syntaxe et sémantique opérationnelle.

On se donne $\mathbb{Z}$ et V un ensemble infini de variables IMP, notées x, y, z. On définit plusieurs grammaires :

Arith. Les expressions arithmétiques $a ::= \underline{k} \mid a_1 \oplus a_2 \mid x$;[1]

Valeurs booléennes. $bv ::= \texttt{true} \mid \texttt{false}$;

Bool. Les expressions booléennes $b ::= bv \mid b_1 \wedge b_2 \mid a_1 \geq a_2$;

Com. Les commandes $c ::= x \texttt{ := } a \mid c_1 \texttt{ ; } c_2 \mid \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \mid \texttt{while } b \texttt{ do } c \mid \texttt{skip}$.

1. Et on arrêtera rapidement de mettre des barres sous les entiers et d'entourer les plus.

Sans explicitement le dire, on s'autorise à étendre les expressions arithmétiques avec, par exemple, les produits, les soustractions. De même pour les expressions booléennes.

On définit, par induction sur c, $\mathsf{Vars}(c)$ l'ensemble des variables dans la commande c. Il y a 5 cas.

Exemple 8.1. La commande

$$z \texttt{ := } 1 \texttt{ ; while } (x > 0) \texttt{ do } (z \texttt{ := } z \times x \texttt{ ; } x \texttt{ := } x - 1)$$

représente un programme calculant la factorielle d'un nombre x. On le notera c_{fact}.

8.1.1 Sémantique opérationnelle à grands pas.

Définition 8.1 (États mémoire). On se donne $\mathscr{M}$ un ensemble de dictionnaires, notés σ, σ', *etc* sur $(\mathrm{V}, \mathbb{Z})$.

Si $x \in \mathrm{dom}(\sigma)$ et $k \in \mathbb{Z}$ on note $\sigma[x \mapsto k]$ l'état mémoire σ' défini par

- $\sigma'(x) := k$;
- $\sigma'(y) := \sigma(y)$ si $y \in \mathrm{dom}(\sigma) \setminus \{x\}$.

Ici, on *écrase* la valeur de x dans l'état mémoire σ.

On définit $c, \sigma \Downarrow \sigma'$ (l'évaluation de c sur σ produit σ', c fait passer de σ à σ') par les règles d'inférences ci-dessous

$$\frac{}{\texttt{skip}, \sigma \Downarrow \sigma}\ \mathscr{C}_{\text{skip}} \qquad \frac{c_1, \sigma \Downarrow \sigma' \qquad c_2, \sigma' \Downarrow \sigma''}{c_1 \texttt{ ; } c_2, \sigma \Downarrow \sigma''}\ \mathscr{C}_{\text{seq}}$$

$$\frac{b, \sigma \Downarrow \texttt{true} \qquad c_1, \sigma \Downarrow \sigma'}{\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \Downarrow \sigma'}\ \mathscr{C}_{\text{it}} \qquad \frac{b, \sigma \Downarrow \texttt{false} \qquad c_2, \sigma \Downarrow \sigma'}{\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma \Downarrow \sigma'}\ \mathscr{C}_{\text{if}}$$

$$\sigma' = \sigma[x \mapsto k]\ \frac{a, \sigma \Downarrow k}{x \texttt{ := } a, \sigma \Downarrow \sigma'}\ \mathscr{C}_{\text{aff}} \qquad \frac{b, \sigma \Downarrow \texttt{false}}{\texttt{while } b \texttt{ do } c, \sigma \Downarrow \sigma}\ \mathscr{C}_{\text{wf}}$$

$$\frac{b, \sigma \Downarrow \texttt{true} \qquad c, \sigma \Downarrow \sigma' \qquad \texttt{while } b \texttt{ do } c, \sigma' \Downarrow \sigma''}{\texttt{while } b \texttt{ do } c, \sigma \Downarrow \sigma''} \mathscr{E}_{\text{wf}}$$

où l'on a deux autres relations (la couleur a de l'importance ici) :

- *l'évaluation des expressions arithmétiques* $a, \sigma \Downarrow k$ (a s'*évalue* en k dans σ)

$$\frac{}{\underline{k}, \sigma \Downarrow k} \qquad \sigma(x) = k \ \frac{}{x, \sigma \Downarrow k} \qquad k = k_1 + k_2 \ \frac{a_1, \sigma \Downarrow k_1 \qquad a_2, \sigma \Downarrow k_2}{a_1 \oplus a_2, \sigma \Downarrow k}$$

- *l'évaluation des expressions booléennes* $b, \sigma \Downarrow bv$ (b s'*évalue* en bv dans σ)

$$\frac{}{bv, \sigma \Downarrow bv} \qquad bv = \texttt{true ssi } bv_1 \text{ et } bv_2 \ \frac{b_1, \sigma \Downarrow bv_1 \qquad b_2, \sigma \Downarrow bv_2}{b_1 \wedge b_2, \sigma \Downarrow bv}$$

$$bv = \texttt{true ssi } k_1 \geq k_2 \ \frac{a_1, \sigma \Downarrow k_1 \qquad a_2, \sigma \Downarrow k_2}{a_1 \geq a_2, \sigma \Downarrow bv.}$$

Remarque 8.1 (des « variables » partout !).

- Les variables dans FUN sont les paramètres des fonctions, elles peuvent être liées, libres, et on peut procéder à de l'α-conversion. [2]
- Les variables d'unification sont des inconnues. Il y a une notion de substitution, mais pas de liaison.
- Les variables dans IMP sont des cases mémoire, des registres, et il n'y a pas de liaison.

Remarque 8.2. Soit c une commande, et $\sigma \in \mathscr{M}$. Il peut arriver que, quel que soit $\sigma' \in \mathscr{M}$, on n'ait pas $c, \sigma \Downarrow \sigma'$, soit parce que $\text{dom}(\sigma)$ est trop petit, et l'exécution se bloque ; soit parce que le programme diverge, par exemple

```
while true do skip
```

2. C'est similaire au cas de la variable x dans $\int_0^7 f(x)\,\mathrm{d}x$.

diverge car on n'a pas de dérivation finies :

$$\frac{\texttt{true}, \sigma \Downarrow \texttt{true} \qquad \overline{\texttt{skip}, \sigma \Downarrow \sigma} \qquad \texttt{while true do skip}, \sigma \Downarrow ?}{\texttt{while true do skip}, \sigma \Downarrow ?}\ \mathscr{E}_{\text{wt}}.$$

On peut définir des petits pas pour IMP (vu plus tard en cours, ou en TD), mais on s'intéresse plus à une autre sémantique, la *sémantique dénotationnelle*.

8.2 Sémantique dénotationnelle de IMP.

$$\begin{array}{ccc} c & \xrightarrow{\quad\mathscr{D}\quad} & \text{fonction partielle } \mathscr{M} \rightharpoonup \mathscr{M} \\ & & \updownarrow \\ & & \text{relation binaire sur } \mathscr{M} \text{ déterministe/fonctionnelle} \end{array}$$

On définit les relations

- $\mathscr{D}(a) \subseteq \mathscr{M} \times \mathbb{Z}$ fonctionnelle ;
- $\mathscr{D}(b) \subseteq \mathscr{M} \times \{\texttt{true}, \texttt{false}\}$ fonctionnelle ;
- $\mathscr{D}(c) \subseteq \mathscr{M} \times \mathscr{M}$ fonctionnelle.

On ne traitera que la définition de $\mathscr{D}(c)$, les autres sont laissées en exercice.

On définit $\mathscr{D}(c)$ par induction sur c, il y a 5 cas.

- $\mathscr{D}(\texttt{skip}) = \{(\sigma, \sigma)\}$;
- $\mathscr{D}(x \texttt{ := } a) = \{(\sigma, \sigma') \mid x \in \mathrm{dom}(\sigma), \sigma' = \sigma[x \mapsto k] \text{ et } (\sigma, k) \in \mathscr{D}(a)\}$;
- $\mathscr{D}(\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2) = \{(\sigma, \sigma' \mid (\sigma, \texttt{true}) \in \mathscr{D}(b), (\sigma, \sigma') \in \mathscr{D}(c_1))\} \cup \{(\sigma, \sigma' \mid (\sigma, \texttt{false}) \in \mathscr{D}(b), (\sigma, \sigma') \in \mathscr{D}(c_2))\}$;
- $\mathscr{D}(c_1 \texttt{ := } c_2) = \{(\sigma, \sigma'') \mid \exists \sigma', (\sigma, \sigma') \in \mathscr{D}(c_1) \text{ et } (\sigma', \sigma'') \in \mathscr{D}(c_2)\}$;[3]
- $\mathscr{D}(\texttt{while } b \texttt{ do } c) = ???$.

3. C'est la composée de $\mathscr{D}(c_2)$ avec $\mathscr{D}(c_1)$.

Pour la sémantique dénotationnelle de la boucle `while`, on s'appuie sur l'« équivalence » des commandes

$$\texttt{while}\ b\ \texttt{do}\ c \qquad \text{et} \qquad \texttt{if}\ b\ \texttt{then}\ (c\ \texttt{:=}\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip}.$$

On introduit, pour $\mathsf{R} \subseteq \mathscr{M} \times \mathscr{M}$, la relation

$$\begin{aligned} F(\mathsf{R}) = \quad & \{(\sigma,\sigma) \mid (\sigma, \texttt{false}) \in \mathscr{D}(b)\} \\ & \cup \{(\sigma,\sigma') \mid (\sigma,\texttt{true}) \in \mathscr{D}(b), \exists \sigma', (\sigma,\sigma') \in \mathscr{D}(c) \text{ et } (\sigma',\sigma'') \in \mathsf{R}\}. \end{aligned}$$

On a envie de définir $\mathscr{D}(\texttt{while}\ b\ \texttt{do}\ c)$ comme un point fixe de F.

L'ensemble des relations binaires fonctionnelles sur $\mathscr{M}$ **n'est pas** un treillis complet (à cause se $\mathsf{R}_1 \cup \mathsf{R}_2$ qui n'est pas nécessairement fonctionnelle). On ne peut donc pas appliquer le théorème de Knaster-Tarski.

En revanche, c'est un domaine : si $e_0 \subseteq e_1 \subseteq \cdots \subseteq e_n \subseteq \cdots$ alors l'union $\bigcup_{i\geq 0} e_i$ existe. L'inclusion $e \subseteq e'$ signifie que e' est « plus définie » que e. L'ensemble des relations fonctionnelles sur $\mathscr{M}$ est donc un domaine avec $\bot = \emptyset$. On sait donc que, pour toute fonction F continue, alors F admet un point fixe, qui est égal à

$$\emptyset \cup F(\emptyset) \cup F^2(\emptyset) \cup \cdots = \bigcup_{i\geq 0} F^i(\emptyset).$$

La fonction F définie plus haut est continue, ce qui nous permet de définir

$$\mathscr{D}(\texttt{while}\ b\ \texttt{do}\ c) = \bigcup_{i\geq 0} F^i(\emptyset).$$

Exemple 8.2. On considère $c_0 = \texttt{while}\ x \neq 3\ \texttt{do}\ x\ \texttt{:=}\ x - 1$. Ainsi, la fonction F définie avant $c = c_0$ est

$$\begin{aligned} F_0(\mathsf{R}) = \quad & \{(\sigma,\sigma) \mid \sigma(x) = 3\} \\ & \cup \{(\sigma,\sigma'') \mid \sigma(x) \neq 3, \exists \sigma', \sigma = [x \mapsto \sigma(x) - 1], (\sigma,\sigma') \in \mathsf{R}\}. \end{aligned}$$

On a

- ▷ $F_0^0(\emptyset) = \{(\sigma,\sigma) \mid \sigma(x) = 3\}$;

- $F_0^1(\emptyset) = \{(\sigma,\sigma) \mid \sigma(x) = 3\} \cup \{(\sigma,\sigma') \mid \sigma' = [x \mapsto 3], \sigma(x) = 4\}$;
- $F_0^2(\emptyset) = \{(\sigma,\sigma) \mid \sigma(x) = 3\} \cup \{(\sigma,\sigma') \mid \sigma' = [x \mapsto 3], \sigma(x) \in \{4,5\}\}$;
- $F_0^2(\emptyset) = \{(\sigma,\sigma) \mid \sigma(x) = 3\} \cup \{(\sigma,\sigma') \mid \sigma' = [x \mapsto 3], \sigma(x) \in \{4,5,6\}\}$;
- *etc.*

On a bien

$$\emptyset \subseteq F_0(\emptyset) \subseteq F_0^2(\emptyset) \subseteq \cdots .$$

Si $\sigma(x) = 0$, alors quel que soit σ', on a $(\sigma,\sigma') \notin \mathscr{D}(c_0)$.

Exemple 8.3. Ainsi défini,

$$\mathscr{D}(\texttt{while true do skip}) = \emptyset.$$

Théorème 8.1. On a $c, \sigma \Downarrow \sigma'$ si et seulement si $(\sigma,\sigma') \in \mathscr{D}(c)$.

Preuve. ▷ « $\Longrightarrow$ » Par induction sur la relation $c, \sigma \Downarrow \sigma'$.

▷ « $\Longleftarrow$ » Par induction sur c, où l'on utilise le résultat suivant :

$$\forall n, \quad (\sigma,\sigma') \in F^n(\emptyset) \implies c, \sigma \Downarrow \sigma'.$$

□

Lemme 8.1. Quels que soient c, σ, σ_1, si c, σ, σ_1 alors,

$$\forall \sigma_2, \quad c, \sigma \Downarrow \sigma_2 \implies \sigma_1 = \sigma_2.$$

Preuve. Une mauvaise idée est de procéder par induction sur c. Il y a 5 cas, et dans le cas `while`, ça bloque parce que la relation grands pas n'est pas définie par induction sur c dans le cas `while`.

On procède par induction sur $c, \sigma \Downarrow \sigma_1$. □

De manière générale, avec IMP, on ne montre pas des résultats de la forme $c, \sigma \Downarrow \sigma' \implies \mathscr{P}$ par induction sur c, car cela ne fonctionne

pas, on n'a pas les bonnes hypothèses. On procède par induction sur la relation $c, \sigma \Downarrow \sigma'$.

8.3 Coinduction.

On retourne sur le théorème de Knaster-Tarski pour la définition d'ensembles et de relations. En notant E l'ensemble ambiant, on travaille dans le treillis complet $(\wp(E), \subseteq)$, avec des fonctions f croissantes dans $\wp(E)$. Le théorème de Knaster-Tarski nous donne ainsi le plus petit pré-point fixe de f, que l'on notera μf. Le principe de la preuve par induction est ainsi :

$$\text{si } A \subseteq E \text{ vérifie } f(A) \subseteq A \text{ alors on a } \mu f \subseteq A.$$

De plus, si f est continue (car $(\wp(E), \subseteq)$ est un domaine), alors on peut calculer explicitement ce plus petit (pré)-point fixe avec la formule $\bigcup_{n\in\mathbb{N}} f^n(\emptyset)$. On part du « bas » et on ajoute des éléments un par un.

Exemple 8.4. Pour l'exemple de `nat`, on a

$$\forall A \subseteq E, \qquad f(A) = \{\mathtt{O}\} \cup \{\mathtt{S}\ x \mid x \in A\},$$

c'est une fonction continue, et on a

$$\mu f = \{\mathtt{S}^n\mathtt{O} \mid n \in \mathbb{N}\},$$

avec $\mathtt{S}^n\ x = \mathtt{S}\ \mathtt{S}\ \cdots\ \mathtt{S}\ x$ et la convention $\mathtt{S}^0\ x = x$. En effet, on a l'appartenance de $\mathtt{S}^n\mathtt{O} \in \bigcup_{m\in\mathbb{N}} f^m(\emptyset)$ et $f(\{\mathtt{S}^n\ \mathtt{O}\}) = \{\mathtt{S}^{n+1}\ \mathtt{O}\}$.

Remarque 8.3 (Remarque fondamentale !)**.** Considérons un treillis complet $(E, \sqsubseteq)$. Alors, le treillis $(E, \sqsupseteq)$ est complet, où l'on note $y \sqsupseteq x$ dès lors que $x \sqsubseteq y$ (on renverse l'ordre).

Un majorant pour $\sqsubseteq$ est un minorant pour $\sqsupseteq$ et inversement. Ainsi, le plus plus petit des majorants $\bigsqcup_{\sqsubseteq} A$ pour $\sqsubseteq$ est le plus petit des minorants $\bigsqcap_{\sqsupseteq} A$ pour $\sqsupseteq$. Réciproquement, le plus petit

> des majorants pour $\sqsubseteq$, $\bigsqcap_{\sqsubseteq} A$ est égal au plus grand majorant pour la relation $\sqsupseteq$, $\bigsqcup_{\sqsupseteq} A$.

On se place ainsi sur le treillis complet $(\wp(E), \supseteq)$. Une fonction est croissante pour $\subseteq$ si et seulement si elle est croissante pour $\supseteq$ (***attention***, elle n'est pas décroissance pour cette deuxième relation). Appliquons le théorème de Knaster-Tarski sur ce nouveau treillis complet à une fonction croissante. Le théorème nous fournis un pré-point fixe pour l'ordre $\supseteq$ (*i.e.* qui vérifie $f(A) \supseteq A$), c'est-à-dire un post-point fixe pour l'ordre $\subseteq$ (*i.e.* qui vérifie $A \subseteq f(A)$). Et, c'est le plus petit point fixe pour $\supseteq$, donc le plus grand point fixe pour $\subseteq$, que l'on notera νf.

Avec le théorème de point fixe sur les domaines, et en supposant f continue, on calcule explicitement que le plus grand point fixe νf vaut l'intersection $\bigcap_{n\in\mathbb{N}} f^n(E)$. On part du haut, et on nettoie progressivement, on raffine notre partie de E.

Ce que l'on a fait là, cela s'appelle de la ***coinduction***.

Exemple 8.5. Par exemple, on définit `conat` par coinduction. En Rocq, cela donne le code ci-dessous.

```
CoInductive conat : Set := cO | cS (n : conat).
```

Code 8.1 | *Définition de* `conat`

Pour illustrer le « nettoyage » effectué dans la définition coinductive, on considère une feuille étiquetée par le mot « `banane` ». A-t-on `cS banane` $\in$ `conat` ? Premièrement, on a `cS banane` $\in E$ car E est l'ensemble (très grand) des arbres étiquetés par des chaînes de caractères. Deuxièmeement, on a `cS banane` $\in f(E)$ car c'est le successeur de `banane` $\in E$. Troisièmement, et c'est là où ça casse, on a `cS banane` $\notin f^2(E)$ parce que `banane` $\notin f(E)$.

Avec la fonction f définie précédemment, on a

$$f^n(E) = \{\texttt{cO}, \texttt{cS cO}, \dots, \texttt{cS}^{n-1}\ \texttt{cO}\} \cup \{\texttt{cS}^n\ x \mid x \in E\}.$$

Ainsi, on récupère tous les entiers de nat, mais d'autres entiers (oui, il y en a plusieurs) infinis, ayant ainsi une dérivation infinie. Par exemple, il existe $\omega \in \mathsf{conat}$ tel que $\omega = \mathtt{cS}\ \omega$. En Rocq, pour le définir, on ferai :

```
CoFixpoint ω := cS ω
```

Code 8.2 | *Définition de ω, un entier infini*

Pour montrer que $\omega \in \mathsf{conat}$, il faut et il suffit de montrer l'inclusion $\{\omega\} \subseteq f(\{\omega\}) = \{\mathtt{c0}, \mathtt{cS}\ \omega\} = \{\mathtt{c0}, \omega\}$, qui est vraie, et on a ainsi $\{\omega\} \subseteq \mathsf{conat}$.

Le principe de la preuve par coinduction permet d'établir qu'un ensemble est contenu dans le plus grand point fixe. Avec le treillis des parties muni de $\subseteq$, cela permet de montrer que $A \subseteq E$ est inclus dans le plus grand post-point fixe de f et, pour cela, il suffit de montrer que $A \subseteq f(A)$, c'est-à-dire que A est un post-point fixe de f. C'est ce que l'on a fait dans l'exemple avec ω.

Par coinduction, on peut par exemple montrer que l'on a, pour tous états mémoire σ, σ',

$$\mathtt{while\ true\ do\ skip}, \sigma \Downarrow \sigma'.$$

8.4 Divergences en IMP.

On donne une définition coinductive de la divergence en IMP, que l'on notera $c, \sigma \Uparrow$ avec les règles

$$\frac{c_1, \sigma \Uparrow}{c_1\ ;\ c_2, \sigma \Uparrow} \qquad \frac{c_1, \sigma \Downarrow \sigma' \qquad c_2, \sigma' \Uparrow}{c_1\ ;\ c_2, \sigma \Uparrow} \qquad \frac{b, \sigma \Downarrow \mathtt{true} \qquad c_1, \sigma \Uparrow}{\mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2, \sigma \Downarrow}$$

$$\frac{b, \sigma \Downarrow \mathtt{false} \qquad c_2, \sigma \Uparrow}{\mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2, \sigma \Downarrow} \qquad \frac{b, \sigma \Downarrow \mathtt{true} \qquad c, \sigma \Uparrow}{\mathtt{while}\ b\ \mathtt{do}\ c, \sigma \Uparrow}$$

$$\frac{b, \sigma \Downarrow \mathtt{true} \qquad c, \sigma \Downarrow \sigma' \qquad \mathtt{while}\ b\ \mathtt{do}\ c, \sigma' \Uparrow}{\mathtt{while}\ b\ \mathtt{do}\ c, \sigma \Uparrow}$$

On n'a pas de règle pour la divergence si $b, \sigma \Downarrow \texttt{false}$, car dans ce cas là, on ne peut pas diverger (c'est équivalent à un `skip`).

Le plus grand point fixe ne contient que des dérivations infinies, qui correspondent à des exécutions divergentes d'un programme IMP à partir d'un état mémoire donné. En effet, ceci vient du fait que, si on interprète ces règles comme des règles inductives, la relation obtenue est l'ensemble vide...

8.5 Logique de Floyd–Hoare.

On considère des formules logiques, des *assertions* (définies formellement ci-après), que l'on notera A, A', B, *etc.* Un triplet de Hoare est de la forme $\{A\}c\{A'\}$ (la notation est inhabituelle pour les triplets, mais c'est une notation commune dans le cas des triplets de Hoare), où l'on nomme A la *précondition* et A' la *postcondition*.

Exemple 8.6. Les triplets suivants sont des triplets de Hoare :

1. $\{x \geq 1\} y \texttt{ := } x + 2 \{x \geq 1 \wedge y \geq 3\}$ qui est une conclusion naturelle ;
2. $\{n \geq 1\} c_{\text{fact}} \{r = n!\}$ où l'on note c_{fact} la commande

   $$x \texttt{ := } n \texttt{ ; } z \texttt{ := } 1 \texttt{ ; while } (x > 0) \texttt{ do } (z \texttt{ := } z \times x \texttt{ ; } x \texttt{ := } x - 1) \texttt{ ,}$$

   qui calcule naturellement la factorielle de n ;
3. $\{x < 0\} c \{\texttt{true}\}$ même s'il ne nous dit rien d'intéressant (tout état mémoire vérifie `true`) ;
4. $\{x < 0\} c \{\texttt{false}\}$ qui diverge dès lors que $x < 0$.

On considère un ensemble $I \ni i$ infini d'*index*, des « inconnues ». On commence par définir les expressions arithmétiques étendues

$$a ::= \underline{k} \mid a_1 \oplus a_2 \mid x \mid i,$$

puis définit les *assertions* par la grammaire ci-dessous :

$$A ::= bv \mid A_1 \vee A_2 \mid A_1 \wedge A_2 \mid a_1 \geq a_2 \mid \exists i, A.$$

On s'autorisera à étendre, implicitement, les opérations réalisées dans les expressions arithmétiques, et les comparaisons effectuées dans les assertions.

On ajoute la liaison d'α-conversion : les assertions $\exists i, x = 3 * i$ et $\exists j, x = 3 * j$ sont α-équivalentes. On note $i\ell(A)$ l'ensemble des index libres de l'assertion A, et on dira que A est *close* dès lors que $i\ell(A) = \emptyset$. On note aussi $A[^k/_i]$ l'assertion A où $k \in \mathbb{Z}$ remplace $i \in I$.

Définition 8.2. Considérons A close et $\sigma \in \mathscr{M}$. On définit par induction sur A (4 cas) une relation constituée de couples (σ, A), notés $\sigma \models A$ (« σ satisfait A »), et en notant $\sigma \not\models A$ lorsque (σ, A) n'est pas dans la relation :

- $\sigma \models \texttt{true}\ \forall \sigma \in \mathscr{M}$;
- $\sigma \models A_1 \vee A_2$ si et seulement si $\sigma \models A_1$ ou $\sigma \models A_2$;
- $\sigma \models a_1 \geq a_2$ si et seulement si on a $a_1, \sigma \Downarrow k_1$ et $a_2, \sigma \Downarrow k_2$ et $k_1 \geq k_2$;
- $\sigma \models \exists i, A$ si et seulement s'il existe $k \in \mathbb{Z}$ tel que $\sigma \models A[^k/_i]$.

On écrit $\models A$ (« A est valide ») lorsque pour tout σ tel que $\mathrm{dom}(\sigma) \supseteq \mathsf{vars}(A)$, on a $\sigma \models A$.

8.5.1 Règles de la logique de Hoare : dérivabilité des triplets de Hoare.

Les triplets de Hoare, notés $\{A\}c\{A'\}$ avec A et A' closes, où A est *précondition*, c est commande IMP, et A' est *postcondition*. On définit

une relation $\vdash \{A\}c\{A'\}$ sur les triplets de Hoare :

$$\frac{\vdash \{A \wedge b\}c_1\{A'\} \quad \vdash \{A \wedge \neg b\}c_2\{A'\}}{\vdash \{A\}\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2\{A'\}} \qquad \frac{}{\vdash \{A\}\texttt{skip}\{A\}}$$

$$\frac{\vdash \{A\}c_1\{A'\} \quad \vdash \{A'\}c_2\{A''\}}{\vdash \{A\}c_1 \texttt{ ; } c_2\{A''\}} \qquad \frac{\vdash \{A \wedge b\}c\{A\}}{\{A\}\texttt{while } b \texttt{ do } c\{A \wedge \neg B\}}$$

$$\frac{\models B \implies A \quad \models A' \implies B'}{\{A\}c\{A'\}} \qquad \frac{}{\{A[a/x]\}x \texttt{ := } a\{A\}}$$

La dernière règle semble à l'envers, mais c'est parce que la logique de Hoare fonctionne fondamentalement à l'envers.

Dans la règle de dérivation pour la boucle `while`, l'assertion manipulée, A, est un *invariant*.

L'avant dernière règle s'appelle la *règle de conséquence* : on ne manipule pas le programme, la commande, mais plutôt les pré- et post-conditions.

La relation $\vdash \{A\}c\{A'\}$ s'appelle la *sémantique opérationnelle* de IMP.

Définition 8.3. On définit la relation de *satisfaction*, sur les triplets de la forme $\{A\}c\{A'\}$ avec A, A' closes, avec $\sigma \models \{A\}c\{A'\}$ si et seulement si dès lors que $\sigma \models A$ et $c, \sigma \Downarrow \sigma'$ alors on a $\sigma' \models A'$.

On définit ensuite la relation de *validité* par $\models \{A\}c\{A'\}$ si et seulement si pour tout $\sigma \in \mathscr{M}$, $\sigma \models \{A\}c\{A'\}$.

Théorème 8.2 (Correction de la logique de Hoare.)**.** Si $\vdash \{A\}c\{A'\}$ alors $\models \{A\}c\{A'\}$.

Preuve. On procède par induction sur $\vdash \{A\}c\{A'\}$. Il y a 6 cas.

▷ *Règle de conséquence.* On sait

$$\models B \implies A \text{ et } \models A' \implies B',$$

et l'hypothèse d'induction. On doit montrer $\models \{B\}c\{B'\}$. Soit σ tel que $\models B$, et supposons $c, \sigma \Downarrow \sigma'$. On a $\models A$ par hypothèse. Puis, par hypothèse d'induction, $\sigma' \models A'$ et donc $\sigma' \models B'$.

▷ *Règle* `while`. Considérons $c =$ `while` b `do` c_0. On sait par induction que $\models \{A \wedge b\}c_0\{A\}$ et l'hypothèse d'induction. Il faut montrer $\models \{A\}$`while` b `do` $c_0\{A \wedge \neg b\}$, c'est à dire, si $\sigma \models A$ et $(\star)$: `while` b `do` $c_0, \sigma \Downarrow \sigma'$ alors $\sigma' \models A \wedge \neg b$. Pour montrer cela, il est nécessaire de faire une induction sur la dérivation de $(\star)$, « sur le nombre d'itérations dans la boucle ».

▷ Autres cas en exercice.

□

Le sens inverse, la réciproque, s'appelle la *complétude*. On l'étudiera rapidement après.

Remarque 8.4. Concrètement, on écrit des programmes annotés.

$\{x \geq 1\}$

$\{x \geq 1 \wedge x+2+x+2 \geq 6\}$

$y := x + 2$;

$\{x \geq 1 \wedge y + y \geq 6\}$

$z := y + y$

$\{x \geq 1 \wedge z \geq 6\}$

$\{x \geq 1 \wedge z \geq 6\}$

8.5.2 Complétude de la logique de Hoare

Pour démontrer la complétude de la logique de Hoare, on s'appuie sur la notion de *plus faible précondition* : étant données une commande c et une assertion B, alors la *plus faible précondition* associée à c, B est l'ensemble des états mémoire

$$\mathrm{wp}(c, B) := \{\sigma \mid c, \sigma \Downarrow \sigma' \implies \sigma' \models B\}.$$

Ainsi, $\mathrm{wp}(c, B)$ est l'ensemble des états mémoire à partir duquels on aboutit à un état satisfaisant B, après une exécution terminante de c.

Proposition 8.1. Pour toute commande c et toute formule B, il existe une assertion $\mathrm{W}(c, B)$ telle que $\sigma \models \mathrm{W}(c, B)$ si et seulement si $\sigma \in \mathrm{wp}(c, B)$.

Preuve. On procède par induction sur c. Tout fonctionne, sauf pour *while*... Pour le cas de la boucle *while*, on utilise la caractérisation suivante :

$$\sigma \in \mathrm{wp}(\texttt{while}\ b\ \texttt{do}\ c_0, B)$$

$$\Updownarrow$$

$$\forall k, \forall \sigma_0, \dots, \sigma_k \text{ si } \sigma_0 = \sigma \text{ et } \forall i < k, (\sigma_i, b \Downarrow \texttt{true} \text{ et } c_0, \sigma_i \Downarrow \sigma_{i+1}) \text{ alors } \sigma_k \models b \vee B.$$

On peut définir cette assertion en définissant des assertions pour :

- ▷ décrire un état mémoire σ_i $(X_1^i = v_1 \wedge \cdots \wedge X_n^i = v_n)$;
- ▷ exprimer les conditions $\sigma_i, c \Downarrow \sigma_{i+1}$ par induction ;
- ▷ exprimer les quantifications $\forall k, \sigma_0, \dots, \sigma_k \dots$ on demande à Kurt Gödel.

Ainsi, on a bien une assertion $\mathrm{W}(c, B)$ telle que

$$\forall \sigma, \qquad \sigma \in \mathrm{wp}(c, B) \iff \sigma \models \mathrm{W}(c, B).$$

□

9. Mémoire structurée, logique de séparation

Sommaire.

La syntaxe des commandes de IMP avec tas est définie par la grammaire

$$c ::= x \texttt{ := } a \mid \texttt{skip} \mid c_1 \texttt{ ; } c_2 \mid \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \mid \texttt{while } b \texttt{ do } c$$
$$x \texttt{ := [}a\texttt{]} \mid \texttt{[}x_1\texttt{]} \texttt{ := } a \mid x \texttt{ := alloc}(k) \mid \texttt{free}(a).$$

Les expressions arithmétiques et booléennes restent inchangées.

Ces quatre nouvelles constructions correspondent respectivement à

- ▷ l'accès à une case mémoire ;
- ▷ la modification d'une valeur d'une case mémoire ;
- ▷ l'allocation de k cases mémoires ;
- ▷ la libération de cases mémoires.

Remarque 9.1. C'est un langage impératif de bas niveau : on manipule de la mémoire directement. On ne s'autorise pas tout, cependant. On ne s'autorise pas, par exemple,

$$\texttt{[}x + i + 1\texttt{]} \texttt{ := [}t + i\texttt{]} + \texttt{[}t + i - 1\texttt{]},$$

mais on demande d'écrire

$$x := [t+i] \ ; \ y := [t+i-1] \ ; \ [x+i+1] := x+y.$$

On raffine IMP : avant, les cases vivaient quelque part, mais on ne sait pas où, ce sont des registres ; maintenant, peut aussi allouer des « blocs » de mémoire, et on peut donc parler de cases mémoires adjacentes.

L'allocation `alloc` est *dynamique*, similaire à `malloc` en C, où l'on alloue de la mémoire dans l'espace mémoire appelé *tas*.

9.1 Sémantique opérationnelle de IMP avec tas.

Définition 9.1 (États mémoire). Un état mémoire est la donnée de

- ▷ σ un *registre*, c-à-d un dictionnaire sur $(\mathrm{V}, \mathbb{Z})$;
- ▷ h un *tas*, c-à-d un dictionnaire sur $(\mathbb{N}, \mathbb{Z})$, c'est un gros tableau. [1]

On définit $h[k_1 \mapsto k_2]$ que si $k_1 \in \mathrm{dom}(h)$ et alors il vaut le dictionnaire où l'on assigne k_2 à k_1.

On définit $h_1 \uplus h_2$ que si $\mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \emptyset$ et vaut l'union de dictionnaires h_1 et h_2.

On définit ainsi la sémantique dénotationnelle comme la relation $\Downarrow$ est une relation quinaire (*i.e.* avec 5 éléments) notée $c, \sigma, h \Downarrow \sigma', h'$ où c est une commande, h, h' sont deux tas, et σ, σ' sont deux registres.

1. On appelle parfois IMP avec tas, IMP avec tableau.

$$\sigma' = \sigma[x \mapsto k] \quad \frac{a, \sigma \Downarrow k}{x := a, \sigma, h \Downarrow \sigma', h} \qquad \frac{c_1, \sigma, h \Downarrow \sigma', h' \qquad c_2, \sigma', h' \Downarrow \sigma'', h''}{c_1 \ ; \ c_2, \sigma, h \Downarrow \sigma'', h''}$$

$$\begin{matrix} k' = h(k) \\ \sigma' = \sigma[x \mapsto k'] \end{matrix} \quad \frac{a, \sigma \Downarrow k}{x := [a], \sigma, h \Downarrow \sigma' h} \qquad h' = h[k_1 \mapsto k_2]^{\,2} \quad \frac{a_1, \sigma \Downarrow k_1 \qquad a_2, \sigma \Downarrow k_2}{[a_1] := a_2, \sigma, h \Downarrow \sigma, h'}$$

$$\begin{matrix} \{k', \dots, k'+k-1\} \cap \mathrm{dom}(h) = \emptyset \\ k > 0 \\ h' = h \uplus \{k' \mapsto 0, \dots, k'+k-1 \mapsto 0\} \\ \sigma' = \sigma[x \mapsto k'] \end{matrix} \quad \frac{}{x := \texttt{alloc}(k), \sigma, h \Downarrow \sigma', h'}$$

$$h = h' \uplus \{k \mapsto k'\} \quad \frac{a, \sigma \Downarrow k}{\texttt{free}(a), \sigma, h \Downarrow \sigma, h'.}$$

9.2 Logique de séparation.

Définition 9.2. On définit les assertions dans IMP avec tas comme l'enrichissement des assertions IMP avec les constructions `emp`, $\mapsto$ et $*$:

$$A ::= \cdots \mid \texttt{emp} \mid a_1 \mapsto a_2 \mid A_1 * A_2.$$

On enrichit ainsi la relation de satisfaction :

- $\sigma, h \models \texttt{emp}$ si et seulement si $h = \emptyset$;
- $\sigma, h \models a_1 \to a_2$ si et seulement si $a_1, \sigma \Downarrow k_1$, et $a_2, \sigma \Downarrow k_2$ et $h = [k_1 = k_2]$ (le tas est un singleton) ;
- $\sigma, h \models A_1 * A_2$ si et seulement si $h = h_1 \uplus h_2$ avec $\sigma, h_1 \models A_1$ et $\sigma, h_2 \models A_2$.

L'opérateur $*$ est appelé *conjonction séparante* : on découpe le tas en deux, chaque partie étant « observée » par une sous-assertion.

Note 9.1 (Notations).
- On note $a \mapsto \_$ pour $\exists i, a \mapsto i$.
- On note $a_1 \hookrightarrow a_2$ pour $(a_1 \mapsto a_2) * \texttt{true}$.
- On note $a \mapsto (a_1, \dots, a_n)$ pour

$$(a \mapsto a_1) * (a+1 \mapsto a_2) * \cdots * (a+n-1 \mapsto a_n).$$

▷ On note $[A]$ pour $A \wedge \texttt{emp}$.

Ces notations signifient respectivement :

- ▷ La première formule dit qu'une case mémoire est allouée en a (ou plutôt que si a s'évalue en k, alors il y a une case mémoire allouée en k).
- ▷ La seconde formule dit que la case mémoire a_1 est allouée, et contient a_2 quelque part dans le tas.
- ▷ La troisième formule dit que les n cases mémoires suivant a contiennent respectivement a_1, puis a_2, *etc.*
- ▷ La quatrième formule permet de ne pas parler du tas. Intuitivement A « observe » uniquement la composante σ et alors A est une formule de la logique de Hoare.

9.3 Triplets de Hoare pour la logique de séparation.

On rappelle que $\{A\}c\{A'\}$ est définir avec A, A' closes. La validité d'un triplet de Hoare en logique de séparation est défini par $\models \{A\}c\{A'\}$ si et seulement si dès que $\sigma, h \models A$ et $c, \sigma, h \Downarrow \sigma', h'$ alors $\sigma', h' \models A'$.

Parmi les règles d'inférences pour la logique de séparation, il y a la « *frame rule* » ou « *règle d'encadrement* » :

$$\frac{\vdash \{A\}c\{A'\}}{\vdash \{A * B\}c\{A' * B\}.} \quad \text{aucune variable } x \in \mathsf{vars}(B) \text{ n'est modifiée par } c$$

Elle permet de *zoomer*, et de ce concentrer sur un comportent local.

Les règles suivantes définissent une logique sur les commandes de IMP

avec tas.

$$\frac{}{\vdash \{\texttt{emp}\} x := \texttt{alloc}(k) \{x \mapsto (0, \ldots, 0)\}}$$

$$\frac{}{\vdash \{a \mapsto \_\}\texttt{free}(a)\{\texttt{emp}\}} \qquad \frac{}{\vdash \{a_1 \mapsto \_\}[a_1] := a_2 \{a_1 \mapsto a_2\}}$$

$$\frac{}{\vdash \{[x = x_0] * (a \mapsto x_1)\} x := [a] \{[x = x_1] * (a[x_0/x] \mapsto x_1)\}}$$

10. Réécriture.

Sommaire.

Définition 10.1. Soit $\to$ une relation binaire sur un ensemble E. Le 2-uplet $(E, \to)$ est un *SRA*, pour *système de réécriture abstraite*.

Soit $x_0 \in E$. Une *divergence* issue de x_0 est une suite $(x_i)_{i\in\mathbb{N}}$ telle que, pour tout i, on a $x_i \to x_{i+1}$.

La relation $\to$ est *terminante* ou *termine* si et seulement si, quel que soit $x \in E$, il n'y a pas de divergence issue de x.

La relation $\to$ diverge s'il existe une divergence.

Exemple 10.1. En général, une relation réflexive est divergente.

Théorème 10.1. Une relation $(E, \to)$ est terminante si et seulement si elle satisfait le *principe d'induction bien fondée (PIBF)* suivant :

Pour tout prédicat $\mathscr{P}$ sur E, si pour tout $x \in E$

$$\Big[\forall y \in E, x \to y \text{ implique } \mathscr{P}(y)\Big] \text{ implique } \mathscr{P}(x)$$

alors, pour tout $x \in E$, $\mathscr{P}(x)$.

En particulier, dans le principe d'induction bien fondée, on demande que les feuilles (les éléments sans successeurs) vérifient le prédicat.

Preuve. ▷ « PIBF $\implies$ terminaison ». Montrons que, quel que soit $x \in E$,

$$\mathscr{P}(x) \ : \ \text{« il n'y a pas de divergence issue de } x \text{ »}.$$

Soit $\text{Next}(x) = \{y \in E \mid x \to y\}$. On suppose que, pour tout $y \in \text{Next}(x)$, on a $\mathscr{P}(y)$. On en déduit $\mathscr{P}(x)$ car, sinon, une divergence ne passerait pas par $y \in \text{Next}(x)$. Par le principe d'induction bien fondée, on en déduit

$$\forall x \in E, \mathscr{P}(x),$$

autrement dit, la relation $\to$ termine.

▷ « $\neg$PIBF $\implies$ diverge », par contraposée. On suppose qu'il existe un prédicat $\mathscr{P}$ tel que,

$$\forall x, (\forall y, x \to y \text{ implique } \mathscr{P}(y)) \text{ implique } \mathscr{P}(x),$$

et que l'on n'ait pas, $\forall x \in E, \mathscr{P}(x)$ autrement dit qu'il existe $x_0 \in E$ tel que $\neg\mathscr{P}(x)$.

Intéressons-nous à $\text{Next}(x_0) = \{y \in E \mid x_0 \to y\}$. Si, pour tout $y \in \text{Next}(x_0)$ on a $\mathscr{P}(y)$ alors par hypothèse $\mathscr{P}(x_0)$, ce qui est impossible. Ainsi, il existe $x_1 \in \text{Next}(x_0)$ tel que $\neg\mathscr{P}(x_1)$. On itère ce raisonnement, ceci crée notre divergence.

□

Remarque 10.1. L'induction bien fondée s'appelle aussi l'induction *noethérienne*, en référence à Emmy Noether, mathématicienne allemande du IX–Xème siècle.

Une application de ce principe d'induction est le *lemme de König*.

Définition 10.2.
- ▷ Un arbre est *fini* s'il a un nombre fini de nœuds (*infini* sinon).
- ▷ Un arbre est à *branchement fini* si tout nœud a un nombre fini d'enfants immédiats.
- ▷ Une branche est *infinie* si elle contient un nombre infini de nœuds.

Lemme 10.1 (Lemme de König)**.** Si un arbre est à branchement fini est infini alors il contient une branche infinie.

Preuve. On considère E l'ensemble des nœuds de l'arbre, et on définit la relation $\to$ par : on a $x \to y$ si y est enfant immédiat de x. On montre qu'un arbre à branchement fini sans branche infinie (*i.e.* la relation $\to$ termine) est fini. On choisit la propriété $\mathcal{P}(x)$: « le sous-arbre enraciné en x est fini. »

Montrons que, quel que soit x, $\mathcal{P}(x)$ et pour ce faire, utilisons le principe d'induction bien fondée puisque la relation $\to$ termine. On doit montrer que, si $\forall y \in \mathrm{Next}(x), \mathcal{P}(y)$ implique $\mathcal{P}(x)$. Ceci est vrai car l'embranchement est fini. □

10.1 Liens avec les définitions inductives.

On considère E l'ensemble inductif défini par la grammaire suivante :

$$t ::= \mathtt{F} \mid \mathtt{N}(t_1, k, t_2).$$

C'est aussi le plus petit point fixe de l'opérateur f associé (par le théorème de Knaster–Tarski).

On définit la relation $\to$ binaire sur E par : on a $x \to y$ si et seulement si on a $x = \mathtt{N}(y, k, z)$ ou $x = \mathtt{N}(z, k, y)$.

On sait que la relation $\to$ termine. En effet, l'ensemble des arbres finis est un point fixe de la fonction f, donc E ne contient que des arbres finis.

Le principe d'induction bien fondée nous dit que, pour $\mathscr{P}$ un prédicat sur E, pour montrer $\forall x, \mathscr{P}(x)$, il suffit de montrer que, quel que soit x, si $(\forall y, x \to y$ implique $\mathscr{P}(y))$ alors $\mathscr{P}(x)$. Autrement dit, il suffit de montrer que $\mathscr{P}(\mathtt{E})$ puis de montrer que, si $\mathscr{P}(t_1)$ et $\mathscr{P}(t_2)$ alors on a que $\mathscr{P}(\mathtt{N}(t_1, k, t_2))$.

On retrouve le principe d'induction usuel.

Ce même raisonnement, on peut le réaliser quel que soit l'ensemble inductif, car la relation de « sous-élément » termine toujours puisque il n'y a que des éléments finis dans l'ensemble inductif.

10.2 Établir la terminaison.

Théorème 10.2. Soient $(B, >)$ un SRA terminant, et $(A, \to)$ un SRA. Soit $\varphi : A \to B$ un *plongement*, c'est à dire une application vérifiant

$$\forall a, a' \in A, \qquad a \to a' \text{ implique } \varphi(a) > \varphi(a').$$

Alors, la relation $\to$ termine.

Théorème 10.3. Soient $(A, \to_A)$ et $(B, \to_B)$ deux SRA.

Le *produit lexicographique* de $(A, \to_A)$ et $(B, \to_B)$ est le SRA, que l'on notera $(A \times B, \to_{A\times B})$, défini par

$$(a, b) \to_{A\times B} (a', b') \text{ ssi } \begin{cases} (1)\ a \to_A a' \text{ (et } b' \text{ quelconque)} \\ \qquad \text{ou} \\ (2)\ a = a' \text{ et } b \to_B b' \end{cases}.$$

Alors, les relations $(A, \to_A)$ et $(B, \to_B)$ terminent si et seulement si la relation $(A \times B, \to_{A\times B})$ termine.

Preuve. ▷ « $\Longrightarrow$ ». Supposons qu'il existe une divergence pour $(A \times B, \to_{A\times B})$:

$$(a_0, b_0) \to_{A\times B} (a_1, b_1) \to_{A\times B} (a_2, b_2) \to_{A\times B} \cdots .$$

Dans cette divergence,

- soit on a utilisé (1) une infinité de fois, et alors en projetant sur la première composante et en ne conservant que les fois où l'on utilise (1), on obtient une divergence $\to_A$;
- soit on a utilisé (1) un nombre fini de fois, et alors à partir d'un certain rang N, pour tout $i \geq N$, on a l'égalité $a_i = a_N$, et donc on obtient une divergence pour $\to_B$:

$$b_N \to_B b_{N+1} \to_B b_{N+2} \to \cdots .$$

▷ « $\Longleftarrow$ ». On montre que, si on a une divergence pour $\to_A$ alors on a une divergence pour $\to_{A\times B}$ (on utilise (1) une infinité de fois) ; puis que si on a une divergence pour $\to_B$ alors on a une divergence pour $\to_{A\times B}$ (on utilise (2) une infinité de fois).

□

10.3 Application à l'algorithme d'unification.

On note $(\mathscr{P}, \sigma) \to (\mathscr{P}', \sigma')$ la relation définie par l'algorithme d'unification (on néglige le cas où $(\mathscr{P}, \sigma) \to \bot$).

On note $|\mathscr{P}|$ la somme des tailles (vues comme des arbres) des contraintes de $\mathscr{P}$ et $|\mathsf{Vars}\ \mathscr{P}|$ le nombre de variables.

On définit $\varphi : (\mathscr{P}, \sigma) \mapsto (|\mathsf{Vars}\ \mathscr{P}|, |\mathscr{P}|)$.

Rappelons la définition de la relation $\to$ dans l'algorithme d'unification :

1. $(\{f(t_1,\dots,t_k) \stackrel{?}{=} f(u_1,\dots,u_n) \sqcup \mathscr{P}, \sigma\}) \to (\{t_1 \stackrel{?}{=} u_1,\dots,t_k \stackrel{?}{=} u_k\} \cup \mathscr{P}, \sigma)$;
2. $(\{f(t_1,\dots,t_k) \stackrel{?}{=} g(u_1,\dots,u_n) \sqcup \mathscr{P}, \sigma\}) \to \bot$ si $f \neq g$;
3. $(\{X \stackrel{?}{=} t\} \sqcup \mathscr{P}, \sigma) \to (\mathscr{P}[t/X], [t/X] \circ \sigma)$ où $X \notin \mathsf{Vars}(t)$;
4. $(\{X \stackrel{?}{=} t\} \sqcup \mathscr{P}, \sigma) \to \bot$ si $X \in \mathsf{Vars}(t)$ et $t \neq X$;
5. $(\{X \stackrel{?}{=} X\} \sqcup \mathscr{P}, \sigma) \to (\mathscr{P}, \sigma)$.

Appliquons le plongement pour montrer que $\to$ termine. On s'appuie sur le fait que le produit $(\mathbb{N}, >) \times (\mathbb{N}, >)$ est terminant (produit lexicographique).

Dans 1, $|\mathsf{Vars}\ \mathscr{P}|$ ne change pas et $|\mathscr{P}|$ diminue. Puis dans 3, $|\mathsf{Vars}\ \mathscr{P}|$ diminue. Et dans 5, on a $|\mathsf{Vars}\ \mathscr{P}|$ qui décroit ou ne change pas, mais $|\mathscr{P}|$ diminue. Dans les autres cas, on arrive, soit sur $\bot$.

On en conclut que l'algorithme d'unification termine.

10.4 Multiensembles.

Définition 10.3. Soit A un ensemble. Un *multiensemble* sur A est la donnée d'une fonction $M : A \to \mathbb{N}$. Un multiensemble M est *fini* si $\{a \in A \mid M(a) > 0\}$ est fini.

Le multiensemble vide, noté $\emptyset$, vaut $a \mapsto 0$.

Pour deux multiensembles M_1 et M_2 sur A, on définit

- $(M_1 \cup M_2)(a) = M_1(a) + M_2(a)$;
- $(M_1 \ominus M_2)(a) = M_1(a) \ominus M_2(a)$ où l'on a $(n+k) \ominus n = k$ mais $n \ominus (n+k) = 0$.

On note $M_1 \subseteq M_2$ si, pour tout $a \in A$, on a $M_1(a) \leq M_2(a)$.

La *taille* de M est $|M| = \sum_{a \in A} M(a)$.

On note $x \in M$ dès lors que $x \in A$ et que $M(x) > 0$.

Exemple 10.2. Si on lit $\{1,1,1,2,3,4,3,5\}$ comme un multiensemble M, on obtient que $M(1)=3$, et $M(2)=1$, et $M(3)=2$, et $M(4)=1$, et $M(5)=1$, et finalement pour tout autre entier n, $M(n)=0$.

Définition 10.4 (Extension multiensemble.)**.** Soit $(A,>)$ un SRA. On lui associe une relation notée $>_{\text{mul}}$ définie sur les multiensembles *finis* sur A en définissant $M >_{\text{mul}} N$ si et seulement s'il existe X,Y deux multiensembles sur A tels que

- $\emptyset \neq X \subseteq M$;
- $N = (M \ominus X) \cup Y$ [1]
- $\forall y \in Y, \exists x \in X, x > y$.

Les multiensembles X et Y sont les « témoins » de $M >_{\text{mul}} N$.

Exemple 10.3. Dans $(\mathbb{N},>)$, on a

$$\{1,2,\underbrace{5}_{X}\} >_{\text{mul}} \{1,2,\underbrace{4,4,4,4,3,3,3,3,3}_{Y}\}.$$

Théorème 10.4. La relation $>$ termine si et seulement si $>_{\text{mul}}$ termine.

Preuve.
- « $\Longleftarrow$ ». Une divergence de $>$ induit une divergence de $>_{\text{mul}}$.
- « $\Longrightarrow$ ». On se donne une divergence pour $>_{\text{mul}}$:

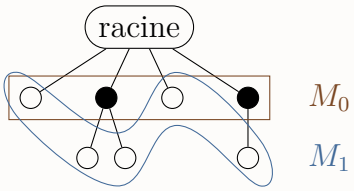
$$M_0 >_{\text{mul}} M_1 >_{\text{mul}} M_2 >_{\text{mul}} \cdots,$$

et on montre que $>$ diverge À chaque $M_i >_{\text{mul}} M_{i+1}$ correspondent X_i et Y_i suivant la définition de $>_{\text{mul}}$.

1. C'est ici la soustraction usuelle : il n'y a pas de soustraction qui « pose problème ».

On sait qu'il y a une infinité de i tel que $Y_i \neq \emptyset$. En effet, si au bout d'un moment Y_i est toujours vide alors $|M_i|$ décroit strictement, impossible.

Représentons cela sur un arbre.



On itère le parcours en obtenant un arbre à branchement fini, qui est infini (observation du dessin) donc par le lemme de König il a une branche infinie. Par construction, un enfant de a correspond à $a > a'$, d'où divergence pour $>$.

□

Théorème 10.5 (Récursion bien fondée)**.** On appelle *fonction* de A_1 dans A_2 la donnée d'une relation fonctionnelle totale incluse dans $A_1 \times A_2$. On note $f : A_1 \to A_2$.

On se donne $(A, >)$ un SRA **terminant**.

Pour $a \in A$, on note

- $\mathrm{Pred}(a) := \{a' \mid a > a'\}$;[2]
- $\mathrm{Pred}^+(a) := \{a' \mid a >^+ a'\}$;
- $\mathrm{Pred}^\star(a) := \{a' \mid a >^\star a'\} = \mathrm{Pred}^+(a) \cup \{a\}$.[3]

Pour $f : A \to B$ et $A' \subseteq A$, on note $f \restriction A' := \{(a, f(a)) \mid a \in A'\}$.

On se donne une fonction F telle que, pour tout $a \in A$, et tout $h \in \mathrm{Pred}(a) \to B$, on a $F(a, h) \in B$. Alors, il existe une unique fonction $f : A \to B$ telle que

$$\forall a \in A, f(a) = F(a, f \restriction (\mathrm{Pred}(a))).$$

Preuve. **Unicité.** Soient f, g telles que, pour tout $a \in A$, on ait

- ▷ $f(a) = F(a, f \restriction \mathrm{Pred}(a))$;
- ▷ $g(a) = F(a, g \restriction \mathrm{Pred}(a))$.

Montrons que $\forall a \in A, f(a) = g(a)$ par induction bien fondée (car $>$ termine).

Soit $a \in A$. On suppose $\forall b \in \mathrm{Pred}(b), f(b) = g(b)$ l'hypothèse d'induction. Alors, $f \restriction \mathrm{Pred}(a) = g \restriction \mathrm{Pred}(a)$, et donc $f(a) = g(a)$.

Existence. On montre, par induction bien fondée, que $\mathscr{P}(a)$, la propriété ci-dessous, est vraie quel que soit $a \in A$:

$$\exists f_a : \mathrm{Pred}^\star(a) \to B, \forall b \in \mathrm{Pred}^\star(a), f_a(b) = F(b, f \restriction \mathrm{Pred}(b)).$$

Soit $a \in A$. On suppose $\forall b \in \mathrm{Pred}(a), \mathscr{P}(b)$ (f_b existe). Posons la relation $h := \bigcup_{b \in \mathrm{Pred}(a)} f_b$. La relation h est

- ▷ fonctionnelle (*c.f.* preuve d'unicité) ;
- ▷ définie sur $\mathrm{Pred}^+(a)$.

On conclut la preuve en posant

$$f_a := h \cup \{(a, F(a, h))\}.$$

□

Exemple 10.4. Pour définir une fonction `length` : `nlist` → `nat`. La relation $>$ sur `nlist` où $k :: \ell > \ell$ est une relation bien fondée. On pose la fonction $F(\ell, h)$ par :

```
let F ℓ h = match ℓ with
| [] -> 0
```

2. On le notait Next avant, mais le successeur pour $>$ est un prédécesseur pour $<$ (ce qui est plus usuel).

3. On rappelle que l'on note $\mathscr{R}^+$ et respectivement $\mathscr{R}^\star$ la clôture transitive, et respectivement la clôture réflexive et transitive.

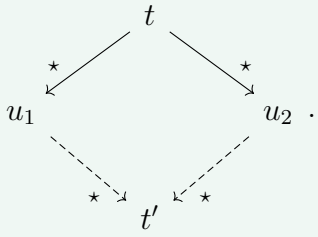
```
    | x :: xs -> 1 + h(xs)
```

Code 10.1 | *Définition récursive bien fondée de* `length`

où l'on a ici `xs` $\in$ Pred(`x :: xs`).

10.5 Confluence.

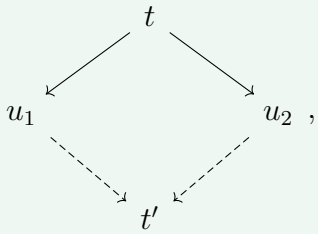
Définition 10.5. Soit $(A, \to)$ un SRA. On dit que $\to$ est *confluente* en $t \in A$ si, dès que $t \to^\star u_1$ et $t \to^\star u_2$ il existe t' tel que $u_1 \to^\star t'$ et $u_2 \to^\star t'$.



Les flèches en pointillés représentent l'existence.

On dit que $\to$ est *confluente* si $\to$ est confluente en tout $a \in A$.

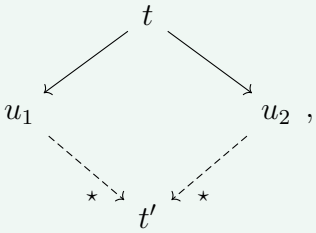
La propriété du diamant correspond au diagramme ci-dessous :



c'est-à-dire si $t \to u_1$ et $t \to u_2$ alors il existe t' tel que $u_1 \to t'$ et $u_2 \to t'$.

La propriété de *confluence locale* correspond au diagramme ci-

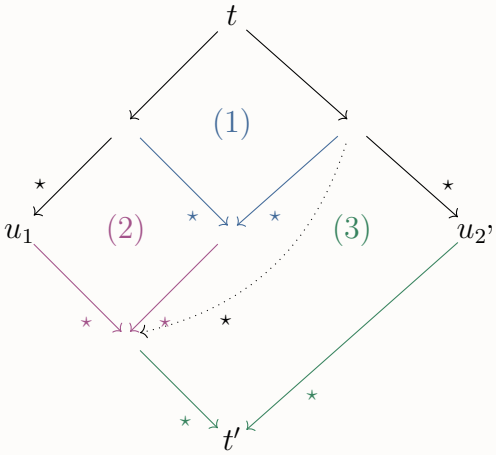
dessous :



c'est-à-dire si $t \to u_1$ et $t \to u_2$ alors il existe t' tel que $u_1 \to \star t'$ et $u_2 \to \star t'$.

Lemme 10.2 (Lemme de Newman)**.** Soit $(A, \to)$ terminante et localement confluente. Alors, $(A, \to)$ confluente.

Preuve. On montre que, quel que soit $t \in A$, que $\to$ est confluente en t par induction bien fondée. On suppose que quel que soit t'' tel que $t \to t'$, alors la relation $\to$ est confluente en t' ; Montrons la confluence en t. Soit $t \to^\star u_1$ et $t \to^\star u_2$. Si $t = t_1$ ou $t = u_2$, c'est immédiat. On suppose donc



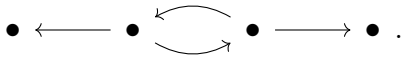
où l'on utilise

- ▷ (1) par confluence locale ;

- ▷ (2) par hypothèse d'induction ;
- ▷ (3) par hypothèse d'induction.

□

Remarque 10.2. L'hypothèse de relation terminante est cruciale. En effet, la relation ci-dessous est un contre exemple : la relation $\to$ est non terminante, localement confluente mais pas confluente.



10.6 Système de réécriture de mots.

Les systèmes de réécriture de mots sont parmi les systèmes de réécriture les plus simples. On peut définir un système de réécriture de termes, où « terme » correspond à « terme » dans la partie Typage, mais on ne les étudiera pas dans ce cours.

Définition 10.6 (*c.f.* cours de FDI)**.** On se donne Σ un ensemble de lettres. On note :

- ▷ $\Sigma^\star$ l'ensemble des mots sur Σ,
- ▷ ε le mot vide,
- ▷ uv la concaténation de u et v (avec $u, v \in \Sigma^\star$).

Définition 10.7. Un *SRM* (*système de réécriture de mots*) sur Σ est donné par un ensemble $\mathscr{R}$ de couples de mots sur Σ noté généralement

$$\mathscr{R} = \{(\ell, r) \mid \ell \neq \varepsilon\}.$$

Le SRM $\mathscr{R}$ détermine une relation binaire sur $\Sigma^\star$ définie par $u \to_{\mathscr{R}} v$ dès lors qu'il existe $(x, y) \in (\Sigma^\star)^2$ et $(\ell, r) \in \mathscr{R}$ tels que l'on ait $u = x\ell y$ et $v = xry$.

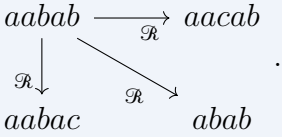
Exemple 10.5. Sur $\Sigma = \{a, b, c\}$ et $\mathscr{R}_0$ donné par

$$\begin{aligned} ab &\to ac \\ ccc &\to bb \\ aa &\to a, \end{aligned}$$

autrement dit

$$\mathscr{R}_0 = \{(ab, ac), (ccc, bb), (aa, a)\},$$

et alors



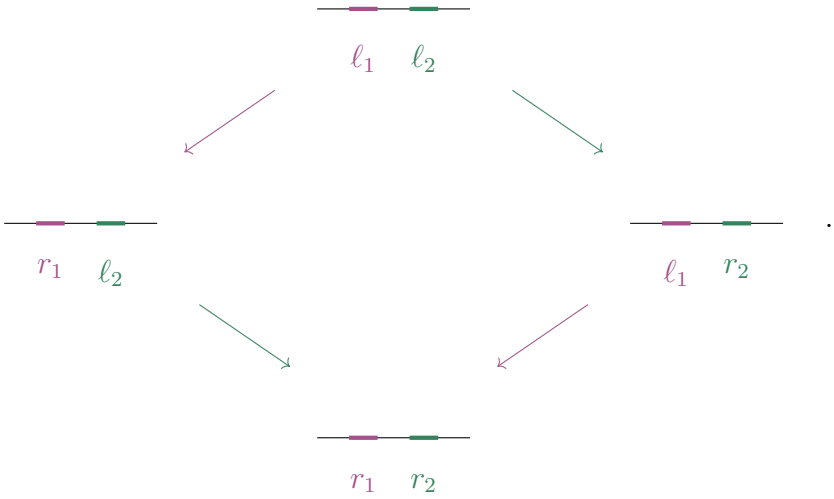
.

La relation $\to_{\mathscr{R}_0}$ est-elle terminante ? Oui, il suffit de réaliser un plongement sur le produit lexicographique donné par $\varphi : u \mapsto (|u|, \#_b(u))$, où $\#_b(u)$ est le nombre de b dans u et $|u|$ est la taille de u.

10.6.1 Étude de la confluence locale dans les SRM.

Soient $(\ell_1, r_1), (\ell_2, r_2) \in \mathscr{R}$ tels que $t \to_{\mathscr{R}} u_1$ avec (ℓ_1, r_1) et $t \to_{\mathscr{R}}$ avec (ℓ_2, r_2).

***1er cas* : indépendance.** On a la propriété du diamant.



***2ème cas* : inclusion.** S'il existe (ℓ, ℓ') tel que $\ell_1 = \ell\ell_2\ell'$, alors, on a que $\ell_1 \to_{\mathscr{R}} r_1$ et $\ell_2 \to_{\mathscr{R}} \ell r_2 \ell'$.

Peut-on confluer ? Ce n'est pas évident.

***3ème cas* : overlap.** S'il existe (ℓ, ℓ', ℓ'') tel que $\ell_1 = \ell\ell'$ et $\ell_2 = \ell'\ell''$, alors $t \to r_1\ell''$ et $t \to \ell r_2$.

Peut-on confluer ? Ce n'est pas évident.

Définition 10.8. Soit $\mathscr{R}$ un SRM sur Σ. Soient $(\ell_1, r_1), (\ell_2, r_2) \in \mathscr{R}$. Supposons qu'il existe des mots z, v_1, v_2 tels que

- $|z| < |\ell_1|$;
- $\ell_1 v_1 = z\ell_2 v_2$;
- $\varepsilon \in \{v_1, v_2\}$.

On dit alors que $\{r_1 v_1, z r_2 v_2\}$ est une paire critique de $\mathscr{R}$.

Exemple 10.6. Avec $\Sigma = \{a, b, c\}$ avec le SRM $\mathscr{R}$ défini par

$$\begin{array}{lrcl} (1) & aba & \to & abc \\ (2) & ab & \to & ba \end{array},$$

on se demande si

- $\to_{\mathscr{R}}$ termine ? Oui avec le nombre de a et le nombre d'inversions a–b.
- quelles sont les paires critiques ? On procède cas par cas :
  - (1) avec (2), on a $aba \to abc$ et $aba \to baa$ donc $\{abc, baa\}$ est critique ;
  - (1) avec (1), on a $ababa \to abcba$ et $ababa \to ababc$ donc $\{abcba, ababc\}$ est critique ;
  - (2) avec (2), il n'y a pas de paires critiques.

Pourquoi s'intéresser aux paires critiques ? Et bien, cela prend son sens grâce au théorème ci-dessous.

Théorème 10.6. La relation $\to_{\mathscr{R}}$ est localement confluente si et seulement si toutes ses paires critiques sont *joignables*, c'est-à-dire que, pour $\{u_1, u_2\}$ critique, il existe t' tel que $u_1 \to^\star t'$ et $u_2 \to^\star t'$.

Deuxième partie

Logique.

Introduction.

Dans ce cours, on s'intéressera à quatre thèmes :

- ▷ la théorie des modèles (▷ les « vraies » mathématiques) ;
- ▷ la théorie de la démonstration (▷ les preuves) ;
- ▷ la théorie des ensembles (▷ les objets) ;
- ▷ les théorèmes de Gödel (▷ les limites).

On ne s'intéressera pas à la calculabilité, car déjà vue en cours de FDI. Ce cours peut être utile à ceux préparant l'agrégation d'informatique.

11. Le calcul propositionnel.

Le *calcul propositionnel*, c'est la « grammaire » de la logique. Dans ce chapitre, on s'intéressera à

1. la construction des formules (▷ la syntaxe) ;
2. la sémantique et les théorèmes de compacité (▷ la compacité sémantique).

11.1 Syntaxe.

Définition 11.1. Le *langage*, ou *alphabet*, est un ensemble d'éléments fini ou pas. Les éléments sont les *lettres*, et les suites finies sont les *mots*.

Définition 11.2. On choisit l'alphabet :

- ▷ $\mathscr{P} = \{x_0, x_1, \ldots\}$ des variables propositionnelles ;
- ▷ un ensemble de *connecteurs* ou *symboles logiques*, défini par $\{\neg, \vee, \wedge, \to, \leftrightarrow\}$, il n'y a pas $\exists$ et $\forall$ pour l'instant.
- ▷ les parenthèses $\{(,)\}$.

Les formules logiques sont des mots. On les fabrique avec des briques de base (les variables) et des opérations de construction : si F_1 et F_2 sont deux formules, alors $\neg F$, $(F_1 \vee F_2)$, $(F_1 \wedge F_2)$, $(F_1 \to F_2)$ et $(F_1 \leftrightarrow F_2)$ aussi.

Définition 11.3 (« par le haut », « mathématique »)**.** L'ensemble $\mathscr{F}$ des formules du calcul propositionnel construit sur $\mathscr{P}$ est le plus petit ensemble contenant $\mathscr{P}$ et stable par les opérations de construction.

Définition 11.4 (« par le bas », « informatique »)**.** L'ensemble $\mathscr{F}$ des formules logiques du calcul propositionnel sur $\mathscr{P}$ est défini par

- $\mathscr{F}_0 = \mathscr{P}$;
- $\mathscr{F}_{n+1} = \mathscr{F}_n \cup \left\{ \begin{array}{c} \neg F_1 \\ (F_1 \vee F_2) \\ (F_1 \wedge F_2) \\ (F_1 \to F_2) \\ (F_1 \leftrightarrow F_2) \end{array} \middle| \; F_1, F_2 \in \mathscr{F}_n \right\}$

puis on pose $\mathscr{F} = \bigcup_{n \in \mathbb{N}} \mathscr{F}_n$.

On peut montrer l'équivalence des deux définitions.

Théorème 11.1 (Lecture unique)**.** Toute formule $G \in \mathscr{F}$ vérifie une et une seule de ces propriétés :

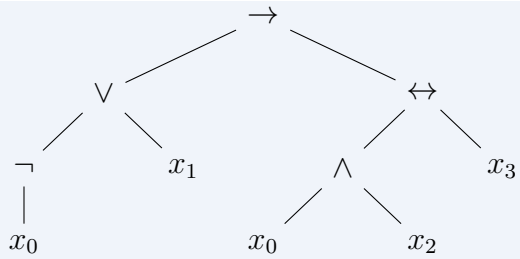
- $G \in \mathscr{P}$;
- il existe $F \in \mathscr{F}$ telle que $G = \neg F$;
- il existe $F_1, F_2 \in \mathscr{F}$ telle que $G = (F_1 \vee F_2)$;
- il existe $F_1, F_2 \in \mathscr{F}$ telle que $G = (F_1 \wedge F_2)$;
- il existe $F_1, F_2 \in \mathscr{F}$ telle que $G = (F_1 \to F_2)$;
- il existe $F_1, F_2 \in \mathscr{F}$ telle que $G = (F_1 \leftrightarrow F_2)$.

Preuve. En exercice. □

Corollaire 11.1. Il y a une bijection entre les formules et les arbres dont

- les feuilles sont étiquetées par des variables ;
- les nœuds internes sont étiquetés par des connecteurs ;
- ceux étiquetés par $\neg$ ont un fils, les autres deux.

Exemple 11.1. La formule $((\neg x_0 \vee x_1) \to ((x_0 \wedge x_2) \leftrightarrow x_3))$ correspond à l'arbre



11.2 Sémantique.

Lemme 11.1. Soit ν une fonction de $\mathscr{P}$ dans $\{0,1\}$ appelée *valuation*. Alors ν s'étend de manière unique en une fonction $\bar{\nu}$ de $\mathscr{F}$ dans $\{0,1\}$ telle que

- sur $\mathscr{P}$, $\nu = \bar{\nu}$;
- si $F, G \in \mathscr{F}$ sont des formules alors
  - $\bar{\nu}(\neg F) = 1 - \bar{\nu}(F)$;
  - $\bar{\nu}(F \vee G) = 1$ ssi $\bar{\nu}(F) = 1$ ou[1] $\bar{\nu}(G) = 1$;
  - $\bar{\nu}(F \wedge G) = \bar{\nu}(F) \times \bar{\nu}(G)$;
  - $\bar{\nu}(F \to G) = 1$ ssi $\bar{\nu}(G) = 1$ ou $\bar{\nu}(F) = 0$;
  - $\bar{\nu}(F \leftrightarrow G) = 1$ ssi $\bar{\nu}(F) = \bar{\nu}(G)$.

Par abus de notations, on notera ν pour $\bar{\nu}$ par la suite.

Preuve. **Existence.** On définit en utilisant le lemme de lecture unique, et par induction sur $\mathscr{F}$:

- $\bar{\nu}$ est définie sur $\mathscr{F}_0 = \mathscr{P}$;
- si $\bar{\nu}$ est définie sur $\mathscr{F}_n$ alors pour $F \in \mathscr{F}_{n+1}$, on a la disjonction de cas
  - si $F = \neg G$ avec $G \in \mathscr{F}_n$, et on définit $\bar{\nu}(F) = 1 - \bar{\nu}(F_1)$;
  - *etc* pour les autres cas.

Unicité. On montre que si $\lambda = \nu$ sur $\mathscr{P}$ alors $\bar{\lambda} = \bar{\nu}$ si $\bar{\lambda}$ et $\bar{\nu}$

1. C'est un « ou » inclusif : on peut avoir les deux (ce qui est très différent du « ou » exclusif dans la langue française).

vérifient les égalités précédents.

□

Exemple 11.2 (Table de vérité)**.** Pour la formule

$$F = ((x_1 \to x_2) \to (x_2 \to x_1)),$$

on construit la table

x_1	0	0	1	1
x_2	0	1	0	1
$x_1 \to x_2$	1	1	0	1
$x_2 \to x_1$	1	0	1	1
F	1	0	1	1

Définition 11.5.

- ▷ Une formule F est dite *satisfaite par une valuation* ν si $\nu(F) = 1$.
- ▷ Une *tautologie* est une formule satisfaite pour toutes les valuations.
- ▷ Un ensemble $\mathscr{E}$ de formules est *satisfiable* s'il existe une valuation qui satisfait toutes les formules de $\mathscr{E}$.
- ▷ Un ensemble $\mathscr{E}$ de formules est *finiment satisfiable* si tout sous-ensemble fini de $\mathscr{E}$ est satisfiable.
- ▷ Une formule F est *conséquence sémantique* d'un ensemble de formules $\mathscr{E}$ si toute valuation qui satisfait $\mathscr{E}$ satisfait F.
- ▷ Un ensemble de formules $\mathscr{E}$ est *contradictoire* s'il n'est pas satisfiable.
- ▷ Un ensemble de formules $\mathscr{E}$ est *finiment contradictoire* s'il existe un sous-ensemble fini contradictoire de $\mathscr{E}$.

Théorème 11.2 (compacité du calcul propositionnel)**.** On donne trois énoncés équivalents (équivalence des trois énoncés laissée en exercice) du théorème de compacité du calcul propositionnel.

Version 1. Un ensemble de formules $\mathscr{E}$ est satisfiable si et seulement s'il est finiment satisfiable.

Version 2. Un ensemble de formules $\mathscr{E}$ est contradictoire si et seulement s'il est finiment contradictoire.

Version 3. Pour tout ensemble $\mathscr{E}$ de formules du calcul propositionnel, et toute formule F, F est conséquence sémantique de $\mathscr{E}$ si et seulement si F est conséquence sémantique d'un sous-ensemble fini de $\mathscr{E}$.

Preuve. Dans le cas où $\mathscr{P} = \{x_0, x_1, \ldots\}$ est au plus dénombrable (le cas non dénombrable sera traité après). On démontre le cas « difficile » de la version 1 (*i.e.* finiment satisfiable implique satisfiable). Soit $\mathscr{E}$ un ensemble de formules finiment satisfiable. On construit par récurrence une valuation ν qui satisfasse $\mathscr{E}$ par récurrence : on construit $\varepsilon_0, \ldots, \varepsilon_n, \ldots$ tels que $\nu(x_0) = \varepsilon_0, \ldots, \nu(x_n) = \varepsilon_n, \ldots$

- ▷ Cas de base. On définit la valeur de ε_n pour $x_0 \in \mathscr{P}$.
  1. soit, pour tout sous-ensemble fini B de $\mathscr{E}$, il existe une valuation λ qui satisfait B avec $\lambda(x_0) = 0$;
  2. soit, il existe un sous-ensemble fini B_0 de $\mathscr{E}$, pour toute valuation λ qui satisfait B_0, on a $\lambda(x_0) = 1$.

  Si on est dans le cas 1, on pose $\varepsilon_0 = 0$, et sinon (cas 2) on pose $\varepsilon_0 = 1$.
- ▷ Cas de récurrence. On montre, par récurrence sur n, la propriété suivante :

  > il existe une suite $\varepsilon_0, \ldots, \varepsilon_n$ (que l'on étend, la suite ne change pas en fonction de n) de booléens telle que, pour tout sous-ensemble fini B de $\mathscr{E}$, il existe une valuation ν satisfaisant B et telle que $\nu(x_0) = \varepsilon_0$, …, et $\nu(x_n) = \varepsilon_n$.

  – Pour $n = 0$, soit on est dans le cas 1, et on prend $\varepsilon_0 = 0$ et on a la propriété ; soit on est dans le cas 2 ;, et on prend B un sous-ensemble fini de $\mathscr{E}$, alors $B \cup B_0$ est

un ensemble fini donc satisfiable par une valuation ν. La valuation satisfait B_0 donc $\nu(x_0) = 1$ et ν satisfait B. On a donc la propriété au rang 0.

– Hérédité. Par hypothèse de récurrence, on a une suite $\varepsilon_0, \dots, \varepsilon_n$.

1. Soit, pour tout sous-ensemble fini B de $\mathscr{E}$, il existe ν qui satisfait B et telle que $\nu(x_0) = \varepsilon_0$, …, $\nu(x_n) = \varepsilon_n$, et $\nu(x_{n+1}) = 0$. On pose $\varepsilon_{n+1} = 0$.
2. Soit il existe B_{n+1} un sous-ensemble fini de $\mathscr{E}$ tel que, pour toute valuation ν telle que ν satisfait B_{n+1} et $\nu(x_0) = \varepsilon_0, \dots, \nu(x_n) = \varepsilon_n$, on a $\nu(x_{n+1}) = 1$ et on pose $\varepsilon_{n+1} = 1$.

Montrons l'hérédité :

1. vrai par définition ;
2. soit B un sous-ensemble fini de $\mathscr{E}$. On considère $B \cup B_{n+1}$, soit ν telle que $\nu(x_0) = \varepsilon_0$, …, $\nu(x_n) = \varepsilon_n$. On a que ν satisfait B_{n+1} donc $\nu(x_{n+1}) = 1 = \varepsilon_{n+1}$ et ν satisfait B.

On a donc la propriété pour tout n.

Finalement, soit δ une valuation telle que, pour tout i, $\delta(x_i) = \varepsilon_i$. Montrons que δ satisfait $\mathscr{E}$. Soit $F \in \mathscr{E}$. On sait que F est un mot (fini), donc contient un ensemble fini de variables inclus dans $\{x_0, \dots, x_n\}$. D'après la propriété par récurrence au rang n, il existe une valuation ν qui satisfait F et telle que $\nu(x_0) = \varepsilon_0, \dots, \nu(x_n) = \varepsilon_n$, et donc ν et δ coïncident sur les variables de F. Donc (lemme simple), elles coïncident sur toutes les formules qui n'utilisent que ces variables. Donc, $\delta(F) = 1$, et on en conclut que δ satisfait $\mathscr{E}$. □

Dans le cas non-dénombrable, on utilise le *lemme de Zorn*, un équivalent de l'*axiome du choix*.

Définition 11.6. Un ensemble ordonné $(X, \mathscr{R})$ est inductif si pour

tout sous-ensemble Y de X totalement ordonné par $\mathscr{R}$ (*i.e.* une chaîne) admet un majorant dans X.

Remarque 11.1. On considère ici un majorant et non un plus grand élément (un maximum).

Exemple 11.3.
1. Dans le cas $(\mathscr{P}(X), \subseteq)$, le majorant est l'union des parties de la chaîne, il est donc inductif.
2. Dans le cas $(\mathbb{R}, \leq)$, il n'est pas inductif car $\mathbb{R}$ n'a pas de majorant dans $\mathbb{R}$.

Lemme 11.2 (Lemme de Zorn)**.** Si $(X, \mathscr{R})$ est un ensemble ordonné inductif non-vide, il admet au moins un élément maximal.

Remarque 11.2. Un élément maximal n'est pas nécessairement le plus grand.

Preuve. Soit $\mathscr{C}$ un ensemble de formules finiment satisfiable, et $\mathscr{P}$ un ensemble de variables. On note $\mathscr{V}$ l'ensemble des valuations partielles prolongeables pour toute partie finie $\mathscr{C}$ de $\mathscr{C}$ en une valuation satisfaisant $\mathscr{C}$. C'est-à-dire :

$$\mathscr{V} := \left\{ \varphi \in \bigcup_{X \subseteq \mathscr{P}} \{0,1\}^X \;\middle|\; \forall \mathscr{C} \in \wp_{\mathrm{f}}(\mathscr{C}), \exists \delta \in \{0,1\}^{\mathscr{P}}, \begin{array}{l} \delta_{|\operatorname{dom}\varphi} = \varphi \\ \forall F \in \mathscr{C}, \delta(F) = 1 \end{array} \right\}.$$

L'ensemble $\mathscr{V}$ est non-vide car contient l'application vide de $\{0,1\}^{\emptyset}$ car $\mathscr{C}$ est finiment satisfiable. On définit la relation d'ordre $\preccurlyeq$ sur $\mathscr{V}$ par :

$$\varphi \preccurlyeq \psi \quad \text{ssi} \quad \psi \text{ prolonge } \varphi.$$

Montrons que $(\mathscr{V}, \preccurlyeq)$ est inductif. Soit $\mathscr{C}$ une chaîne de $\mathscr{V}$ et construisons un majorant de $\mathscr{C}$. Soit λ la valuation partielle définie sur $\operatorname{dom}\lambda = \bigcup_{\varphi\in\mathscr{C}} \operatorname{dom}\varphi$, par : si $x_i \in \operatorname{dom}\lambda$ alors il existe $\varphi \in \mathscr{C}$ tel que $x_i \in \operatorname{dom}\varphi$ et on pose $\lambda(x_i) = \varphi(x_i)$.

La valuation λ est définie de manière unique, *i.e.* ne dépend pas du choix de φ. En effet, si $\varphi \in \mathscr{C}$ et $\psi \in \mathscr{C}$, avec $x_i \in \operatorname{dom}\varphi \cap \operatorname{dom}\psi$, alors on a $\varphi \preccurlyeq \psi$ ou $\psi \preccurlyeq \varphi$, donc $\varphi(x_i) = \psi(x_i)$.

Autrement dit, λ est la limite de $\mathscr{C}$. Montrons que $\lambda \in \mathscr{V}$. Soit B une partie finie de $\mathscr{C}$. On cherche μ qui prolonge λ et satisfait B. L'ensemble de formules B est fini, donc utilise un ensemble fini de variables, dont un sous-ensemble fini $\{x_{i_1}, \dots, x_{i_n}\} \subseteq \operatorname{dom}(\lambda)$. Il existe $\varphi_1, \dots, \varphi_n$ dans $\mathscr{C}$ telle que $x_{i_1} \in \operatorname{dom}\varphi_1, \dots, x_{i_n} \in \operatorname{dom}\varphi_n$. Comme $\mathscr{C}$ est une chaîne, donc soit $\varphi_0 = \max_{i\in[\![1,n]\!]} \varphi_i$ et on a $\varphi_0 \in \mathscr{C}$. On a, de plus, $x_{i_1}, \dots, x_{i_n} \in \operatorname{dom}\varphi_0$. Soit $\varphi_0 \in \mathscr{V}$ prolongeable en ψ_0 qui satisfait B. On définit :

$$\begin{aligned} \mu : \mathscr{P} &\longrightarrow \{0,1\} \\ x \in \operatorname{dom}\lambda &\longmapsto \lambda(x) \\ x \in \operatorname{var} B &\longmapsto \psi_0(x) \\ \text{sinon} &\longmapsto 0. \end{aligned}$$

On vérifie que la définition est cohérente sur l'intersection car λ et ψ_0 prolongent tous les deux φ_0 et donc $\lambda \in \mathscr{V}$ d'où $\mathscr{V}$ est inductif.

Soit γ un élément maximal de $\mathscr{V}$. Pour montrer le théorème, il suffit de montrer que $\operatorname{dom}\gamma = \mathscr{P}$. Si $\operatorname{dom}\gamma \neq \mathscr{P}$, soit $x \notin \operatorname{dom}\gamma$. Montrons que γ n'est pas maximal en définissant $\gamma' \in \mathscr{V}$ qui vérifie $\gamma \prec \gamma'$. On prend $\operatorname{dom}\gamma' = \operatorname{dom}\gamma \cup \{x\}$.

- ▷ Si, pour toute partie finie B de $\mathscr{E}$, il existe une valuation δ qui prolonge γ et qui satisfait B telle que $\delta(x) = 0$, alors on pose $\gamma'(x) = 0$.
- ▷ Sinon, on pose $\gamma'(x) = 1$.

Montrons que $\gamma' \in \mathscr{V}$. Soit B un ensemble fini de formules de $\mathscr{E}$.

- ▷ Si $\gamma'(x) = 0$ alors il existe une valuation δ prolongeant γ et satisfaisant B telle que $\delta(x) = 0$, et donc δ prolonge γ'.
- ▷ Si $\gamma'(x) = 1$ alors il existe une partie finie B_0 de $\mathscr{C}$ telle que toute valuation δ prolongeant γ et satisfaisant B_0 vérifie que $\delta(x) = 1$. On choisit une valuation qui prolonge γ et satisfait $B \cup B_0$; elle prolonge γ'.

On a donc une contradiction avec la maximalité de γ. On en conclut que $\operatorname{dom}\gamma = \mathscr{P}$, ce qui termine la preuve du théorème. □

12. La logique du premier ordre.

12.1 Les termes.

On commence par définir les *termes*, qui correspondent à des objets mathématiques. Tandis que les formules relient des termes et correspondent plus à des énoncés mathématiques.

Définition 12.1. Le langage $\mathscr{L}$ (du premier ordre) est la donnée d'une famille (pas nécessairement finie) de symboles de trois sortes :

- ▷ les symboles de *constantes*, notées c ;
- ▷ les symboles de *fonctions*, avec un entier associé, leur *arité*, notées $f(x_1, \ldots, x_n)$ où n est l'arité ;
- ▷ les symboles de *relations*, avec leur arité, notées R, appelés *prédicats*.

Les trois ensembles sont disjoints.

Remarque 12.1.

- ▷ Les constantes peuvent être vues comme des fonctions d'arité 0.
- ▷ On aura toujours dans les relations : « $=$ » d'arité 2, et « $\bot$ » d'arité 0.
- ▷ On a toujours un ensemble de variables $\mathscr{V}$.

Exemple 12.1. Le langage $\mathscr{L}_{\mathrm{g}}$ de la théorie des groupes est défini par :

- une constante : c,
- deux fonctions : f_1 d'arité 2 et f_2 d'arité 1 ;
- la relation $=$.

Ces symboles sont notés usuellement e, $*$, $\square^{-1}$ ou bien 0, $+$, $-$.

Exemple 12.2. Le langage $\mathscr{L}_{\mathrm{co}}$ des corps ordonnés est défini par :

- deux constantes 0 et 1,
- quatre fonctions $+$, $\times$, $-$ et $\square^{-1}$,
- deux relations $=$ et $\leq$.

Exemple 12.3. Le langage $\mathscr{L}_{\mathrm{ens}}$ de la théorie des ensembles est défini par :

- une constante $\emptyset$,
- trois fonctions $\cap$, $\cup$ et $\square^{\mathsf{c}}$,
- trois relations $=$, $\in$ et $\subseteq$.

Définition 12.2. **Par le haut.** L'ensemble $\mathscr{T}$ des termes sur le langage $\mathscr{L}$ est le plus petit ensemble de mots sur $\mathscr{L} \cup \mathscr{V} \cup \{(,), ,\}$ tel

- qu'il contienne $\mathscr{V}$ et les constantes ;
- qui est stable par application des fonctions, c'est-à-dire que pour des termes $t_1, \ldots, t_n$ et un symbole de fonction f d'arité n, alors $f(t_1, \ldots, t_n)$ est un terme. [1]

Par le bas. On pose

$$\mathscr{T}_0 = \mathscr{V} \cup \{c \mid c \text{ est un symbole de constante de } \mathscr{L}\},$$

puis

$$\mathcal{T}_{k+1} = \mathcal{T}_k \cup \left\{ f(t_1, \dots t_n) \;\middle|\; \begin{array}{c} f \text{ fonction d'arité } n \\ t_1, \dots, t_n \in \mathcal{T}_k \end{array} \right\},$$

et enfin

$$\mathcal{T} = \bigcup_{n \in \mathbb{N}} \mathcal{T}_n.$$

Remarque 12.2. Dans la définition des termes, un n'utilise les relations.

Exemple 12.4.
- ▷ Dans $\mathscr{L}_{\text{g}}$, $*(*(x, \square^{-1}(y)), e)$ est un terme, qu'on écrira plus simplement en $(x * y^{-1}) * e$.
- ▷ Dans $\mathscr{L}_{\text{co}}$, $(x + x) + (-0)^{-1}$ est un terme.
- ▷ Dans $\mathscr{L}_{\text{ens}}$, $(\emptyset^{\mathsf{c}} \cup \emptyset) \cap (x \cup y)^{\mathsf{c}}$ est un terme.

Définition 12.3. Si t et u sont des termes et x est une variable, alors $t[x : u]$ est le mot dans lequel les lettres de x ont été remplacées par le mot u. Le mot $t[x : u]$ est un terme (preuve en exercice).

Exemple 12.5. Avec $t = (x * y^{-1}) * e$ et $u = x * e$, alors on a

$$t[x : u] = ((x * e) * y^{-1}) * e.$$

Définition 12.4.
- ▷ Un terme *clos* est un terme sans variable (par exemple $(0 + 0)^{-1}$).
- ▷ La *hauteur* d'un terme est le plis petit k tel que $t \in \mathcal{T}_k$.

1. Attention : le « … » n'est pas un terme mais juste une manière d'écrire qu'on place les termes à côté des autres.

Exercice 12.1. ▷ Énoncer et prouver le lemme de lecture unique pour les termes.
▷ Énoncer et prouver un lemme de bijection entre les termes et un ensemble d'arbres étiquetés.

12.2 Les formules.

Définition 12.5. ▷ Les formules sont des mots sur l'alphabet

$$\mathscr{L} \cup \mathscr{V} \cup \{(,), ,, \exists, \forall, \wedge, \vee, \neg, \to\}.$$

▷ Une *formule atomique* est une formule de la forme $R(t_1, \dots, t_n)$ où R est un symbole de relation d'arité n et $t_1, \dots, t_n$ des termes.
▷ L'ensemble des *formules* $\mathscr{F}$ du langage $\mathscr{L}$ est défini par
– on pose $\mathscr{F}_0$ l'ensemble des formules atomiques ;
– on pose $\mathscr{F}_{k+1} = \mathscr{F}_k \cup \left\{ \begin{array}{c|c} (\neg F) & \\ (F \to G) & \\ (F \vee G) & F, G \in \mathscr{F}_k \\ (F \wedge G) & x \in \mathscr{V} \\ \exists x\, F & \\ \exists x\, G & \end{array} \right\}$;
– et on pose enfin $\mathscr{F} = \bigcup_{n \in \mathbb{N}} \mathscr{F}_n$.

Exercice 12.2. La définition ci-dessus est « par le bas ». Donner une définition par le haut de l'ensemble $\mathscr{F}$.

Exemple 12.6. ▷ Dans $\mathscr{L}_g$, un des axiomes de la théorie des groupes s'écrit

$$\forall x\, \exists x\, (x * y = e \wedge y * x = e).$$

▷ Dans $\mathscr{L}_{co}$, l'énoncé « le corps est de caractéristique 3 »

s'écrit

$$\forall x\,(x + (x + x) = 0).$$

▷ Dans $\mathscr{L}_{\text{ens}}$, la loi de De Morgan s'écrit

$$\forall x\,\forall y\,(x^{\mathsf{c}} \cup y^{\mathsf{c}} = (x \cap y)^{\mathsf{c}}).$$

Exercice 12.3.
▷ Donner et montrer le lemme de lecture unique.
▷ Énoncer et donner un lemme d'écriture en arbre.

Remarque 12.3 (Conventions d'écriture.)**.** On note :

- ▷ $x \leq y$ au lieu de $\leq(x, y)$;
- ▷ $\exists x \geq 0\,(F)$ au lieu de $\exists x\,(x \geq 0 \wedge F)$;
- ▷ $\forall x \geq 0\,(F)$ au lieu de $\forall x\,(x \geq 0 \to F)$;
- ▷ $A \leftrightarrow B$ au lieu de $(A \to B) \wedge (B \to A)$;
- ▷ $t \neq u$ au lieu de $\neg(t = u)$.

On enlèves les parenthèses avec les conventions de priorité

0. les symboles de relations (le plus prioritaire) ;
1. les symboles $\neg, \exists, \forall$;
2. les symboles $\wedge$ et $\vee$;
3. le symbole $\to$ (le moins prioritaire).

Exemple 12.7. Ainsi, $\forall x\ A \wedge B \to \neg C \vee D$ s'écrit

$$(((\forall x\ A) \wedge B) \to ((\neg C) \vee D)).$$

Remarque 12.4. Le calcul propositionnel est un cas particulier de la logique du premier ordre où l'on ne manipule que des relations d'arité 0 (pas besoin des fonctions et des variables) : les « variables » du calcul propositionnel sont des formules atomiques ; et on n'a pas de relation « = ».

Remarque 12.5. On ne peut pas exprimer *a priori* :

- ▷ des quantifications sur en ensemble[2] ;
- ▷ « $\exists n\, \exists x_1 \dots \exists x_n$ » une formule qui dépend d'un paramètre ;
- ▷ le principe de récurrence : si on a $\mathscr{P}(0)$ pour une propriété $\mathscr{P}$ et que si $\mathscr{P}(n) \to \mathscr{P}(n+1)$ alors on a $\mathscr{P}(n)$ pour tout n.

Quelques définitions techniques qui permettent de manipuler les formules.

Définition 12.6. L'ensemble des sous-formules de F, noté $\mathrm{S}(F)$ est défini par induction :

- ▷ si F est atomique, alors on définit $\mathrm{S}(F) = \{F\}$;
- ▷ si $F = F_1 \oplus F_2$ (avec $\oplus$ qui est $\vee$, $\to$ ou $\wedge$) alors on définit $\mathrm{S}(F) = \mathrm{S}(F_1) \cup \mathrm{S}(F_2) \cup \{F\}$;
- ▷ si $F = \neg F_1$, ou $F = \mathsf{Q} x\, F_1$ avec $\mathsf{Q} \in \{\forall, \exists\}$, alors on définit $\mathrm{S}(F) = \mathrm{S}(F_1) \cup \{F\}$.

C'est l'ensemble des formules que l'on voit comme des sous-arbres de l'arbre équivalent à la formule F.

Définition 12.7.

- ▷ La *taille* d'une formule, est le nombre de connecteurs ($\neg$, $\vee$, $\wedge$, $\to$), et de quantificateurs ($\forall$, $\exists$).
- ▷ La racine de l'arbre est
  - rien su la formule est atomique ;
  - « $\oplus$ » si $F = F_1 \oplus F_2$ avec $\oplus$ un connecteur (binaire ou unaire) ;
  - « Q » si $F = \mathsf{Q} x\, F_1$ avec Q un quantificateur.

Définition 12.8. ▷ Une *occurrence* d'une variable est un endroit où la variable apparait dans la formule (*i.e.* une feuille étiquetée par cette variable).

2. En dehors de $\mathscr{L}_{\mathrm{ens}}$, en tout cas.

- ▷ Une occurrence d'une variable est *liée* si elle se trouve dans une sous-formule dont l'opérateur principal est un quantificateur appelé à cette variable (*i.e.* un $\forall x\, F'$ ou un $\exists x\, F'$).
- ▷ Une occurrence d'une variable est *libre* quand elle n'est pas liée.
- ▷ Une variable est libre si elle a au moins une occurrence libre, sinon elle est liée.

Remarque 12.6. On note $F(x_1, \dots, x_n)$ pour dire que les variables libres sont F sont parmi $\{x_1, \dots, x_n\}$.

Définition 12.9. Une formule est *close* si elle n'a pas de variables libres.

Définition 12.10 (Substitution)**.** On note $F[x := t]$ la formule obtenue en remplaçant toutes les occurrences libres de x par t, après renommage éventuel des occurrences des variables liées de F qui apparaissent dans t.

Définition 12.11 (Renommage)**.** On donne une définition informelle et incomplète ici. On dit que les formules F et G sont α-équivalentes si elle sont syntaxiquement identiques à un renommage près des occurrences liées des variables.

Exemple 12.8. On pose

$$F(x, z) := \forall y\, (x * y = y * z) \wedge \forall x\, (x * x = 1),$$

et alors

- ▷ $F(z, z) = F[x := z] = \forall y\, (z * y = y * z) \wedge \forall x\, (x * x = 1)$;
- ▷ $F(y^{-1}, x) = F[x := y^{-1}] = \forall u (y^{-1} * u = u * z) \wedge \forall x (x * x = 1)$.

On a procédé à un renommage de y à u.

12.3 Les démonstrations en déduction naturelle.

Définition 12.12. Un *séquent* est un coupe noté $\Gamma \vdash F$ (où $\vdash$ se lit « montre » ou « thèse ») tel que Γ est un ensemble fini de formules appelé *contexte* (*i.e.* l'ensemble des hypothèses), la formule F est la *conséquence* du séquent.

Remarque 12.7. Les formules ne sont pas nécessairement closes. Et on note souvent Γ comme une liste.

Définition 12.13. On dit que $\Gamma \vdash F$ est *prouvable*, *démontrable* ou *dérivable*, s'il peut être obtenu par une suite finie de règles (*c.f.* ci-après). On dit qu'une formule F est *prouvable* si $\emptyset \vdash F$ l'est.

Définition 12.14 (Règles de la démonstration)**.** Une règle s'écrit

$$\frac{\textit{prémisses} : \text{des séquents}}{\textit{conclusion} : \text{un séquent}}\ \text{nom de la règle}\ .$$

Axiome.

$$\frac{}{\Gamma, A \vdash A}\ \mathsf{ax}$$

Affaiblissement.

$$\frac{\Gamma \vdash A}{\Gamma, B \vdash A}\ \mathsf{aff}$$

Implication.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \to_{\mathsf{i}} \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \to_{\mathsf{e}}$$ [3]

Conjonction.

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge_{\mathsf{i}} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \vee_{\mathsf{e}}^{\mathsf{g}} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \vee_{\mathsf{e}}^{\mathsf{d}}$$

Disjonction.

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee_{\mathsf{i}}^{\mathsf{g}} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee_{\mathsf{i}}^{\mathsf{d}}$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee_{\mathsf{e}}{}^{4}.$$

Négation.

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \neg_{\mathsf{i}} \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash \bot}\ \neg_{\mathsf{e}}$$

Absurdité classique.

$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\ \bot_{\mathsf{e}}$$

(En logique intuitionniste, on retire l'hypothèse $\neg A$ dans la prémisse.)

Quantificateur universel.

si x n'est pas libre dans les formules de Γ

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x\, A}\ \forall_{\mathsf{i}}$$

quitte à renommer les variables liées de A qui apparaissent dans t

$$\frac{\Gamma \vdash \forall x\, A}{\Gamma \vdash A[x := t]}\ \forall_{\mathsf{e}}$$

Quantificateur existentiel.

$$\frac{\Gamma \vdash A[x := t]}{\Gamma \vdash \exists x\, A}\ \exists_{\mathsf{i}}$$

$$\text{avec } x \text{ ni libre dans } C \text{ ou dans les formules de } \Gamma \qquad \frac{\Gamma \vdash \exists x\, A \qquad \Gamma, A \vdash C}{\Gamma \vdash C}\ \exists_{\mathsf{e}}$$

12.4 La sémantique.

Définition 12.15. Soit $\mathscr{L}$ un langage de la sémantique du premier ordre. On appelle *interprétation* (ou *modèle*, ou *structure*) du langage $\mathscr{L}$ l'ensemble $\mathscr{M}$ des données suivantes :

- ▷ un ensemble non vide, noté $|\mathscr{M}|$, appelé *domaine* ou *ensemble de base* de $\mathscr{M}$;
- ▷ pour chaque symbole c de constante, un élément $c_{\mathscr{M}}$ de $|\mathscr{M}|$;
- ▷ pour chaque symbole f de fonction n-aire, une fonction $f_{\mathscr{M}} : |\mathscr{M}|^n \to |\mathscr{M}|$;
- ▷ pour chaque symbole R de relation n-aire (sauf pour l'égalité « $=$ »), un sous-ensemble $R_{\mathscr{M}}$ de $|\mathscr{M}|^n$.

Remarque 12.8.
- ▷ La relation « $=$ » est toujours interprétée par la vraie égalité :
$$\{(a, a) \mid a \in |\mathscr{M}|\}.$$
- ▷ On note, par abus de notation, $\mathscr{M}$ pour $|\mathscr{M}|$.
- ▷ Par convention, $|\mathscr{M}|^0 = \{\emptyset\}$.

Exemple 12.9. Avec $\mathscr{L}_{\text{corps}} = \{0, 1, +, \times, -, \square^{-1}\}$, on peut choisir
- ▷ $|\mathscr{M}| = \mathbb{R}$ avec $0_{\mathbb{R}}$, $1_{\mathbb{R}}$, $+_{\mathbb{R}}$, $\times_{\mathbb{R}}$, $-_{\mathbb{R}}$ et $\square^{-1}_{\mathbb{R}}$;
- ▷ ou $|\mathscr{M}| = \mathbb{R}$ avec $2_{\mathbb{R}}$, $2_{\mathbb{R}}$, $-_{\mathbb{R}}$, $+_{\mathbb{R}}$, *etc.*

Définissions la *vérité*.

3. Aussi appelée *modus ponens*
4. C'est un raisonnement par cas

Définition 12.16. Soit $\mathscr{M}$ une interprétation de $\mathscr{L}$.

- ▷ Un *environnement* est une fonction de l'ensemble des variables dans $|\mathscr{M}|$.
- ▷ Si e est un environnement et $a \in |\mathscr{M}|$, on note $e[x := a]$ l'environnement e' tel que $e'(x) = a$ et pour $y \neq x$, $e(y) = e'(y)$.
- ▷ La *valeur* d'un terme t dans l'environnement e, noté $\mathscr{V}al_{\mathscr{M}}(t, e)$, est définie par induction sur l'ensemble des termes de la façon suivante :
  - $\mathscr{V}al_{\mathscr{M}}(c, e) = c_{\mathscr{M}}$ si c est une constante ;
  - $\mathscr{V}al_{\mathscr{M}}(c, e) = e(x)$ si x est une variable ;
  - $\mathscr{V}al_{\mathscr{M}}(f(t_1, \dots, t_n), e) = f_{\mathscr{M}}(\mathscr{V}al_{\mathscr{M}}(t_1, e), \dots, \mathscr{V}al_{\mathscr{M}}(t_n, e))$.

Remarque 12.9. La valeur est $\mathscr{V}al_{\mathscr{M}}(t, e)$ est un élément de $|\mathscr{M}|$.

Exemple 12.10. Dans $\mathscr{L}_{\text{arith}} = \{0, 1, +, \times\}$, avec le modèle

$$\mathscr{M} : \mathbb{N}, 0_{\mathbb{N}}, 1_{\mathbb{N}}, +_{\mathbb{N}}, \times_{\mathbb{N}},$$

et l'environnement

$$e : \quad x_1 \mapsto 2_{\mathbb{N}} \quad x_2 \mapsto 0_{\mathbb{N}} \quad x_3 \mapsto 3_{\mathbb{N}},$$

alors la valeur du terme $t := (1 \times x_1) + (x_2 \times x_3) + x_2$ est $2_{\mathbb{N}} = (1 \times 2) + (0 \times 3) + 0$.

Lemme 12.1. La valeur $\mathscr{V}al_{\mathscr{M}}(t, e)$ ne dépend que de la valeur de e sur les variables de t. □

Notation.
- ▷ Lorsque cela est possible, on oublie $\mathscr{M}$ et e dans la notation, et on note $\mathscr{V}al(t)$.
- ▷ À la place de $\mathscr{V}al_{\mathscr{M}}(t, e)$ quand $x_1, \dots, x_n$ sont les variables de t et $e(x_1) = a_1, \dots, e(x_n) = a_n$, on note $t[a_1, \dots, a_n]$ ou aussi $t[x_1 := a_1, \dots, x_n := a_n]$. C'est un *terme à paramètre*,

mais attention ce n'est ***ni un terme, ni une substitution***.

Définition 12.17. Soit $\mathscr{M}$ une interprétation d'un langage $\mathscr{L}$. La *valeur* d'une formule F de $\mathscr{L}$ dans l'environnement e est un élément de $\{0,1\}$ noté $\mathscr{V}al_{\mathscr{M}}(F,e)$ et définie par induction sur l'ensemble des formules par

- $\mathscr{V}al_{\mathscr{M}}(R(t_1,\dots,t_n),e) = 1$ ssi $(\mathscr{V}al_{\mathscr{M}}(t_1,e),\dots,\mathscr{V}al_{\mathscr{M}}(t_n,e)) \in R_{\mathscr{M}}$;
- $\mathscr{V}al_{\mathscr{M}}(\bot,e) = 0$;
- $\mathscr{V}al_{\mathscr{M}}(\neg F,e) = 1 - \mathscr{V}al_{\mathscr{M}}(F,e)$;
- $\mathscr{V}al_{\mathscr{M}}(F \wedge G,e) = 1$ ssi $\mathscr{V}al_{\mathscr{M}}(F,e) = 1$ et $\mathscr{V}al_{\mathscr{M}}(G,e) = 1$;
- $\mathscr{V}al_{\mathscr{M}}(F \vee G,e) = 1$ ssi $\mathscr{V}al_{\mathscr{M}}(F,e) = 1$ ou $\mathscr{V}al_{\mathscr{M}}(G,e) = 1$;
- $\mathscr{V}al_{\mathscr{M}}(F \to G,e) = 1$ ssi $\mathscr{V}al_{\mathscr{M}}(F,e) = 0$ ou $\mathscr{V}al_{\mathscr{M}}(G,e) = 1$;
- $\mathscr{V}al_{\mathscr{M}}(\forall x\, F,e) = 1$ ssi pour tout $a \in |\mathscr{M}|$, $\mathscr{V}al_{\mathscr{M}}(F,e[x := a]) = 1$;
- $\mathscr{V}al_{\mathscr{M}}(\exists x\, F,e) = 1$ ssi il existe $a \in |\mathscr{M}|$, $\mathscr{V}al_{\mathscr{M}}(F,e[x := a]) = 1$.

Remarque 12.10.

- On se débrouille pour que les connecteurs aient leur sens courant, les « mathématiques naïves ».
- Dans le cas du calcul propositionnel, si R est d'arité 0, *i.e.* une variable propositionnelle, comme $|\mathscr{M}|^0 = \{\emptyset\}$ alors on a deux possibilité :
  - ou bien $R = \emptyset$, et alors on convient que $\mathscr{V}al_{\mathscr{M}}(R,e) = 0$;
  - ou bien $R = \{\emptyset\}$, et alors on convient que $\mathscr{V}al_{\mathscr{M}}(R,e) = 1$.

Remarque 12.11. On verra plus tard qu'on peut construire les entiers avec

- $0 : \emptyset$,
- $1 : \{\emptyset\}$,
- $2 : \{\emptyset, \{\emptyset\}\}$,
- $\vdots \quad \vdots$
- $n+1 : n \cup \{n\}$,

▷ ⋮ ⋮

Notation. À la place de $\mathcal{Val}_{\mathcal{M}}(F, e) = 1$, on notera $\mathcal{M}, e \models F$ ou bien $\mathcal{M} \models F$. On dit que $\mathcal{M}$ *satisfait* F, que $\mathcal{M}$ est un *modèle* de F (dans l'environnement e), que F est est vraie dans $\mathcal{M}$.

Lemme 12.2. La valeur $\mathcal{Val}_{\mathcal{M}}(F, e)$ ne dépend que de la valeur de e sur les variables libres de F.

Preuve. En exercice. □

Corollaire 12.1. Si F est close, alors $\mathcal{Val}_{\mathcal{M}}(F, e)$ ne dépend pas de e et on note $\mathcal{M} \models F$ ou $\mathcal{M} \not\models F$.

Remarque 12.12. Dans le cas des formules closes, on doit passer un environnement à cause de $\forall$ et $\exists$.

Notation. On note $F[a_1, \ldots, a_n]$ pour $\mathcal{Val}_{\mathcal{M}}(F, e)$ avec $e(x_1) = a_1, \ldots, e(x_n) = a_n$. C'est une *formule à paramètres*, mais ce n'est ***pas une formule***.

Exemple 12.11. Dans $\mathcal{L} = \{S\}$ où S est une relation binaire, on considère deux modèles :

- $\mathcal{N} : |\mathcal{N}| = \mathbb{N}$ avec $S_{\mathcal{N}} = \{(x, y) \mid x < y\}$,
- $\mathcal{R} : |\mathcal{R}| = \mathbb{R}$ avec $S_{\mathcal{R}} = \{(x, y) \mid x < y\}$;

et deux formules

- $F = \forall x\, \forall y\, (S\, x\, y \to \exists z\, (S\, x\, z \wedge S\, z\, y))$,
- $G = \exists x\, \forall y\, (x = y \vee S\, x\, y)$;

alors on a

$$\mathcal{N} \not\models F \quad \mathcal{R} \models F \quad \mathcal{N} \models G \quad \mathcal{R} \not\models G.$$

En effet, la formule F représente le fait d'être un ordre dense, et G d'avoir un plus petit élément.

Définition 12.18. Dans un langage $\mathscr{L}$, une formule F est un *théorème* (*logique*) si pour toute structure $\mathscr{M}$ et tout environnement e, on a $\mathscr{M}, e \models F$.

Exemple 12.12. Quelques théorèmes simples : $\forall x \neg \bot$, et $\forall x\, x = x$ et même $x = x$ car on ne demande pas que la formule soit clause.

Dans $\mathscr{L}_{\mathrm{g}} = \{e, *, \square^{-1}\}$, on considère deux formules

- $F = \forall x\, \forall y\, \forall z\, ((x * (y * z) = (x * y) * z) \wedge x * e = e * x = x \wedge \exists t\, (x * t = e \wedge t * x = e))$;
- et $G = \forall e' = \forall e'\, (\forall x\, (x * e' = e' * x = x) \to e = e')$.

Aucun des deux n'est un théorème (il n'est vrai que dans les groupes pour F (c'est même la définition de groupe) et dans les monoïdes pour G (unicité du neutre)), mais $F \to G$ est un théorème logique.

Définition 12.19. Soient $\mathscr{L}$ et $\mathscr{L}'$ deux langages. On dit que $\mathscr{L}'$ *enrichit* $\mathscr{L}$ ou que $\mathscr{L}$ est une *restriction* de $\mathscr{L}'$ si $\mathscr{L} \subseteq \mathscr{L}'$.

Dans ce cas, si $\mathscr{M}$ est une interprétation de $\mathscr{L}$, et si $\mathscr{M}'$ est une interprétation de $\mathscr{L}'$ alors on dit que $\mathscr{M}'$ est un *enrichissement* de $\mathscr{M}$ ou que $\mathscr{M}$ est une *restriction* de $\mathscr{M}'$ ssi $|\mathscr{M}| = |\mathscr{M}'|$ et chaque symbole de $\mathscr{L}$ a la même interprétation dans $\mathscr{M}$ et $\mathscr{M}'$, *i.e.* du point de vue de $\mathscr{L}$, $\mathscr{M}$ et $\mathscr{M}'$ sont les mêmes.

Exemple 12.13. Avec $\mathscr{L} = \{e, *\}$ et $\mathscr{L}' = \{e, *, \square^{-1}\}$ alors $\mathscr{L}'$ est une extension de $\mathscr{L}$. On considère

- $\mathscr{M}$: $\quad |\mathscr{M}| = \mathbb{Z} \quad e_{\mathscr{M}} = 0_{\mathbb{Z}} \quad *_{\mathscr{M}} = +_{\mathbb{Z}}$;
- $\mathscr{M}'$: $\quad |\mathscr{M}'| = \mathbb{Z} \quad e_{\mathscr{M}'} = 0_{\mathbb{Z}} \quad *_{\mathscr{M}'} = +_{\mathbb{Z}} \quad \square^{-1}_{\mathscr{M}'} = \mathrm{id}_{\mathbb{Z}}$,

et alors $\mathscr{M}'$ est une extension de $\mathscr{M}$.

Proposition 12.1. Si $\mathscr{M}$ une interprétation de $\mathscr{L}$ est un enrichis-

sement de $\mathscr{M}'$, une interprétation de $\mathscr{L}'$, alors pour tout environnement e,

1. si t est un terme de $\mathscr{L}$, alors $\mathscr{V}al_{\mathscr{M}}(t,e) = \mathscr{V}al_{\mathscr{M}'}(t,e)$;
2. si F est une formule de $\mathscr{L}$ alors $\mathscr{V}al_{\mathscr{M}}(F,e) = \mathscr{V}al_{\mathscr{M}'}(F,e)$.

Preuve. En exercice. □

Corollaire 12.2. La vérité d'une formule dans une interprétation ne dépend que de la restriction de cette interprétation au langage de la formule. □

Définition 12.20. Deux formules F et G sont *équivalentes* si $F \leftrightarrow G$ est un théorème logique.

Proposition 12.2. Toute formule est équivalente à une formule n'utilisant que les connecteurs logiques $\neg$, $\vee$ et $\exists$. □

Définition 12.21. Soient $\mathscr{M}$ et $\mathscr{N}$ deux interprétations de $\mathscr{L}$.

1. Un *$\mathscr{L}$-morphisme* de $\mathscr{M}$ est une fonction $\varphi : |\mathscr{M}| \to |\mathscr{N}|$ telle que
   - ▷ pour chaque symbole de constante c, on a $\varphi(c_{\mathscr{M}}) = c_{\mathscr{N}}$;
   - ▷ pour chaque symbole f de fonction n-aire, on a
   $$\varphi(f_{\mathscr{M}}(a_1,\dots,a_n)) = f_{\mathscr{N}}(\varphi(a_1),\dots,\varphi(a_n)) ;$$
   - ▷ pour chaque symbole R de relation n-aire (autre que « = »), on a
   $$(a_1,\dots,a_n) \in R_{\mathscr{M}} \text{ ssi } (\varphi(a_1),\dots,\varphi(a_n)) \in R_{\mathscr{N}}.$$
   - ▷ Un *$\mathscr{L}$-isomorphisme* est un $\mathscr{L}$-morphisme bijectif.
   - ▷ Si $\mathscr{M}$ et $\mathscr{N}$ sont *isomorphes* s'il existe un $\mathscr{L}$-isomorphisme de $\mathscr{M}$ à $\mathscr{N}$.

Remarque 12.13.
1. On ne dit rien sur « = » car si on impose la même condition que pour les autres relations alors nécessairement φ est injectif.
2. La notion dépend du langage $\mathscr{L}$.
3. Lorsqu'on a deux structures isomorphes, on les confonds, ce sont les mêmes, c'est un renommage.

Exemple 12.14. Avec $\mathscr{L}_{\text{ann}} = \{0, +, \times, -\}$ et $\mathscr{L}' = \mathscr{L}_{\text{ann}} \cup \{1\}$, et les deux modèles $\mathscr{M} : \mathbb{Z}/3\mathbb{Z}$ et $\mathscr{N} = \mathbb{Z}/12\mathbb{Z}$, on considère la fonction définie (on néglige les cas inintéressants) par $\varphi(\bar{n}) = \overline{4n}$.

Est-ce que φ est un morphisme de $\mathscr{M}$ dans $\mathscr{N}$? Oui... et non... Dans $\mathscr{L}$ c'est le cas, mais pas dans $\mathscr{L}'$ car $\varphi(1) = 4$.

Exemple 12.15. Dans $\mathscr{L} = \{c, f, R\}$ avec f une fonction binaire, et R une relation binaire, on considère

- $\mathscr{M} : \mathbb{R}, 0, +, \leq$;
- $\mathscr{N} :]0, +\infty[, 1, \times, \leq$.

Existe-t-il un morphisme de $\mathscr{M}$ dans $\mathscr{N}$? Oui, il suffit de poser le morphisme $\varphi : x \mapsto \mathrm{e}^x$.

Proposition 12.3. La composée de deux morphismes (*resp.* isomorphisme) est un morphisme (*resp.* un isomorphisme). □

Notation. Si φ est un morphisme de $\mathscr{M}$ dans $\mathscr{N}$ et e un environnement de $\mathscr{M}$, alors on note $\varphi(e)$ pour $\varphi \circ e$. C'est un environnement de $\mathscr{N}$.

Lemme 12.3. Soient $\mathscr{M}$ et $\mathscr{N}$ deux interprétations de $\mathscr{L}$, et φ un morphisme de $\mathscr{M}$ dans $\mathscr{N}$. Alors pour tout terme t et environnement e, on a

$$\varphi(\mathscr{V}al_{\mathscr{M}}(t, e)) = \mathscr{V}al_{\mathscr{N}}(t, \varphi(e)).$$

□

Lemme 12.4. Soient $\mathscr{M}$ et $\mathscr{N}$ deux interprétations de $\mathscr{L}$, et φ un morphisme ***injectif*** de $\mathscr{M}$ dans $\mathscr{N}$. Alors pour toute formule atomique F et environnement e, on a

$$\mathscr{M}, e \models F \text{ ssi } \mathscr{N}, \varphi(e) \models F$$

Lemme 12.5. Soient $\mathscr{M}$ et $\mathscr{N}$ deux interprétations de $\mathscr{L}$, et φ un ***isomorphisme***[5] de $\mathscr{M}$ dans $\mathscr{N}$. Alors pour toute formule F et environnement e, on a

$$\mathscr{M}, e \models F \text{ ssi } \mathscr{N}, \varphi(e) \models F$$

Corollaire 12.3. Deux interprétations isomorphismes satisfont les mêmes formules closes.

Exercice 12.4. Les groupes $\mathbb{Z}/4\mathbb{Z}$ et $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ sont-ils isomorphes ? Non. En effet, les deux formules

- $\exists x\,(x \neq e \wedge x * x \neq e \wedge x * (x * x) \neq e \wedge x * (x * (x * x)) = e)$,
- $\forall x\,(x * x) = e$

ne sont pas vraies dans les deux (pour la première, elle est vraie dans $\mathbb{Z}/4\mathbb{Z}$ mais pas dans $(\mathbb{Z}/2\mathbb{Z})^2$ et pour la seconde, c'est l'inverse).

Remarque 12.14. La réciproque du corollaire est ***fausse*** : deux interprétations qui satisfont les mêmes formules closes ne sont pas nécessairement isomorphes. Par exemple, avec $\mathscr{L} = \{\leq\}$, les interprétations $\mathbb{R}$ et $\mathbb{Q}$ satisfont les mêmes formules closes, mais ne sont pas isomorphes.

Définition 12.22. Soit $\mathscr{L}$ un langage, $\mathscr{M}$ et $\mathscr{N}$ deux interpréta-

5. On utilise ici la *surjectivité* pour le « $\exists$ ».

tions de $\mathscr{L}$. On dit que $\mathscr{N}$ est une *extension* de $\mathscr{M}$ (ou $\mathscr{M}$ est une *sous-interprétation* de $\mathscr{N}$) si les conditions suivants sont satisfaites :

- $|\mathscr{M}| \subseteq |\mathscr{N}|$;
- pour tout symbole de constante c, on a $c_{\mathscr{M}} = c_{\mathscr{N}}$;
- pour tout symbole de fonction n-aire f, on a $f_{\mathscr{M}} = f_{\mathscr{N}}\big|_{|\mathscr{M}|^n}$ (donc en particulier $f_{\mathscr{N}}(|\mathscr{M}|^n) \subseteq |\mathscr{M}|$) ;
- pour tout symbole de relation n-aire R, on a $R_{\mathscr{M}} = R_{\mathscr{N}} \cap |\mathscr{M}|^n$.

Proposition 12.4. Soient $\mathscr{M}$ et $\mathscr{N}$ deux interprétations de $\mathscr{L}$. Alors $\mathscr{M}$ est isomorphe à une sous-interprétation $\mathscr{M}'$ de $\mathscr{N}$ si et seulement si, il existe un morphisme injectif de $\mathscr{M}$ dans $\mathscr{N}$.

Exemple 12.16 (Construction de $\mathbb{Z}$ à partir de $\mathbb{N}$)**.** On pose la relation $(p,q) \sim (p',q')$ si $p+q' = p'+q$. C'est une relation d'équivalence sur $\mathbb{N}^2$. On pose $\mathbb{Z} := \mathbb{N}^2/\!\sim$ (il y a un isomorphisme $\mathbb{N}^2/\!\sim \to \mathbb{Z}$ par $(p,q) \mapsto p-q$). Est-ce qu'on a $\mathbb{N} \subseteq \mathbb{N}^2/\!\sim$? D'un point de vue ensembliste, non. Mais, généralement, l'inclusion signifie avoir un morphisme injectif de $\mathbb{N}$ dans $\mathbb{N}^2/\!\sim$.

Définition 12.23. Une *théorie* est un ensemble (fini ou pas) de formules closes. Les éléments de la théorie sont appelés *axiomes*.

Exemple 12.17. La *théorie des groupes* est

$$T_{\text{groupe}} := \Big\{ \forall x\,(x * e = e * x = x),\\ \forall x\,(x * x^{-1} = e \wedge x^{-1} * x = e),\\ \forall x\,\forall y\,\forall z\,(x * (y * z) = (x * y) * z) \Big\}$$

dans le langage $\mathscr{L}_{\text{g}}$.

Exemple 12.18. La *théorie des ensembles infinis* est

$$T_{\text{ens infinis}} := \Big\{ \exists x\,(x = x),\quad \exists x\,\exists y\,(x \neq y),\quad \exists x\,\exists y\,\exists z\,(x \neq y \wedge y \neq z \wedge z \neq x)\quad \dots \Big\}$$

dans le langage $\mathscr{L}_{\text{ens}}$.

Définition 12.24 (Sémantique).
- ▷ Une interprétation $\mathscr{M}$ *satisfait* T (ou $\mathscr{M}$ est un *modèle* de T), noté $\mathscr{M} \models T$, si $\mathscr{M}$ satisfait toutes les formules de T.
- ▷ Une théorie T est *contradictoire* s'il n'existe pas de modèle de T. Sinon, on dit qu'elle est *non-contradictoire*, ou *satisfiable*, ou *satisfaisable*.

Exemple 12.19. Les deux théories précédentes, T_{groupes} et $T_{\text{ens infinis}}$, sont non-contradictoires.

Définition 12.25 (Syntaxique). Soit T une théorie.
- ▷ Soit A une formule. On note $T \vdash A$ s'il existe un sous-ensemble fini T' tel que $T' \subseteq T$ et $T' \vdash A$.
- ▷ On dit que T est *consistante* si $T \nvdash \bot$, sinon T est *inconsistante*.
- ▷ On dit que T est *complète* (« *axiome-complète* ») si T est consistante et, pour toute formule $F \in \mathscr{F}$, on a $T \vdash F$ ou on a $T \vdash \neg F$.

Exemple 12.20. La théorie des groupes n'est pas complète : par exemple,

$$F := \forall x\,\forall y\,(x * y = y * x)$$

est parfois vraie, parfois fausse, cela dépend du groupe considéré.

Exemple 12.21. La théorie

$$T = \mathbf{Th}(\mathbb{N}) := \{\text{les formules } F \text{ vraies dans } \mathbb{N}\}$$

est complète mais pas pratique.

De par le théorème d'*incomplétude de Gödel* (c'est un sens différent du « complet » défini avant), on montre qu'on ne peut pas avoir de *joli* ensemble d'axiomes pour $\mathbb{N}$.

Proposition 12.5. Soit T une théorie complète.

1. Soit A une formule close. On a $T \vdash \neg A$ ssi $T \nvdash A$.
2. Soient A et B des formules closes. On a $T \vdash A \vee B$ ssi $T \vdash A$ ou $T \vdash B$.

Preuve.

▷ Si $T \vdash \neg A$ et $T \vdash A$, alors il existe $T', T'' \subseteq_{\text{fini}} T$ tels que $T' \vdash \neg A$ et $T'' \vdash A$. On a donc $T' \cup T'' \vdash \bot$ par :

$$\dfrac{\dfrac{T' \vdash \neg A}{T' \cup T'' \vdash \neg A}\,\mathsf{aff} \qquad \dfrac{T'' \vdash A}{T' \cup T'' \vdash A}\,\mathsf{aff}}{T' \cup T'' \vdash \bot}\,\neg_{\mathsf{e}}$$

On en conclut que $T \vdash \bot$, absurde car T supposée complète donc consistante. On a donc $T \vdash \neg A$ implique $T \nvdash A$.

Réciproquement, si $T \nvdash A$ et $T \nvdash \neg A$, alors c'est impossible car T est complète. On a donc $T \nvdash A$ implique $T \vdash \neg A$.

▷ Si $T \vdash A$ ou $T \vdash B$, alors par la règle $\vee^{\mathsf{g}}_{\mathsf{i}}$ ou $\vee^{\mathsf{d}}_{\mathsf{i}}$, on montre que $T \vdash A \vee B$.

Réciproquement, si $T \vdash A \vee B$ et $T \nvdash A$ et $T \nvdash B$ alors, par 1, on a $T \vdash \neg A$ et $T \vdash \neg B$. On montre ainsi (en exercice) que $T \vdash \neg(A \vee B)$ d'où $T \vdash \bot$ par $\neg_{\mathsf{e}}$. C'est impossible car T est complète donc consistante, d'où $T \vdash A \vee B$ im-

plique $T \vdash A$ ou $T \vdash B$.

□

12.5 Théorème de complétude de Gödel.

Théorème 12.1 (Complétude de Gödel (à double sens))**.**

Version 1. Soit T une théorie et F une formule close. On a $T \vdash F$ ssi $T \models F$.

Version 2. Une théorie T est consistante (syntaxe) ssi elle est non-contradictoire (sémantique).

Remarque 12.15. La version 1 se décompose en deux théorèmes :

- le *théorème de correction* (ce que l'on prouve est vrai)

$$T \vdash F \implies T \models F\,;$$

- le *théorème de complétude* (ce qui est vrai est prouvable)

$$T \models F \implies T \vdash F.$$

Pour la version 2, on peut aussi la décomposer en deux théorèmes[6] :

- la *correction*, T non-contradictoire implique T consistante ;
- la *complétude*, T consistante implique T non-contradictoire.

Par contraposée, on a aussi qu'une théorie contradictoire est inconsistante.

Proposition 12.6. Les deux versions du théorème de correction sont équivalentes.

6. On a une négation dans ce théorème, donc ce n'est pas syntaxe implique sémantique pour la correction, mais non sémantique implique non syntaxe.

Preuve. ▷ D'une part, on montre (par contraposée) « non V2 implique non V1 ». Soit T non-contradictoire et inconsistante. Il existe un modèle $\mathscr{M}$ tel que $\mathscr{M} \models T$ et $T \vdash \bot$. Or, par définition, $\mathscr{M} \not\models \bot$ donc $T \not\models \bot$.

▷ D'autre part, on montre « V2 implique V1 ». Soit T et F tels que $T \vdash F$. Ainsi, $T \cup \neg F \vdash \bot$, d'où $T \cup \{\neg F\}$ est inconsistante, et d'où, par la version 2 de la correction, on a que $T \cup \{\neg F\}$ contradictoire, donc on n'a pas de modèle. On a alors que, touts les modèles de T sont des modèles de F, autrement dit $T \models F$.

□

Proposition 12.7. Les deux versions du théorème de complétude (sens unique) sont équivalentes.

Preuve. ▷ Soit T contradictoire. Elle n'a pas de modèle. Ainsi, on a $T \models \bot$ d'où $T \vdash \bot$ par la version 1, elle est donc inconsistante.

▷ Soit $T \models F$. Considérons $T \cup \{\neg F\}$: cette théorie n'a pas de modèle, donc est contradictoire, donc est inconsistante, et on a donc que $T \cup \{\neg F\} \vdash \bot$ d'où $T \vdash F$ par $\bot_e$.

□

Remarque 12.16 (Attention !). On utilise « $\models$ » dans deux sens.

▷ Dans le sens *modèle* $\models$ *formule*, on dit qu'une formule est vraie dans un modèle, c'est le sens des mathématiques classiques.

▷ Dans le sens *théorie* $\models$ *formule*, on dit qu'une formule est vraie dans tous les modèles de la théorie, c'est un sens des mathématiques plus inhabituel.

12.5.1 Preuve du théorème de correction.

Exercice 12.5. Montrer que le lemme ci-dessous implique la version 1 de la correction.

Lemme 12.6. Soient T une théorie, $\mathscr{M}$ un modèle et F une formule close. Si $\mathscr{M} \models T$ et $T \vdash F$ alors $\mathscr{M} \models F$.

Preuve. Comme d'habitude, pour montrer quelque chose sur les formules closes, on commence par les formules et même les termes. On commence par montrer que la substitution dans les termes a un sens sémantique.

Lemme 12.7. Soient t et u des termes et e un environnement. Soient $v := t[x := u]$ et $e' := e[x := \mathscr{V}al(u, e)]$. Alors, $\mathscr{V}al(v, e) = \mathscr{V}al(t, e')$.

Preuve. En exercice. □

Lemme 12.8. Soit A une formule, t un terme, et e un environnement. Si $e' := e[x := \mathscr{V}al(t, e)]$ alors $\mathscr{M}, e \models A[x := t]$ ssi $\mathscr{M}, e' \models A$.

Preuve. En exercice. □

On termine la preuve en montrant la proposition ci-dessous. □

Montrons cette proposition plus forte que le lemme.

Proposition 12.8. Soient Γ un ensemble de formules et A une formule. Soit $\mathscr{M}$ une interprétation et soit e un environnement. Si $\mathscr{M}, e \models \Gamma$, et $\Gamma \models A$ alors $\mathscr{M}, e \models A$.

Preuve. Par induction sur la preuve de $\Gamma \vdash A$, on montre la proposition précédente.

- ▷ Cas inductif $\to_{\mathsf{i}}$. On sait que A est de la forme $B \to C$, et

on montre que de $\Gamma, B \vdash C$ on montre $\Gamma \vdash B \to C$. Soient $\mathscr{M}$ et e tels que $\mathscr{M}, e \models \Gamma$. Montrons que $\mathscr{M}, e \models B \to C$. Il faut donc montrer que si $\mathscr{M}, e \models B$ alors $\mathscr{M}, e \models C$. Si $\mathscr{M}, e \models B$ alors $\mathscr{M}, e \models \Gamma \cup \{B\}$. Or, comme $\Gamma, B \vdash C$ alors par hypothèse d'induction, on a que $\mathscr{M}, e \models C$.

- ▷ Cas inductif $\forall_e$. Si A est de la forme $B[x := t]$, alors de $\Gamma \vdash \forall x\, B$, on en déduit que $\Gamma \vdash B[x := t]$. Soit $\mathscr{M}, e \models \Gamma$ et $a := \mathscr{V}al(t, e)$. Par hypothèse de récurrence, on a que $\mathscr{M}, e \models \forall x\, B$ donc $\mathscr{M}, e[x := a] \models B$ et d'après le lemme précédent, on a que $\mathscr{M}, e \models B[x := t]$.
- ▷ Les autre cas inductifs sont laissé en exercices.
- ▷ Cas de base ax. Si $A \in \Gamma$ et $\mathscr{M}, e \models \Gamma$ alors $\mathscr{M}, e \models A$.
- ▷ Cas de base $=_i$. On a, pour tout $\mathscr{M}, e$ que $\mathscr{M}, e \models t = t$.

□

Cette proposition permet de conclure la preuve du lemme précédent.

12.5.2 Preuve du théorème de complétude.

On va montrer la version 2, en ***trois étapes***. Soit T une théorie consistante sur le langage $\mathscr{L}$.

1. On enrichit le langage $\mathscr{L}$ en $\mathscr{L}'$ avec des constantes, appelées *témoins de Henkin*, et qui nous donnerons les éléments de notre ensemble de base : les termes.
2. Pour définir complètement le modèle, on complète la théorie T en une théorie Th sur $\mathscr{L}'$.
3. On quotiente pour avoir la vraie égalité dans le modèle.

Cette construction est assez similaire à la définition de $\mathbb{C}$ comme le quotient $\mathbb{R}[X]/(X^2 + 1)$.

Proposition 12.9. On peut étendre $\mathscr{L}$ en $\mathscr{L}'$ et T en T' consistante telle que, pour toute formule $F(x)$ de $\mathscr{L}'$, ayant pour seule variable libre x, il existe un symbole de constante c_F de $\mathscr{L}'$ telle que l'on ait $T' \vdash \exists x\, F(x) \to F(c_F)$, d'où le nom de témoin.

Preuve. On fait la construction « par le bas » :

- $\mathscr{L}_0 = \mathscr{L}$;
- $T_0 = T$;
- $\mathscr{L}_{n+1} = \mathscr{L}_n \cup \{c_F \mid F \text{ formule à une variable libre de } \mathscr{L}_n\}$;
- $T_{n+1} = T_n \cup \{\exists x\, F \to F(c_F) \mid F \text{ formule de } \mathscr{L}_n\}$;
- et enfin $\mathscr{L}' = \bigcup_{n\in\mathbb{N}} \mathscr{L}_n$ et $T' = \bigcup_{n\in\mathbb{N}} T_n$.

On commence par montrer quelques lemmes.

Lemme 12.9. Soient Γ un ensemble de formules et A une formule. Soit c un symbole de constante qui n'apparait ni dans Γ ni dans A. Si $\Gamma \vdash A[x := c]$ alors $\Gamma \vdash \forall x\, A$.

Preuve. Idée de la preuve. On peut supposer que x n'apparait pas dans Γ, ni dans la preuve de $\Gamma \vdash A[x := c]$, sinon on renomme x en y dans l'énoncé du lemme. Alors, de la preuve de $\Gamma \vdash A[x := c]$, on peut déduire une preuve de $\Gamma \vdash A(x)$ en replaçant c par x. Avec la règle $\forall_i$, on en conclut que $\Gamma \vdash \forall x\, A$. □

Lemme 12.10. Pour toute formule F à une variable libre x sur le langage $\mathscr{L}'$,

$$T' \vdash \exists x\, F(x) \to F(c_F).$$

Preuve. La formule F a un nombre fini de constantes (car c'est un mot fini), donc F est une formule sur $\mathscr{L}_n$ pour un certain $n \in \mathbb{N}$, donc $(\exists x\, F(x) \to F(c_F)) \in T_{n+1} \subseteq T'$. □

Il nous reste à montrer que la théorie T' est consistante.

Il suffit de montrer que tous les T_n sont consistantes. En effet, si T' est non-consistante, il existe un ensemble fini $T'' \subseteq T'$ et $T'' \vdash \bot$. Comme T'' fini, il existe un certain $n \in \mathbb{N}$ tel que $T'' \subseteq T_n$ et donc $T_n \vdash \bot$.

On montre par récurrence sur n que T_n est consistante.

- ▷ On a $T_0 = T$ qui est consistante par hypothèse.
- ▷ Supposons T_n consistante et que $T_{n+1} \vdash \bot$. Alors, il existe des formules à une variable libre $F_1, \dots, F_k$ écrites sur $\mathscr{L}_n$ et
$$T_n \cup \{\, \exists x\, F_i \to F_i(c_{F_i}) \mid 1 \leq i \leq k \,\} \vdash \bot.$$
Ainsi (exercice)
$$T_n \vdash \Big(\bigwedge_{1\leq i\leq k} \big(\exists x\, F_i \to F_i(c_{F_i})\big)\Big) \to \bot.$$
Les c_{F_i} ne sont pas dans T_n d'où, d'après le lemme 12.9, que
$$T_n \vdash \forall y_1\, \forall y_2\, \dots \forall y_n \Big(\bigwedge_{1\leq i\leq k} \big(\exists x\, F_i \to F_i(y_i)\big)\Big) \to \bot.$$
On peut montrer que (théorème logique)
$$(\star) \qquad \vdash \forall y\, (A(y) \to \bot) \leftrightarrow (\exists y\, A(y) \to \bot),$$
d'où
$$T_n \vdash \Big(\exists y_1\, \exists y_2\, \dots \exists y_n \bigwedge_{1\leq i\leq k} \big(\exists x\, F_i \to F_i(y_i)\big)\Big) \to \bot.$$
On a aussi
$$(\star\star) \qquad \vdash \exists y_1 \exists y_2 \big(A(y_1) \wedge A(y_2)\big) \leftrightarrow \big(\exists y_1 A(y_1)\big) \wedge \big(\exists y_2 A(y_2)\big),$$
et pour y non libre dans A, on a
$$\vdash \exists y\, (A \to B) \leftrightarrow (A \to \exists y\, B).$$

On a donc

$$T_n \vdash \Big(\bigwedge_{1\leq i\leq k} (\exists x\, F_i(x) \to \exists y_i\, F_i(y_i)) \Big) \to \bot.$$

Or,

$$(\star\star\star) \qquad \vdash \bigwedge_{1\leq i\leq k} (\exists x\, F_i(x) \to \exists y_i\, F_i(y_i)).$$

On a donc $T_n \vdash \bot$, ce qui contredit l'hypothèse, d'où T_{n+1} consistante.

En exercice, on pourra montrer les théorèmes logiques $(\star)$, $(\star\star)$, et $(\star\star\star)$.

□

Ensuite, on veut compléter T' en préservant le résultat de la proposition précédente. On cherche Th (axiome-)complète telle que $T' \subseteq \mathrm{Th}$ et pour toute formule à une variable libre F de $\mathscr{L}'$, on a

$$\mathrm{Th} \vdash \exists x\, F \to F(c_F).$$

Faisons le cas dénombrable (sinon, lemme de Zorn) : supposons $\mathscr{L}'$ au plus dénombrable. Soit $(F_n)_{n\in\mathbb{N}}$ une énumération des formules closes de $\mathscr{L}'$. On définit par récurrence

- $K_0 := T'$;
- si K_n est complète, alors $K_{n+1} := K_n$;
- si K_n n'est pas complet, alors soit le plus petit $p \in \mathbb{N}$ tel que l'on ait $K_n \nvdash F_p$ et $K_n \nvdash \neg F_p$, et on pose $K_{n+1} := K_n \cup \{F_p\}$.

Lemme 12.11. On pose $\mathrm{Th} := \bigcup_{n\in\mathbb{N}} T_n$. La théorie Th a les propriétés voulues.

Preuve.

1. On a $T' \subseteq \mathrm{Th}$.
2. La théorie Th est consistante. En effet, il suffit de montrer que tous les K_n le sont (par les mêmes argument que la preuve précédente). Montrons le par récurrence.
   - La théorie $K_0 = T'$ est consistante par hypothèse.

- ▷ Si $K_{n+1} = K_n$ alors K_{n+1} est consistante par hypothèse de récurrence.
- ▷ Si $K_{n+1} = K_n \cup \{F_p\}$, et si $K_n, F_p \vdash \bot$, alors par la règle $\neg_i$, on a $K_n \vdash \neg F_p$, ce qui est faux. Ainsi K_{n+1} est consistante.

On en conclut que Th est consistante.

3. La théorie Th est complète. Sinon, à chaque étape $K_{n+1} = K_n \cup \{F_{q_n}\}$ et il existe F_p telle que $\mathrm{Th} \nvdash F_p$ et $\mathrm{Th} \nvdash \neg F_p$. Ainsi, pour tout $n \in \mathbb{N}$, $K_n \nvdash F_p$ et $K_n \nvdash \neg F_p$, d'où pour tout $n \in \mathbb{N}$, $p_n \leq p$ avec des p_n distincts. C'est absurde, il n'y a qu'un nombre fini d'entiers inférieurs à un entier donné.

□

On construit un quotient avec « = » comme relation d'équivalence, puis on vérifie que les fonctions et relations sont bien définies (ne dépendent pas du représentant choisit, comme pour les groupes quotients).

Soit $\mathscr{C}$ l'ensemble des termes clos de $\mathscr{L}'$, qui n'est pas vide car il contient les termes $c_{x=x}$ (avec la définition de c_F ci-avant). On définit sur $\mathscr{C}$ une relation $\sim$, où $t \sim t'$ ssi $\mathrm{Th} \vdash t = t'$.

Exercice 12.6. Montrer que $\sim$ est une relation d'équivalence.

On pose enfin $|\mathscr{M}| := \mathscr{C}/\sim$. On notera $\bar{t}$ la casse de t. On définit l'interprétation des symboles de $\mathscr{L}'$:

- ▷ si c est une constante, alors $c_{\mathscr{M}} := \bar{c}$;
- ▷ si f est un symbole de fonctions d'arité n,

$$f_{\mathscr{M}}(\bar{t}_1, \ldots, \bar{t}_n) := \overline{f(t_1, \ldots, t_n)}.$$

Lemme 12.12. La définition de dépend pas des représentants

choisis, c'est-à-dire si $\bar{u}_1 = \bar{t}_1, \dots, \bar{u}_n = \bar{t}_n$ alors

$$\overline{f(t_1, \dots, t_n)} = \overline{f(u_1, \dots, u_n)}.$$

Preuve.
- ▷ On a $\mathrm{Th} \vdash t_i = u_i$ pour tout i par hypothèse
- ▷ donc avec $=_{\mathsf{i}}$, on a $\mathrm{Th} \vdash f(t_1, \dots, t_n) = f(t_1, \dots, t_n)$
- ▷ donc avec $=_{\mathsf{e}}$, on a $\mathrm{Th} \vdash f(u_1, \dots, t_n) = f(t_1, \dots, t_n)$
- ▷ *...etc...*
- ▷ donc avec $=_{\mathsf{e}}$, on a $\mathrm{Th} \vdash f(u_1, \dots, u_n) = f(t_1, \dots, t_n)$

□

[*suite de la définition de l'interprétation*]

- ▷ si R est un symbole de relation d'arité n, on définit

$$(\bar{t}_1, \dots, \bar{t}_n) \in R_{\mathscr{M}} \text{ ssi } \mathrm{Th} \vdash R(t_1, \dots, t_n).$$

Exercice 12.7. Montrer que cette définition de dépend pas des représentants choisis.

Lemme 12.13. Soit F une formule à n variables libres et $t_1, \dots, t_n$ des termes clos. Alors, $\mathscr{M} \models F[\bar{t}_1, \dots, \bar{t}_n]$ ssi $\mathrm{Th} \vdash F[t_1, \dots, t_n]$, où l'on interprète la formule à paramètre dans l'environnement e avec $e(y_i) = \bar{t}_i$ alors $\mathscr{M}, e \models F(y_1, \dots, y_n)$.

Preuve. Par induction sur F en supposant que F n'utilise que $\neg$, $\vee$, $\exists$ comme connecteurs. En effet, on a pour toute formule G, il existe F qui n'utilise que $\neg$, $\vee$, $\exists$ et $\vdash F \leftrightarrow G$, ce qui permet de conclure directement pour G si le résultat est vrai sur F.

- ▷ Pour $F = \bot$, alors on a $\mathrm{Th} \nvdash \bot$ car Th consistante et $\mathscr{M} \models \bot$ par définition.
- ▷ Pour $F = R(u_1, \dots, u_m)$, où les u_i sont des termes non nécessairement clos et où $u_1, \dots, u_m$ sont des termes à n

variables $x_1, \dots, x_n$. On pose

$$F[t_1, \dots, t_n] := R(\underbrace{u_1(t_1, \dots, t_n)}_{v_1}, \dots, \underbrace{u_m(t_1, \dots, t_n)}_{v_m})$$

où l'on définit $v_i := u_i(t_1, \dots, t_n)$ qui est clos car les t_i sont clos. On veut montrer que

$$\mathscr{M} \models \underbrace{F[\bar{t}_1, \dots, \bar{t}_n]}_{R(\bar{v}_1, \dots, \bar{v}_m)} \text{ ssi } \mathrm{Th} \vdash \underbrace{F[t_1, \dots, t_n]}_{R(v_1, \dots, v_m)}.$$

Or, on a l'équivalence $\mathscr{M} \models R(\bar{v}_1, \dots, \bar{v}_m)$ ssi $(\bar{v}_1, \dots, \bar{v}_m) \in R_{\mathscr{M}}$ ssi $\mathrm{Th} \vdash R(v_1, \dots, v_m)$.

- ▷ Pour $F = F_1 \vee F_2$, et $t_1, \dots, t_n$ sont des termes clos, on veut montrer que

$$\begin{aligned} &\mathscr{M} \models F_1[\bar{t}_1, \dots, \bar{t}_n] \vee F_2[\bar{t}_1, \dots, \bar{t}_n] \\ \text{ssi } &\mathrm{Th} \vdash F_1[t_1, \dots, t_n] \vee F_2[t_1, \dots, t_n]. \end{aligned}$$

Or,

$$\begin{aligned} &\mathscr{M} \models F_1[\bar{t}_1, \dots, \bar{t}_n] \vee F_2[\bar{t}_1, \dots, \bar{t}_n] \\ \text{ssi } &\mathscr{M} \models F_1[\bar{t}_1, \dots, \bar{t}_n] \text{ ou } \mathscr{M} \models F_2[\bar{t}_1, \dots, \bar{t}_n] \\ \text{ssi } &\mathrm{Th} \vdash F_1[t_1, \dots, t_n] \text{ ou } \mathrm{Th} \vdash F_2[t_1, \dots, t_n] \end{aligned}$$

par hypothèse. Ainsi,

  - avec $\vee_{\mathsf{i}}^{\mathsf{g}}$ et $\vee_{\mathsf{i}}^{\mathsf{d}}$, on a que $\mathrm{Th} \vdash F_1[t_1, \dots, t_n] \vee F_2[t_1, \dots, t_n]$;
  - réciproquement, on utilise le lemme 12.5 car Th est complète.

- ▷ Pour $F = \neg G$, en exercice.
- ▷ Si $F = \exists x\, G$ et $t_1, \dots, t_n$ des termes clos, on a
  - on a $\mathscr{M} \models \exists x\, G[\bar{t}_1, \dots, \bar{t}_n, x]$
  - ssi il existe $t \in \mathscr{C}$ tel que $\mathscr{M} \models G[\bar{t}_1, \dots, \bar{t}_n, \bar{t}]$
  - ssi il existe $t \in \mathscr{C}$ tel que $\mathrm{Th} \vdash G(t_1, \dots, t_n, t)$

et donc $\mathrm{Th} \vdash \exists x G(t_1, \ldots, t_n, x)$ avec $\exists_{\mathsf{i}}$. Réciproquement, si $\mathrm{Th} \vdash \exists x G(t_1, \ldots, t_n, x)$ alors $\mathrm{Th} \vdash G(t_1, \ldots, t_n, c_{G(t_1,\ldots,t_n,x)})$, donc il existe un terme t et $\mathrm{Th} \vdash G(t_1, \ldots, t_n, t)$.

□

Lemme 12.14. On a $\mathscr{M} \models \mathrm{Th}$ (et donc $\mathscr{M} \models T$).

Preuve. On montre que, pour toute formule F de Th, on a que $\mathscr{M} \models F$. Pour cela, on utilise le lemme précédent : si F est close, alors

$$\mathscr{M} \models F \text{ ssi } \mathrm{Th} \vdash F.$$

□

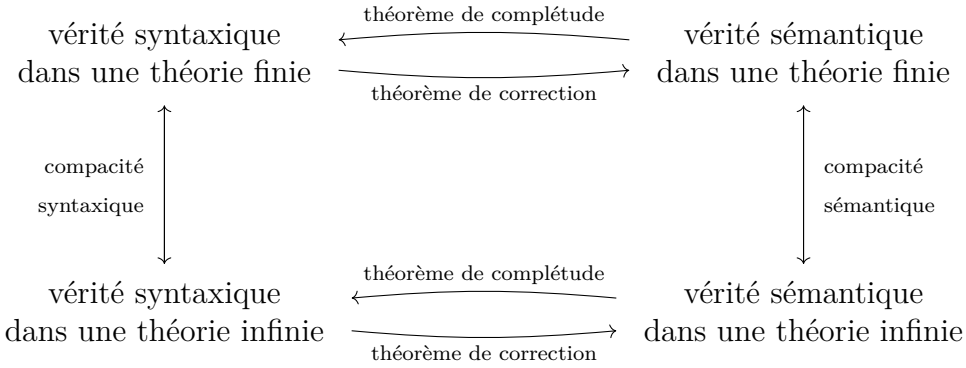
12.5.3 Compacité.

Théorème 12.2 (Compacité (sémantique))**.** Une théorie T et contradictoire ssi elle est finiment contradictoire, *i.e.* il existe $T' \subseteq_{\text{fini}} T$ telle que T' est contradictoire.

Preuve. Soit T contradictoire. On utilise le théorème de complétude. Ainsi T est inconsistante. Il existe donc $T' \subseteq_{\text{fini}} T$ avec T' inconsistante par le théorème de compacité syntaxique ci-dessous (qui est trivialement vrai). On applique de nouveau le théorème de complétude pour en déduire que T' est contradictoire. □

Théorème 12.3 (Compacité (syntaxique))**.** Une théorie T est inconsistante ssi elle est finiment inconsistante.

Preuve. Ceci est évident car une preuve est nécessairement finie. □



Dans la suite de cette sous-section, on étudie des applications du théorème de compacité.

Théorème 12.4. Si une théorie T a des modèles finis arbitrairement grands, alors elle a un modèle infini. □

Corollaire 12.4. Il n'y a pas de théorie des groupes finis *i.e.* un ensemble d'axiomes dont les modèles sont exactement les groupes finis.

Théorème 12.5 (Löwenheim-Skolem)**.** Soit T une théorie dans un langage $\mathscr{L}$ et κ un cardinal et $\kappa \geq \operatorname{card} \mathscr{L}$ et $\kappa \geq \aleph_0$.[7] Si T a un modèle infini, alors T a un modèle de cardinal κ.

Exemple 12.22. ▷ Avec $T = \mathbf{Th}(\mathbb{N})$, on a $\kappa = \operatorname{card} \mathbb{R}$.
▷ Avec $T = \mathbf{ZFC}$, on a $\kappa = \aleph_0 = \operatorname{card} \mathbb{N}$.

7. Ici, $\aleph_0$ est le cardinal de $\mathbb{N}$, on dit donc que κ est infini.

13. L'arithmétique de Peano.

- ▷ Dedekind (1988) et Peano (1889) formalisent l'arithmétique.
- ▷ En 1900, David Hilbert, lors du 2ème ICM à Paris, donne un programme et dont le 2nd problème est la *cohérence de l'arithmétique*.
- ▷ En 1901, Russel donne son paradoxe concernant l'« ensemble » de tous les ensembles.
- ▷ En 1930, (Hilbert) est toujours optimiste : « On doit savoir, on saura ! »

La formalisation de l'arithmétique engendre deux questions :

1. est-ce que tout théorème est prouvable ? (▷ complétude)
2. existe-t-il un algorithme pour décider si un théorème est prouvable ? (▷ décidabilité)

Le second point est appelé « *Entscheidungsproblem* », le problème de décision, en 1928.

- ▷ En 1931, Gödel répond non à ces deux questions.

On a donné plusieurs formalisations des algorithmes :

- ▷ en 1930, le λ-calcul de Church ;
- ▷ en 1931–34, les fonctions récursives de Herbrand et Gödel ;
- ▷ en 1936, les machines de Turing.

On démontre que les trois modèles sont équivalents.

La thèse de Church–Turing nous convainc qu'il n'existe pas de modèle plus évolué « dans la vraie vie ».

13.1 Les axiomes.

On définit le langage $\mathscr{L}_0 = \{Ⓞ, Ⓢ, \oplus, \otimes\}$ où

- ▷ $Ⓞ$ est un symbole de constante ;
- ▷ $Ⓢ$ est un symbole de fonction unaire ;
- ▷ $\oplus$ et $\otimes$ sont deux symboles de fonctions binaires.

On verra plus tard que l'on peut ajouter une relation binaire $\leq$.

Remarque 13.1 (Convention). La structure $\mathbb{N}$ représente la $\mathscr{L}_0$-structure dans laquelle on interprète les symboles de manière habituelle :

- ▷ pour $Ⓞ$, c'est 0 ;
- ▷ pour $Ⓢ$, c'est $\lambda n.n+1$ (*i.e.* $x \mapsto x+1$) ;
- ▷ pour $\oplus$, c'est $\lambda n\, m.n+m$;
- ▷ pour $\otimes$, c'est $\lambda n\, m.n \times m$.

Les axiomes de Peano.

On se place dans le cas égalitaire. L'ensemble $\mathscr{P}$ est composé de $\mathscr{P}_0$ un ensemble fini d'axiomes (A1–A7) et d'un schéma d'induction (SI).

Trois axiomes pour le successeur :

A1. $\forall x\, \neg(Ⓢ x = Ⓞ)$

A2. $\forall x\, \exists y\, \Big(\neg(x = Ⓞ) \to x = Ⓢ y\Big)$

A3. $\forall x\, \forall y\, (Ⓢ x = Ⓢ y \to x = y)$

Deux axiomes pour l'addition :

A4. $\forall x\, (x \oplus Ⓞ = x)$

A5. $\forall x\, \forall y\, (x \oplus (Ⓢ y) = Ⓢ(x \oplus y))$

Deux axiomes pour la multiplication :

A6. $\forall x\, (x \otimes Ⓞ = Ⓞ)$

A7. $\forall x\, \forall y\, \Big(x \otimes (Ⓢ y) = (x \otimes y) \oplus x\Big)$

Et le schéma d'induction :

SI. Pour toute formule F de variables libres $x_0, \ldots, x_n$,

$$\forall x_1 \cdots \forall x_n \Big(\big(F(\textcircled{0}, \ldots, x_1, \ldots, x_n) \wedge \forall x \, (F(x, x_1, \ldots, x_n) \to F(\textcircled{S} x, x_1, \ldots, x_n)) \big) \to \forall x \, F(x, x_1, \ldots, x_n) \Big).$$

Remarque 13.2. ▷ Le schéma est le SI avec hypothèse faible, qui permet de montrer le SI avec hypothèse forte. On adopte la notation $\forall y \leq x \; F(y, x_1, \ldots, x_n)$ pour

$$\forall y \Big((\exists z \; z \oplus y = x) \to F(y, x_1, \ldots, x_n) \Big).$$

Le SI avec hypothèse forte est :

$$\forall x_1 \cdots \forall x_n \Big(\big(F(\textcircled{0}, \ldots, x_1, \ldots, x_n) \wedge \forall x \, \big((\forall y \leq x \; F(y, x_1, \ldots, x_n)) \to F(\textcircled{S} x, x_1, \ldots, x_n) \big) \big) \to \forall x \, F(x, x_1, \ldots, x_n) \Big)$$

- ▷ L'ensemble $\mathscr{P}$ est non-contradictoire car $\mathbb{N}$ est un modèle, appelé *modèle standard*.
- ▷ On peur remplacer le SI par une nouvelle règle de démonstration :

$$\frac{\Gamma \vdash F(\textcircled{0}) \qquad \Gamma \vdash \forall y \Big(F(y) \to F(\textcircled{S} y) \Big)}{\Gamma \vdash \forall x \, F(x)} \; \mathsf{rec} \; .$$

Exercice 13.1. Montrer l'équivalence entre SI et la nouvelle règle rec, *i.e.* on peut démontrer les mêmes théorèmes.

Notation. On note $\textcircled{n}$ le terme $\underbrace{\textcircled{S} \cdots \textcircled{S}}_{n \text{ fois}} \textcircled{0}$ pour $n \in \mathbb{N}$.

Définition 13.1. Dans une $\mathscr{L}_0$-structure, on dit qu'un élément est *standard* s'il est l'interprétation d'un terme $\textcircled{n}$ avec $n \in \mathbb{N}$.

Remarque 13.3. Dans $\mathbb{N}$ (le modèle standard), tout élément est standard.

Théorème 13.1. Il existe des modèles de $\mathscr{P}$ non isomorphes à $\mathbb{N}$.

Preuve. 1. Avec le théorème de Löwenheim-Skolem, il existe un modèle de $\mathscr{P}$ de cardinal κ pour tout $\kappa \geq \aleph_0$, et $\operatorname{card} \mathbb{N} = \aleph_0$.

2. Autre preuve, on considère un symbole de constante c et on pose $\mathscr{L} := \mathscr{L}_0 \cup \{c\}$. On considère la théorie

$$T := \mathscr{P} \cup \{ \neg(c = \textcircled{n}) \mid n \in \mathbb{N} \}.$$

Montrons que T a un modèle. Par le théorème de compacité de la logique du premier ordre, il suffit de montrer que T est finiment satisfiable. Soit $T' \subseteq_{\text{fini}} T$: par exemple,

$$T' \subseteq \mathscr{P} \cup \{\neg(c = \textcircled{n_1}), \neg(c = \textcircled{n_2}), \dots, (c = \textcircled{n_k})\},$$

et $n_k \geq n_1, \dots, n_{k-1}$. On construit un modèle de T' correspondant à $\mathbb{N}$ où c est interprété par $n_k + 1$. Ainsi, T' est satisfiable et donc T aussi avec un modèle $\mathscr{M}$.

Montrons que $\mathbb{N}$ et $\mathscr{M}$ ne sont pas isomorphes. Par l'absurde, supposons que $\varphi : \mathscr{M} \to \mathbb{N}$ soit un isomorphisme. Alors $\gamma := \varphi(c_{\mathscr{M}})$ satisfait les mêmes formules que $c_{\mathscr{M}}$, par exemple, pour tout $n \in \mathbb{N}$, $\mathscr{M} \models \neg(c = \textcircled{n})$. Or, on ne peut pas avoir $\mathbb{N} \models \neg(\textcircled{\gamma} = \textcircled{n})$ pour tout $n \in \mathbb{N}$. ***Absurde.***

□

On a montré que tous les modèles isomorphes à $\mathbb{N}$ n'ont que des éléments standards.

Théorème 13.2. Dans tout modèle $\mathscr{M}$ de $\mathscr{P}$,

1. l'addition est commutative et associative ;
2. la multiplication aussi ;
3. la multiplication est distributive par rapport à l'addition ;
4. tout élément est *régulier* pour l'addition :

$$\mathscr{M} \models \forall x \, \forall y \, \forall z \, (x \oplus y = x \oplus z \to y = z) \, ;$$

5. tout élément non nul est régulier pour la multiplication :

$$\mathscr{M} \models \forall x\, \forall y\, \forall z\, ((\neg(x = \textcircled{0}) \wedge x \otimes y = x \otimes z) \to y = z)\ ;$$

6. la formule suivante définie un ordre total sur $\mathscr{M}$ compatible avec $+$ et $\times$:

$$x \leq y \text{ ssi } \exists z\, (x \oplus z = y).$$

Preuve. On prouve la commutativité de $+$ en trois étapes.

1. On montre $\mathscr{P} \vdash \forall x\, (\textcircled{0} \oplus x = x)$. On utilise le SI avec la formule $F(x) := (\textcircled{0} \oplus x = x)$.
   - ▷ On a $\mathscr{P} \vdash \textcircled{0} \oplus \textcircled{0} = \textcircled{0}$ par A4.
   - ▷ On montre $\mathscr{P} \vdash \forall x\, F(x) \to F(\textcircled{S} x)$, c'est à dire :

$$\forall x \left((\textcircled{0} \oplus x = x) \to (\textcircled{0} \oplus (\textcircled{S} x) = \textcircled{S} x) \right).$$

     On peut le montrer par A5.

   Questions/Remarques :
   - ▷ Pourquoi pas une récurrence normale ? On n'est pas forcément dans $\mathbb{N}$!
   - ▷ Grâce au théorème de complétude, on peut raisonner sur les modèles, donc en maths naïves.
2. On montre $\mathscr{P} \vdash \forall x \forall y\, \textcircled{S}(x \oplus y) = (\textcircled{S} x) \oplus y$. On veut utiliser le schéma d'induction avec $F(x, y) := \textcircled{S}(x \oplus y) = (\textcircled{S} x) \oplus y$. Mais ça ne marche pas...(Pourquoi ?)

   La bonne formule est $F(y, x) := \textcircled{S}(x \oplus y) = (\textcircled{S} x) \oplus y$.
   - ▷ On montre $\mathscr{P} \vdash F(\textcircled{0}, x)$, c'est à dire

$$\mathscr{P} \vdash \textcircled{S}(x \oplus \textcircled{0}) = (\textcircled{S} x) \oplus \textcircled{0}.$$

     Ceci est vrai car

$$\textcircled{S}(x \oplus \textcircled{0}) \underset{\text{A4}}{=} \textcircled{S} x \underset{\text{A4}}{=} (\textcircled{S} x) \oplus \textcircled{0}.$$

▷ On a $\mathscr{P} \vdash F(y,x) \to F(\textcircled{S}y, x)$ car : si $\textcircled{S}(x \oplus y) = (\textcircled{S}x) \oplus y$, alors

$$\textcircled{S}(x \oplus (\textcircled{S}y)) \underset{\text{A5}}{=} \textcircled{S}\textcircled{S}(x \oplus y) \underset{\text{hyp}}{=} \textcircled{S}((\textcircled{S}x) \oplus y) \underset{\text{A5}}{=} (\textcircled{S}x) = \oplus(\textcircled{S}y).$$

3. On utilise le SI avec $F(x,y) := (x \oplus y = y \oplus x)$. D'une part, on a $F(\textcircled{0}, y) = (\textcircled{0} \oplus y = y \oplus \textcircled{0})$ par 1 et A4. D'autre part, si l'on a $x \oplus y = y \oplus x$ alors $(\textcircled{S}x) \oplus y = y \oplus (\textcircled{S}x)$ par A5 et 2. Par le SI, on conclut.

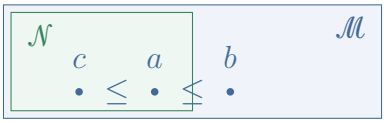
□

Exercice 13.2. Finir la preuve du théorème.

13.2 Liens entre $\mathbb{N}$ et un modèle $\mathscr{M}$ de $\mathscr{P}$.

Définition 13.2. Si $\mathscr{M} \models \mathscr{P}_0$ et $\mathscr{N} \models \mathscr{P}_0$ et $\mathscr{N}$ une sous-interprétation de $\mathscr{M}$, on dit que $\mathscr{N}$ est un segment initial de $\mathscr{M}$, ou que $\mathscr{M}$ est une extension finale de $\mathscr{N}$, si pour tous $a, b, c \in |\mathscr{M}|$ avec $a \in |\mathscr{N}|$ on a :

1. si $\mathscr{M} \models c \leq a$ alors $c \in |\mathscr{N}|$;
2. si $b \notin |\mathscr{N}|$ alors $\mathscr{M} \models a \leq b$.



Remarque 13.4.
▷ Les points peuvent être incomparables et dans $\mathscr{M}$.
▷ L'ensemble $\mathscr{P}_0$ est très faible, on ne montre même pas que $\oplus$ commute ou que $\leq$ est une relation d'ordre (*c.f.* TD).

Théorème 13.3. Soit $\mathscr{M} \models \mathscr{P}_0$. Alors, le sous-ensemble de $\mathscr{M}$

suivant est une sous-interprétation de $\mathscr{M}$ qui est un segment initial et qui est isomorphe à $\mathbb{N}$:

$$\left\{ a \in |\mathscr{M}| \;\middle|\; \begin{array}{l} \text{il existe } n \in \mathbb{N} \text{ et } a \\ \text{est l'interprétation} \\ \text{de } \underline{n} \text{ dans } \mathscr{M} \end{array} \right\}.$$

Preuve.
1. Pour tout $n \in \mathbb{N}$, on a $\mathscr{P}_0 \vdash \underline{n+1} = \underline{\mathbf{S}}\,\underline{n}$.
2. Pour tout $n, m \in \mathbb{N}$, on a $\mathscr{P}_0 \vdash \underline{m} \oplus \underline{n} = \underline{m+n}$.
3. Pour tout $n, m \in \mathbb{N}$, on a $\mathscr{P}_0 \vdash \underline{m} \otimes \underline{n} = \underline{m \times n}$.
4. Pour tout $n \in \mathbb{N}_\star$, on a $\mathscr{P}_0 \vdash \neg(\underline{n} = \underline{0})$.
5. Pour tout $n \neq m$, on a $\mathscr{P}_0 \vdash \neg(\underline{m} = \underline{n})$.
6. Pour tout $n \in \mathbb{N}$ (admis), on a
$$\mathscr{P}_0 \vdash \forall x \left(x \leq \underline{n} \to (x = \underline{0} \vee x = \underline{1} \vee \cdots \vee x = \underline{n}) \right).$$
7. Pour tout x, on a $\mathscr{P}_0 \vdash \forall x\, (x \leq \underline{n} \vee \underline{n} \leq x)$.

□

13.3 Les fonctions représentables.

Cette section détaille un outil technique pour montrer le théorème d'incomplétude de Gödel vu plus tard. On code tout avec des entiers !

Définition 13.3. Soit $f : \mathbb{N}^p \to \mathbb{N}$ une fonction totale et $F(x_0, \ldots, x_p)$ une formule de $\mathscr{L}_0$. On dit que F *représente* f si, pour tout p-uplet d'entiers $(n_1, \ldots, n_p)$ on a :

$$\mathscr{P}_0 \vdash \forall y \left(F(y, \underline{n_1}, \ldots, \underline{n_p}) \leftrightarrow y = \underline{f(n_1, \ldots, n_p)} \right).$$

On dit que f est *représentable* s'il existe une formule qui la représente.

Un ensemble de p-uplets $A \subseteq \mathbb{N}^p$ est *représenté* par $F(x_1, \ldots, x_p)$

si pour tout p-uplet d'entiers $(n_1, \dots, n_p)$, on a

1. si $(n_1, \dots, n_p) \in A$ alors $\mathscr{P}_0 \vdash F(n_1, \dots, n_p)$;
2. si $(n_1, \dots, n_p) \notin A$ alors $\mathscr{P}_0 \vdash \neg F(n_1, \dots, n_p)$.

On dit que A est *représentable* s'il existe une formule qui le représente.

Exercice 13.3. Montrer qu'un ensemble est représentable ssi sa fonction indicatrice l'est.

Exemple 13.1 (Les briques de base des fonctions récursives)**.**

- La fonction nulle $f : \mathbb{N} \to \mathbb{N}, x \mapsto 0$ est représentable par $F(x_0, x_1) := x_0 = \textcircled{0}$.
- Les fonctions constantes $f : \mathbb{N} \to \mathbb{N}, x \mapsto n$ sont représentables par $F(x_0, x_1) := x_0 = \textcircled{n}$, où $n \in \mathbb{N}$.
- Les projections $\pi_p^i : \mathbb{N}^p \to \mathbb{N}, (x_1, \dots, x_p) \mapsto x_i$ sont représentables par $F(x_0, x_1, \dots, x_p) := x_0 = x_i$.
- La fonction successeur $f : \mathbb{N} \to \mathbb{N}, x \mapsto x + 1$ est représentable par $F(x_0, x_1) := x_0 = (\textcircled{S}\, x_1)$.
- L'addition $f : \mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x + y$ est représentable par $F(x_0, x_1, x_2) := x_0 = x_1 \oplus x_2$.
- La multiplication $f : \mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x \times y$ est représentable par $F(x_0, x_1, x_2) := x_0 = x_1 \otimes x_2$.

On introduit trois nouvelles opérations.

Récurrence. Soient $g(x_1, \dots, x_p)$ et $h(x_1, \dots, x_{p+2})$ des fonctions partielles. On définit la fonction partielle f par :

- $f(0, x_1, \dots, x_p) := g(x_1, \dots, x_p)$;
- $f(x_0 + 1, x_1, \dots, x_p) := h(x_0, f(x_0, \dots, x_p), x_1, \dots, x_p)$.

Composition. Soient $f_1, \dots, f_n$ des fonctions partielles de p variables et g une fonction partielle de n variables. Alors, la fonction composée $g(f_1, \dots, f_n)$ est définie en $(x_1, \dots, x_p)$ ssi les fonctions f_i le sont et g est définie en $\big(f_1(x_1, \dots, x_p), \dots, f_n(x_1, \dots, x_p)\big)$.

Schéma μ. Soit $f(x_1, \dots x_{p+1})$ une fonction partielle. Soit

$$g(x_1, \dots, x_p) := \mu y.\ (f(x_1, \dots, x_p, y) = 0).$$

Elle est définie en $(x_1, \dots, x_p)$ si et seulement s'il existe y tel que $f(x_1, \dots, x_p, y) = 0$ et tous les $f(x_1, \dots, x_p, x)$ sont définies pour $x \leq y$. Dans ce cas, $g(x_1, \dots, x_p)$ est le plus petit y tel que $f(x_1, \dots, x_p, y) = 0$.

Définition 13.4. L'ensemble des fonctions récursives primitives (*resp.* récursives) est le plus petit ensemble des fonctions contenant les briques de base et stable par composition et récurrence (*resp.* par composition, récurrence et schéma μ).

Exemple 13.2. Les fonctions

$$f(x_1, x_2, y) := y^2 - (x_1 + x_2)y + x_1 x_2$$

et

$$f(x_1, x_2) := \min(x_1, x_2)$$

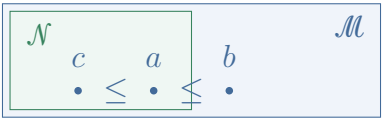
sont récursives primitives.

Définition 13.5. Une fonction récursive *totale* est une fonction récursive définie partout.

Remarque 13.5.
- ▷ Une fonction récursive primitive est totale.
- ▷ Une fonction récursive primitive peut se fabriquer avec un seul schéma μ à la fin (*c.f.* cours de FDI).
- ▷ *Rappel.* Une fonction $f : \mathbb{N}^p \to \mathbb{N}$ totale est représentée par la formule $F(x_0, \dots, x_p)$ de $\mathcal{L}_0$ su pour tout p-uplet d'entiers $(n_1, \dots, n_p)$ on a :

$$\mathcal{P}_0 \vdash \forall y \left(F(y, \overline{n_1}, \dots, \overline{n_p}) \leftrightarrow y = \overline{f(n_1, \dots, n_p)}\right).$$

- ▷ *Rappel.* Si $\mathscr{M} \models \mathscr{P}_0$ alors l'ensemble de $|\mathscr{M}|$ constitué de l'interprétation des termes standards est une sous-interprétation de $\mathscr{M}$ qui en est un segment initial et qui est isomorphe à $\mathbb{N}$.
- ▷ *Rappel.* Une sous-interprétation $\mathscr{N}$ est un segment initial de $\mathscr{M}$ si
  - $a \in \mathscr{N}$ et $b \in \mathscr{M} \setminus \mathscr{N}$ alors $b \geq a$;
  - $a \in \mathscr{N}$ et $c \leq a$ alors $c \in \mathscr{N}$.



Théorème 13.4. Toute fonction récursive totale est représentable.

On a déjà montré que les briques de base sont représentables. On montre trois lemmes qui montreront le théorème ci-dessus.

Lemme 13.1. L'ensemble des fonctions représentables est clos par composition.

Preuve. Soient $f_1(x_1, \ldots, x_p), \ldots, f_n(x_1, \ldots, x_p)$ et $g(x_1, \ldots, x_n)$ des fonctions représentées par $F_1(x_0, \ldots, x_p), \ldots, F_n(x_0, \ldots, x_p)$ et $G(x_0, \ldots, G_n)$. On va montrer que $h = g(f_1, \ldots, f_n)$ est représentée par

$$H(x_0, \ldots, x_o) := \exists y_0 \cdots \exists y_n \Big(G(x_0, y_1, \ldots, y_n) \wedge \bigwedge_{1 \leq i \leq n} F_i(y_i, x_1, \ldots, x_p) \Big).$$

En effet, pour tous entiers $n_1, \ldots, n_{\max(p,n)}$:

- ▷ $\mathscr{P}_0 \vdash \forall y\, F_i(y_1, \overline{n_1}, \ldots, \overline{n_p}) \leftrightarrow y = \overline{f_i(n_1, \ldots, n_p)}$;
- ▷ $\mathscr{P}_0 \vdash \forall y\, G(y_1, \overline{n_1}, \ldots, \overline{n_n}) \leftrightarrow y = \overline{g(n_1, \ldots, n_n)}$.

Dans tout modèle $\mathscr{M}$ de $\mathscr{P}_0$, pour tout $y \in |\mathscr{M}|$, et tous $n_1, \dots, n_p \in \mathbb{N}$ on a $H(y, n_1, \dots, n_p)$ est vraie ssi il existe $y_1, \dots, y_n$ dans $|\mathscr{M}|$ et pour tout i, $F_i(y_i, x_1, \dots, x_p)$ est vrai et $G(y, y_1, \dots, y_n)$. Donc, par les hypothèses précédents, on a $H(y, n_1, \dots, n_p)$ ssi il existe $y_1, \dots, y_n$ dans $|\mathscr{M}|$ et pour tout i, $y_i = f_i(n_1, \dots, n_p)$ et $y = g(y_1, \dots, y_p)$, ssi

$$y = g(f_1(n_1, \dots, n_p), \dots, f_n(n_1, \dots, n_p))$$

ssi $y = h(n_1, \dots, n_p)$. On conclut

$$\mathscr{P}_0 \vdash \forall y \left(H(y, \overline{n_1}, \dots, \overline{n_p}) \leftrightarrow y = \overline{h(n_1, \dots, n_p)} \right).$$

□

Lemme 13.2. Si, à partir d'une fonction représentable totale, on obtient par schéma μ une fonction totale, alors cette fonction est représentable.

Preuve. Soit $g : \mathbb{N}^{p+1} \to \mathbb{N}$ une fonction représentable totale, et soit $f : \mathbb{N}^p \to \mathbb{N}$ définie par

$$f(x_1, \dots, x_p) := \mu x_0. \left(g(x_0, \dots, x_p) = 0 \right).$$

Montrons que si f est totale alors elle est représentable. Soit $G(y, x_0, \dots, x_p)$ qui représente g. Alors, pour tous $n_1, \dots, n_p$ on a

$$\mathscr{P}_0 \vdash \forall y \, G(y, \overline{n_1}, \dots, \overline{n_p}) \leftrightarrow y = \overline{g(n_1, \dots, n_p)}.$$

Considérons la formule

$$F(y, n_1, \dots, n_p) := G(0, y, x_1, \dots, x_p) \wedge \forall z < y, \neg G(0, z, x_1, \dots, x_p),$$

où l'on note $\forall z < y \, H$ pour $\forall z \, (\exists u \, \neg(h = \overline{0}) \wedge z \oplus h = y) \to H$. Montrons que F représente f. Soit $\mathscr{M}$ un modèle de $\mathscr{P}_0$. Soient $n_1, \dots, n_p$ des entiers et $y \in |\mathscr{M}|$. On a $F(y, n_1, \dots, n_p)$ vrai ssi $G(0, y, n_1, \dots, n_p)$ vrai et, pour tout $z < y$, $\neg G(0, z, n_1, \dots, n_p)$

est vrai. Montrons que $b := f(n_1, \dots, n_p)$ est le seul élément à satisfaire $F(y, n_1, \dots, n_p)$. On a bien $G(0, b, n_1, \dots, n_p)$ par définition de f et pour tout entier $z < b$, on a $\neg G(0, z, n_1, \dots, n_p)$. Mais, si on a $z < b$ et z n'est pas un entier ? Ce cas n'existe pas car la sous-représentation isomorphe à $\mathbb{N}$ est un segment initial, il n'y a donc que des entiers qui sont inférieurs à b dans $|\mathscr{M}|$. Ainsi, $F(b, n_1, \dots, n_p)$. Montrons que b est le seul. Soit y tel que $F(y, n_1, \dots, n_p)$. Montrons que $y = b$.

- Si y est un entier, c'est vrai par définition de b.
- Si y n'est pas un entier, alors $y > b$. Donc, $g(y, x_1, \dots, x_p) = 0$ et $b < y$ avec $g(b, x_1, \dots, x_p) = 0$. Ainsi, $\forall z < y \neg G(0, z, x_1, \dots, x_p)$ est fausse, et donc $F(y, n_1, \dots, n_p)$ est fausse.

□

Lemme 13.3. L'ensemble des fonctions totales est stable par définition par récurrence.

Preuve. Soient f, g, h telles que

- $f(0, x_1, \dots, x_p) = g(x_1, \dots, x_p)$
- $f(x_0 + 1, x_1, \dots, x_p) = h(x_0, f(x_0, \dots, x_p), x_1, \dots, x_p)$

Soient G, H représentant g et h. On a dans $\mathbb{N}$: $y = f(x_0, \dots, x_p)$ ssi il existe $z_0, \dots, z_{x_0}$ tel que

- $z_0 = g(x_1, \dots, x_p)$
- $z_1 = h(0, z_0, x_1, \dots, x_p)$
- $z_2 = h(1, z_1, x_1, \dots, x_p)$
- $\vdots$
- $z_{x_0} = h(x_0 - 1, z_{x_0 - 1}, x_1, \dots, x_p)$
- $y = z_{x_0}$

Zut ! On ne peut pas écrire $\exists z_0 \cdots \exists z_{x_0}$! On va utiliser une fonction qui permet de coder une suite d'entiers dans un couple d'entier (a, b). Interruption de la preuve. □

Lemme 13.4 (Fonction β de Gödel)**.** Il existe une fonction β à trois variables, récursive primitive et représentable, tel que pour tout $p \in \mathbb{N}$ et toute suite $(n_0, \dots, n_p) \in \mathbb{N}^{p+1}$, il existe des entiers a et b tels que pour tout $0 \leq i \leq p$, on ait $\beta(i, a, b) = n_i$.

Preuve. Soient $(a_0, \dots, a_p)$ une suite d'entiers deux à deux premiers, et $(n_0, \dots, n_p)$ une suite d'entiers. Alors il existe $b \in \mathbb{N}$ tel que, pour tout $0 \leq i \leq p$, $b \equiv n_i \pmod{a_i}$ (par le théorème Chinois).

Choisissons a et les a_i (qui induisent b) ? On pose $a = m!$. Alors, on pose $a_i := a(i+1)+1$ pour tout $0 \leq i \leq p$. Les a_i sont bien deux à deux premiers. En effet, pour $j > i$, si $c \mid a_i$ et $c \mid a_j$ avec c premier, alors $c \mid (a_i - a_j)$ donc $c \mid a(j-i)$ et donc $c \leq m$, donc $c \mid m$. Ainsi, il existe bien b tel que $b \equiv n_i \pmod{a_i}$. On définit ainsi $\beta(i, a, b)$ comme le reste de la division de b par $a(i+1)+1$. La fonction β est représentée par

$$B(x_0, i, a, b) := \exists x_4\, b = x_4 \otimes \textcircled{S}(a \otimes (\textcircled{S} i)) \wedge x_4 < \textcircled{S}(x \otimes \textcircled{S} i).$$

On considère $B'(x_0, x_1, x_2, x_3) := B(x_0, x_1, x_2, x_3) \wedge \forall x_4 < x_0\ \neg B(x_4, x_1, x_2, x_3)$. Cette dernière formule représente aussi β mais aussi que x_0 sera un entier standard. □

On reprend la preuve du lemme 13.3.

Preuve. Soient f, g, h telles que

- $f(0, x_1, \dots, x_p) = g(x_1, \dots, x_p)$
- $f(x_0 + 1, x_1, \dots, x_p) = h(x_0, f(x_0, \dots, x_p), x_1, \dots, x_p)$

Soient G, H représentant g et h. On a dans $\mathbb{N}$: $y = f(x_0, \dots, x_p)$ ssi il existe $z_0, \dots, z_{x_0}$ tel que

- $z_0 = g(x_1, \dots, x_p)$
- $z_1 = h(0, z_0, x_1, \dots, x_p)$

- $z_2 = h(1, z_1, x_1, \dots, x_p)$
- $\vdots$
- $z_{x_0} = h(x_0 - 1, z_{x_0-1}, x_1, \dots, x_p)$
- $y = z_{x_0}$

ssi

$$\exists a\, \exists b \Big[(\exists z_0\, B'(z_0, \textcircled{0}, a, b) \wedge G(z_0, x_1, \dots, x_p)) \wedge \forall i < x_0\, \exists z\, \exists z' \begin{pmatrix} B'(z, i, a, b) \\ \wedge \quad B'(z', \textcircled{S} i, a, b) \\ \wedge \quad H(z', i, z, x_1, \dots, x_p) \end{pmatrix} \wedge B'(y, x_0, a, b) \Big]$$

est vraie. Montrons que F représente f.

Soit $\mathscr{M} \models \mathscr{P}_0$, et $n_0, \dots, n_p$ des entiers et $c \in |\mathscr{M}|$.

- Si c interprète $\overline{f(n_0, \dots, n_p)}$ alors en choisissant a et b avec le lemme précédent sur la fonction β, on a bien $F(c, n_0, \dots, n_p)$.
- Réciproquement, si $\mathscr{M} \models F(d, \overline{n_0}, \dots, \overline{n_p})$ alors il existe a, b, z_0 tels que $B'(z_0, \textcircled{0}, a, b)$ et $G(z_0, n_1, \dots, n_p)$, et donc $z_0 = g(n_1, \dots, n_p)$. Et, pour tout $i \leq n_0$, il existe r_i et s_i tels que

  $$B'(r_i, i, a, b) \wedge B'(s_i, i + 1, a, b) \wedge H(s_i, i, r_i, n_1, \dots, n_p)$$

  donc $r_i = f(i, n_1, \dots, n_p)$ grâce aux propriétés de B' et car r_i est un entier naturel, et donc par récurrence $d = f(n_0, \dots, n_p)$.

□

Ceci conclut la preuve du théorème 13.4.

Maintenant que l'on a transformé les fonctions en formules, on va faire l'opposé. Notre but est de montrer le théorème suivant : soit

T une théorie consistante contenant $\mathscr{P}_0$ alors T est indécidable. La « partie technique » de l'indécidabilité de Gödel est la preuve par diagonalisation.

13.4 Indécidabilité des théories consistantes contenant $\mathscr{P}_0$.

On va coder :

1. les suites d'entiers ;
2. les termes ;
3. les formules ;
4. les preuves.

Lemme 13.5 (Récursion). Soient $p, n \in \mathbb{N}$ et

- $k_1, \dots, k_n : \mathbb{N} \to \mathbb{N}$ telles que $\forall y, \forall i,\ k_i(y) < y$;
- $g : \mathbb{N}^p \to \mathbb{N}$;
- $h : \mathbb{N}^{p+n+1} \to \mathbb{N}$

des fonctions récursives primitives (*resp.* récursives). Alors, la fonction $f : \mathbb{N}^{p+1} \to \mathbb{N}$ définie de la façon suivante est récursive primitive (*resp.* récursive primitive) :

$$f(0, x_1, \dots, x_p) := g(x_1, \dots, x_p)$$

et $f(y, x_1, \dots, x_p) := h(y, f(k_1(y), x_1, \dots, x_p), \dots, f(k_n(y), x_1, \dots, x_p), x_1, \dots, x_p)$. □

Lemme 13.6 (Définition par cas). Soient $P_1, \dots, P_n$ des ensembles récursifs primitifs (*resp.* récursifs) disjoints de $\mathbb{N}^m$ et $f_1, \dots, f_{n+1}$ des fonctions récursives primitives (*resp.* récursives) $\mathbb{N}^m \to \mathbb{N}$

alors la fonction suivante est récursive primitive (*resp.* récursive) :

$$f(x_1,\dots,x_m) := \begin{cases} f_1(x_1,\dots,x_m) & \text{si } P_1(x_1,\dots,x_m) \\ f_2(x_1,\dots,x_m) & \text{si } P_2(x_1,\dots,x_m) \\ \vdots & \vdots \\ f_n(x_1,\dots,x_m) & \text{si } P_n(x_1,\dots,x_m) \\ f_{n+1}(x_1,\dots,x_m) & \text{sinon} \end{cases}$$

□

Lemme 13.7 (Définition par cas et récursion)**.** Soient $p, n, m \in \mathbb{N}$, et

- $g : \mathbb{N}^p \to \mathbb{N}$
- $k_1, \dots, k_m : \mathbb{N} \to \mathbb{N}$
- $f_1, \dots, f_n : \mathbb{N}^{m+p+1} \to \mathbb{N}$
- $f_{n+1} : \mathbb{N}^p \to \mathbb{N}$

des fonctions récursives primitives (*resp.* récursives) et $P_1, \dots, P_n$ des ensembles disjoints de $\mathbb{N}^p$ récursifs primitifs (*resp.* récursifs) alors la fonction suivante est récursive primitive :

$$f(0, x_1, \dots, x_p) := g(x_1, \dots, x_p)$$

et

$$f(y, x_1, \dots, x_p) := \begin{cases} f_1(y, f(k_1(y), x_1, \dots, x_p), \dots, f(k_m(y), x_1, \dots, x_p), x_1, \dots, x_p) & \text{si } P_1(x_1, \dots, x_p) \\ f_2(y, f(k_1(y), x_1, \dots, x_p), \dots, f(k_m(y), x_1, \dots, x_p), x_1, \dots, x_p) & \text{si } P_2(x_1, \dots, x_p) \\ \vdots \\ f_n(y, f(k_1(y), x_1, \dots, x_p), \dots, f(k_m(y), x_1, \dots, x_p), x_1, \dots, x_p) & \text{si } P_n(x_1, \dots, x_p) \\ f_{n+1}(x_1, \dots, x_p) \end{cases}$$

□

13.4.1 Codage des suites d'entiers.

Proposition 13.1. Pour tout entier non nul p il existe des fonctions récursives primitives bijectives $\alpha_p : \mathbb{N}^p \to \mathbb{N}$ et $\beta_p^1, \dots, \beta_p^p$:

$\mathbb{N} \to \mathbb{N}$ telles que la réciproque de α_p est $(\beta_p^1, \dots, \beta_p^p)$ et, de plus, si $x > 1$ et $p \geq 2$ alors $\beta_p^i(x) < x$.

Preuve. L'idée est qu'on utilise la fonction de Cantor (ou l'énumération de Peano) :

$$\alpha_2(n, m) := \frac{(n+m)(n+m+1)}{2} + n$$

et on pose

$$\alpha_{p+1}(x_1, \dots, x_{p+1}) := \alpha_p(x_1, \dots, x_{p-1}, \alpha_2(x_p, x_{p+1})).$$

Ainsi,

$$\alpha_p(x_1, \dots, x_p) = \alpha_2(x_1, \alpha_2(x_2, \dots)).$$

□

13.4.2 Les termes.

On suppose que l'ensemble des variables est $\{x_i \mid i \in \mathbb{N}\}$.

Définition 13.6. Le nombre de Gödel d'un terme t sur $\mathscr{L}$, noté $\sharp t$, est défini par :

- $t = \textcircled{0}$ alors $\sharp t := \alpha_3(0, 0, 0)$;
- $t = x_n$ alors $\sharp t := \alpha_3(n+1, 0, 0)$;
- $t = \textcircled{S}\, t_1$ alors $\sharp t := \alpha_3(\sharp t_1, 0, 1)$;
- $t = t_1 \oplus t_2$ alors $\sharp t := \alpha_3(\sharp t_1, \sharp t_2, 2)$;
- $t = t_1 \otimes t_2$ alors $\sharp t := \alpha_3(\sharp t_1, \sharp t_2, 3)$.

Lemme 13.8. Le codage est injectif.

Preuve. Expliciter la fonction de décodage définie sur l'espace image. □

Lemme 13.9. L'ensemble Term $:= \{\sharp t \mid t \text{ est un terme de } \mathscr{L}_0\}$ est récursif primitif.

Preuve. Montrons que la fonction caractéristique T de Term est récursif primitif. On utilise le lemme de définition par cas et récursion donné précédemment :

- si $\beta_3^3(x) = 0$ et $\beta_3^2(x) = 0$ alors $T(x) = 1$ (x est le code de Ⓞ ou $x_{\beta_3^1(x)-1}$) ;
- si $\beta_3^3(x) = 1$ et $\beta_3^2(x) = 0$ alors $T(x) = T(\beta_3^1(x))$ (x est le code de Ⓢt) ;
- si $\beta_3^3(x) = 2$ alors $T(x) = T(\beta_3^1(x)) \cdot T(\beta_3^2(x))$ (x est le code de $t \oplus t$) ;
- si $\beta_3^3(x) = 3$ alors $T(x) = T(\beta_3^1(x)) \cdot T(\beta_3^2(x))$ (x est le code de $t \otimes t$) ;
- sinon, $T(x) = 0$.

□

13.4.3 Les formules.

Définition 13.7. On étend $\sharp\cdot$ aux formules :

- $\sharp(t_1 = t_2) := \alpha_3(\sharp t_1, \sharp t_2, 0)$
- $\sharp(\neg F) := \alpha_3(\sharp F, 0, 1)$
- $\sharp(F_1 \vee F_2) := \alpha_3(\sharp F_1, \sharp F_2, 2)$
- $\sharp(F_1 \wedge F_2) := \alpha_3(\sharp F_1, \sharp F_2, 3)$
- $\sharp(F_1 \to F_2) := \alpha_3(\sharp F_1, \sharp F_2, 4)$
- $\sharp(\forall x_k\, F) := \alpha_3(\sharp F, k, 5)$
- $\sharp(\exists x_k\, F) := \alpha_3(\sharp F, k, 6)$
- $\sharp\bot = \alpha_3(0, 0, 7)$.

Lemme 13.10. Le codage ci-dessus est injectif. □

Lemme 13.11. L'ensemble Form := $\{\sharp F \mid F \text{ formule de } \mathscr{L}_0\}$ est récursif primitif. □

13.4.4 Opérations sur les formules.

Lemme 13.12. Les ensembles suivants sont récursifs primitifs :

- $\theta_0 := \{(\sharp t, n) \mid t \text{ est un terme et } x_n \text{ n'a pas d'occurrence dans } t\}$
- $\theta_1 := \{(\sharp t, n) \mid t \text{ est un terme et } x_n \text{ a une occurrence dans } t\}$
- $\phi_0 := \{(\sharp F, n) \mid F \text{ est une formule et } x_n \text{ n'a pas d'occurrence dans } F\}$
- $\phi_1 := \{(\sharp F, n) \mid F \text{ est une formule et } x_n \text{ n'a pas d'occurrence libre dans } F\}$
- $\phi_2 := \{(\sharp F, n) \mid F \text{ est une formule et } x_n \text{ n'a pas d'occurrence liée dans } F\}$
- $\phi_3 := \{(\sharp F, n) \mid F \text{ est une formule et } x_n \text{ a une occurrence libre dans } F\}$
- $\phi_4 := \{(\sharp F, n) \mid F \text{ est une formule et } x_n \text{ a une occurrence liée dans } F\}$
- $\phi_5 := \{\sharp F \mid \ F \text{ est une formule close }\}$

Preuve. On montre le résultat pour θ_0 (le reste en exercice). On définit la fonction caractéristique de θ_0, notée $g_0(x, y)$, par (en utilisant le lemme de définition par cas et récursion) :

- si $\beta_3^3(x) = \beta_3^2(x) = 0$ et $\beta_3^1(x) - 1 \neq y$ alors $g_0(x, y) := 1$;
- si $\beta_3^2(x) = 1$ et $\beta_3^2(x) = 0$ alors $g_0(x, y) := g_0(\beta_3^2(x), y)$;
- si $\beta_3^3(x) = 2$ ou 3 alors $g_0(x, y) := g_0(\beta_3^1(x), y) \times g_0(\beta_3^2(x), y)$;
- sinon, $g_0(x, y) := 0$.

□

Lemme 13.13 (Substitutions). Il existe des fonctions récursives primitives Subst_t et Subst_f à trois variables telles que, si t t u sont des termes, et si G est une formule, alors pour tout entier n,

- $\text{Subst}_\text{t}(n, \sharp t, \sharp u) := \sharp(u[x_n := t])$
- $\text{Subst}_\text{f}(n, \sharp t, \sharp F) := \sharp(F[x_n := t])$.

Preuve. On définit Subst_t par cas/récursion. Pour (n, y, x), on a :

- si $\beta_3^3(x) = 0$ alors

- si $\beta_3^1(x) = n+1$ alors $\mathrm{Subst}_\mathrm{t}(n,y,x) := y$,
- sinon $\mathrm{Subst}_\mathrm{t}(n,y,x) := x$;

▷ si $\beta_3^3(x) = 1$ alors $\mathrm{Subst}_\mathrm{t}(n,y,x) := \alpha_3(\mathrm{Subst}_\mathrm{t}(n,y,\beta_3^1(x)),0,1)$;

▷ si $\beta_3^3(x) = 1$ alors
$\mathrm{Subst}_\mathrm{t}(n,y,x) := \alpha_3(\mathrm{Subst}_\mathrm{t}(n,y,\beta_3^1(x)),\mathrm{Subst}_\mathrm{t}(n,y,\beta_3^2(x)),\beta_3^3(x))$;

▷ sinon $\mathrm{Subst}_\mathrm{t}(n,y,x) := 0$.

Puis, on définit $\mathrm{Subst}_\mathrm{f}$ par :

- ▷ si $\beta_3^3(x) = 0$ alors $\mathrm{Subst}_\mathrm{f}(n,y,x) = \alpha_3(\mathrm{Subst}_\mathrm{t}(n,y,\beta_3^1(x))),\mathrm{Subst}_\mathrm{t}(n,y,\beta_3^1(x),0)$;
- ▷ si $\beta_3^3(x) = 1$ alors $\mathrm{Subst}_\mathrm{f}(n,y,x) = \alpha_3(\mathrm{Subst}_\mathrm{f}(n,y,\beta_3^1(x)),0,1)$;
- ▷ si $\beta_3^3(x) = 2,3,$ ou 4 alors $\mathrm{Subst}_\mathrm{f}(n,y,x) = \alpha_3(\mathrm{Subst}_\mathrm{f}(n,y,\beta_3^1(x)),\mathrm{Subst}_\mathrm{f}(n,y,\beta_3^2(x)),\beta_3^3(x))$;
- ▷ si $\beta_3^3(x) = 5$ ou 6 alors
  - si $\beta_3^2(x) = n$ et x_n est liée dans F donc $\mathrm{Subst}_\mathrm{f}(n,y,x) := x$;
  - sinon donc $\mathrm{Subst}_\mathrm{f}(n,y,x) := \alpha_3(\mathrm{Subst}_\mathrm{f}(n,y,\beta_3^1(x)),\beta_3^2(x),\beta_3^3(x))$;
- ▷ si $\beta_3^3(x) = 7$ alors $\mathrm{Subst}_\mathrm{f}(n,x,y) := x$;
- ▷ sinon, $\mathrm{Subst}_\mathrm{f}(n,x,y) := 0$.

□

13.4.5 Codage des preuves.

On code un contexte comme des suites finies, *i.e.* des listes, de formules (c'est plus facile que pour les ensembles).

Définition 13.8. On définit le codage par :

- ▷ $\sharp[\,] := 0$;
- ▷ $\sharp(F::\Gamma) := 1 + \alpha_2(\sharp\Gamma,\sharp F)$.

Lemme 13.14. Le décodage est unique. □

Lemme 13.15. La substitution d'une formule dans un contexte est récursif primitif. Tester si une variable est libre (*resp.* liée) dans un contexte est récursif primitif. □

13.4.6 Codage des preuves en déduction naturelle.

Remarque 13.6. Le contexte de la conclusion et des prémisses

est le même sauf pour

$$\frac{\Gamma \vdash A}{\Gamma, B \vdash A}\ \mathsf{aff} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \to_{\mathsf{i}} \qquad \frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \to_{\mathsf{i}}$$

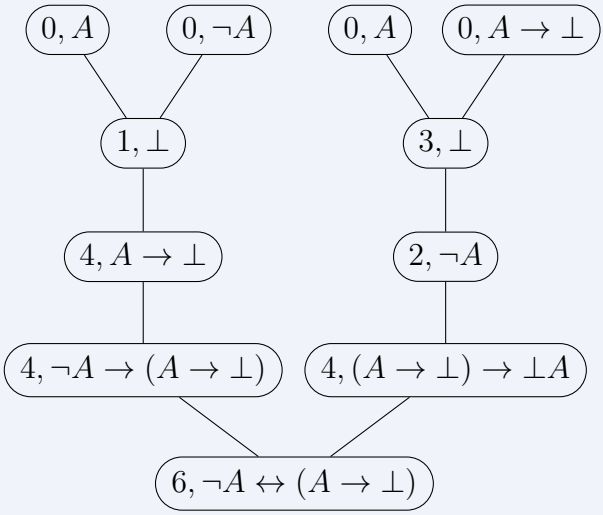
$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\ \bot_{\mathsf{c}} \qquad \frac{}{\Gamma \vdash A}\ \mathsf{ax}.$$

On peut toujours déterminer le contexte du haut à partir du bas donc donner le contexte de la racine suffit. Une preuve est donc finalement un contexte et un arbre de dérivation où les nœuds sont étiquetés par une formule et un numéro de règle.

Exemple 13.3. La preuve

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg A, A \vdash A}\ \mathsf{ax} \quad \dfrac{}{\neg A, A \vdash \neg A}\ \mathsf{ax}}{\neg A, A \vdash \bot}\ \neg_{\mathsf{e}}}{\neg A \vdash A \to \bot}\ \to_{\mathsf{i}}}{\vdash \neg A \to (A \to \bot)}\ \to_{\mathsf{i}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A \to \bot, A \vdash A}\ \mathsf{ax} \quad \dfrac{}{A \to \bot, A \vdash A \to \bot}\ \mathsf{ax}}{A \to \bot, A \vdash \bot}\ \to_{\mathsf{e}}}{A \to \bot \vdash \neg A}\ \neg_{\mathsf{i}}}{\vdash (A \to \bot) \to \neg A}\ \to_{\mathsf{i}}}{\vdash \neg A \leftrightarrow (A \to \bot)}\ \wedge_{\mathsf{i}}$$

peut être codée par l'arbre suivant avec le contexte `[]` à la racine :



Définition 13.9. On numérote

- ♯ax := 0
- $\sharp\neg_{\mathsf{e}} := 1$
- $\sharp\neg_{\mathsf{i}} := 2$
- $\sharp \to_{\mathsf{e}} := 3$
- $\sharp \to_{\mathsf{i}} := 4$
- $\sharp\wedge_{\mathsf{e}} := 5$
- $\sharp\wedge_{\mathsf{i}} := 6$
- *etc.*

Définition 13.10 (Nombre de Gödel des preuves).

- Si $D^\star$ est un arbre de preuve à un seul nœud étiqueté par la formule F et la règle n alors $\sharp D^\star := \alpha_3(n, \sharp F, 0)$.
- Si $D^\star$ est un arbre de preuve dont la racine est étiquetée par la formule F et la règle n à k prémisses avec les sous arbres $D_1^\star, \dots, D_k^\star$

$$\frac{D_1^\star \quad \cdots \quad D_k^\star}{F} \text{ règle } n$$

alors $\sharp D^\star := \alpha_3(n, \sharp F, \alpha_k(\sharp D_1^\star, \dots, \sharp D_k^\star) + 1)$.

On pose ensuite $\sharp D := \alpha_2(\sharp D^\star, \sharp\Gamma)$ pour une preuve D.

Lemme 13.16. C'est un code injectif.

Lemme 13.17. L'ensemble Preuve := $\{\sharp D \mid D \text{ est une preuve}\}$ est récursif primitif.

13.4.7 Théories (in)décidables.

Définition 13.11. Un ensemble A de formules est un ensemble d'*axiomes* de la théorie T si $A \vdash T$ et $T \vdash A$.

Définition 13.12. Une théorie T sur $\mathscr{L}_0$ a un ensemble d'axiomes Ax_T récursif si l'ensemble des numéros de formules de Ax_T est récursif.

Remarque 13.7. Si Ax_T est fini, alors il est récursif (exemple : $\mathscr{P}_0$).

Lemme 13.18. L'ensemble des axiomes de Peano $\mathscr{P}$ est récursif.

Preuve. Il suffit de montrer que l'ensemble des axiomes du schéma de récurrence est récursif. On définit

$$A_F \coloneqq \forall x_1 \cdots \forall x_n \Big(\big(F(0, x_1, \ldots, x_n) \wedge \forall x_0 \, (F(x_0, \ldots, x_n) \to F(\textcircled{S} x_0, x_1, \ldots, x_n)) \big) \to \forall x_0 \, F(x_0, \ldots, x_n) \Big).$$

Idée pour décider si N est le code d'une formule A_F :

1. décoder pour trouver n et F ;
2. calculer $\sharp A_F$ et vérifier si c'est N.

□

Proposition 13.2. Si une théorie T a un ensemble d'axiomes Ax_T alors l'ensemble

$$\mathrm{Dem}_T = \{ \, (\sharp D, \sharp F) \mid D \text{ est une preuve de } F \text{ dans } T \text{ avec } \mathrm{Ax}_T \, \}.$$

Preuve. L'idée de la preuve est la suivante :

1. décider x et y ;
2. vérifier que x est une preuve et y une formule ;
3. vérifier que D est une preuve de F ;
4. vérifier que le contexte final ne contient que des éléments de Ax_T.

□

Dans la suite, on prend $\mathscr{L} \supseteq \mathscr{L}_0$.

Définition 13.13. Une théorie est *décidable* si l'ensemble de ses théorèmes est récursif.

Remarque 13.8 (Rappel)**.** Une théorie est *consistante* si elle a un modèle.

Théorème 13.5. Soit T une théorie consistante contenant $\mathscr{P}_0$. Alors, T est indécidable.

Preuve. On suppose que T est décidable et on construit par diagonalisation une formule F telle que $T \vdash F$ et $T \vdash \neg F$. Soit

$$\theta := \{\, (m,n) \mid m = \sharp(F(n)) \text{ et } T \vdash F(\textcircled{n}) \,\}.$$

L'ensemble T est décidable donc θ aussi. On pose

$$B := \{\, n \in \mathbb{N} \mid (n,n) \notin \theta \,\},$$

qui est récursif.

D'après le théorème de représentation, il existe une formule $G(x)$ représentant B :

- $n \in B \implies \mathscr{P}_0 \vdash G(\textcircled{n})$ donc $T \vdash G(\textcircled{n})$;
- $n \notin B \implies \mathscr{P}_0 \vdash \neg G(\textcircled{n})$ donc $T \vdash \neg G(\textcircled{n})$.

Soit $a = \sharp(G(x))$. Est-ce que $a \in B$?

- On a $a \in B \iff (a,a) \notin \theta \iff T \nvdash G(\textcircled{a})$. Or, si $a \in B$ alors, par définition de G, on a $T \vdash G(\textcircled{a})$. ***Absurde !***
- On a $a \notin B \iff (a,a) \in \theta \iff T \vdash G(\textcircled{a})$. Or, si $a \notin B$ alors, par définition de G, on a $T \vdash \neg G(\textcircled{a})$. Donc T non consistante. ***Absurde !***

□

Exemple 13.4 (Application du théorème)**.** La théorie $T = \mathbf{Th}(\mathbb{N})$ est indécidable.

Exemple 13.5 (Quelques théories décidables).
- ▷ Les ordres denses sans extrémités (la théorie linéaire des rationnels) est une théorie décidable.
- ▷ Les corps réels clos (*théorème de Tarski*) est une théorie décidable.
- ▷ L'arithmétique de Presburger (la théorie linéaire des entiers) est une théorie décidable.
- ▷ Pour chaque p, les corps algébriquement clos de caractéristique p est une théorie décidable.

On peut donc répondre à l'Entscheidungsproblem, le problème de décision.

Théorème 13.6 (Church, indécidabilité du calcul des prédicats). Si $\mathscr{L} \supseteq \mathscr{L}_0$, l'ensemble T des théorèmes logiques sur $\mathscr{L}$ n'est pas récursif.

Preuve. Soit T_0 l'ensemble des théorèmes logiques sur $\mathscr{L}_0$. Soit G la conjonction des axiomes de $\mathscr{P}_0$. Pour toute formule F, on a $\mathscr{P}_0 \vdash F$ ssi $T_0 \vdash (G \to F)$. Donc, si T_0 est récursif alors $\mathscr{P}_0$ est décidable. Donc, si T est récursif, alors T_0 aussi. Donc $\mathscr{P}_0$ est décidable, ***absurde***. □

13.5 Théorèmes d'incomplétude de Gödel

Théorème 13.7 (Premier théorème d'incomplétude de Gödel). Soit T une théorie qui a un ensemble d'axiomes récursifs, et qui est consistante, et qui contient $\mathscr{P}_0$. Alors, T n'est pas axiome-complète.

Preuve. Une théorie qui a un ensemble d'axiomes récursifs et qui est complète, est décidable, ce qui est faux.

En effet, pour F une formule, comment déterminer (algorithmiquement) su $T \vdash F$? On énumère toutes les preuves jusqu'à en trouver une de F ou de $\neg F$. □

Corollaire 13.1. La théorie $\mathscr{P}$ n'est pas complète.

Question.

Peut-on exhiber une formule F telle que $T \nvdash F$ et $T \nvdash \neg F$?

On va construire F qui « dit » que T est consistante.

Définition 13.14. On pose :

- ▷ $\mathrm{Dem}_T := \{ (\sharp D, \sharp F) \mid D \text{ preuve de } F \text{ dans } T \}$;
- ▷ $\mathrm{Dem}_{\mathscr{P}_0} := \{ (\sharp D, \sharp F) \mid D \text{ preuve de } F \text{ dans } \mathscr{P}_0 \}$.

Proposition 13.3.

- ▷ Ces ensembles sont récursifs donc représentés par F_T et $F_{\mathscr{P}_0}$.
- ▷ La fonction $\mathrm{neg} : \mathbb{N} \to \mathbb{N}, \sharp F \mapsto \sharp(\neg F) = \alpha_3(\sharp F, 0, 1)$ est récursive et représentée par $F_{\mathrm{neg}}(x_0, x_1)$:

$$\forall n \in \mathbb{N}, \quad \mathscr{P}_0 \vdash \forall x \, (F_{\mathrm{neg}}(x, \underline{n}) \leftrightarrow x = \underline{\mathrm{neg}(n)}).$$

□

Définition 13.15. On pose

$$\mathrm{Coh}(T) := \neg \exists x_0 \cdots \exists x_3 \Big(F_T(x_0, x_2) \wedge F_T(x_1, x_3) \wedge F_{\mathrm{neg}}(x_2, x_3) \Big).$$

Remarque 13.9. La fonction Coh n'est pas complètement définie, car elle dépend du choix de F_T et de F_{neg}.

Proposition 13.4. La théorie T est consistante ssi $\mathbb{N} \models \mathrm{Coh}(T)$.

Remarque 13.10. On pourrait avoir $\mathscr{M} \models T$, avec T consistante et $\mathscr{M} \models \neg\mathrm{Coh}(T)$. En effet, il suffit que x_0, x_1, x_2, x_3 ne soient pas

des entiers standards.

Théorème 13.8 (Second théorème d'incomplétude de Gödel)**.** Soit T une théorie consistante, axiome-récursive, et contenant $\mathscr{P}_0$. Alors, $T \nvdash \mathrm{Coh}(T)$.

Remarque 13.11. Si $\mathbb{N} \models T$, ce théorème implique le 1er théorème d'incomplétude car $\mathbb{N} \not\models \neg\mathrm{Coh}(T)$, donc $T \not\models \neg\mathrm{Coh}(T)$ et donc T incomplète.

Dans le cas général, ce n'est pas vrai : $T \cup \{\neg\mathrm{Coh}(T)\}$ est une théorie consistante. Par exemple, $\mathscr{P} \cup \{\neg\mathrm{Coh}(\mathscr{P})\}$ est consistante mais $\mathbb{N}$ n'en est pas un modèle.

Définition 13.16. L'ensemble Σ est le plus petit ensemble de formules contenant $\mathscr{L}_0$ qui

- ▷ contient les formules sans quantificateurs ;
- ▷ est clos par $\wedge, \vee, \exists$;
- ▷ est clos par quantification universelle bornée, *i.e.* si $F \in \Sigma$ alors
$$\Big(\forall v_0\, (v_0 < v_1) \to F\Big) \in \Sigma.$$

Exemple 13.6. Les relations « $n \mid m$ » et « m est premier » peuvent s'exprimer avec des formules de Σ.

Lemme 13.19 (Représentation (bis))**.** Toute fonction récursive totale est représentable par une formule de Σ.

Preuve. Les formules que l'on construit dans le lemme 13.3 sont des formules de Σ. □

Lemme 13.20. Il existe des formules F_T et $F_{\mathscr{P}_0}$ qui satisfont :

1. $\vdash \forall v_0\, \forall v_1\, F_{\mathscr{P}_0}(v_0, v_1) \to F_T(v_0, v_1)$;
2. F_T et $F_{\mathscr{P}_0}$ sont dans Σ ;
3. si F est une formule close de Σ alors

$$\mathscr{P} \vdash (F \to \exists x\, F_{\mathscr{P}_0}(x_1, \sharp F)).$$

Preuve. 1. Il suffit de remplacer F_T par par $F_T \vee F_{\mathscr{P}_0}$.
2. C'est une conséquence du lemme précédent.
3. On va le montrer pour une théorie $\mathscr{P}_1$ contenant $\mathscr{P}_0$ et conséquence de $\mathscr{P}$ mais, *a priori*, plus faible que $\mathscr{P}$. Puis, on l'admet pour $\mathscr{P}$, et on admet que $\mathscr{P} \vdash \mathscr{P}_1$. On a le montrer par la proposition suivante.

□

Proposition 13.5. Soit F une formule close sur $\mathscr{L}_0$ dans Σ. Alors,

$$\mathbb{N} \models F \to \exists x_1\, F_{\mathscr{P}_0}(x_1, \sharp F).$$

Preuve. ▷ Si F est fausse, c'est montré.
▷ Si $\mathbb{N} \models F$, il faut montrer que F a une preuve dans $\mathscr{P}_0$, *i.e.* que tout modèle $\mathscr{M} \models \mathscr{P}_0$, on a $\mathscr{M} \models F$ *i.e.* que dans tout extension finale $\mathscr{M}$ de $\mathbb{N}$ alors $\mathscr{M} \models F$, pour cela il suffit de montrer le lemme suivant.

□

Lemme 13.21. Soient $\mathcal{N}$ une $\mathscr{L}_0$-structure et $\mathscr{M}$ une extension finale de $\mathcal{N}$. Soient $F(x_1, \dots, x_p) \in \Sigma$ et $a_1, \dots, a_p \in \mathcal{N}$. Alors, $\mathcal{N} \models F(a_1, \dots, a_p)$ implique $\mathscr{M} \models F(a_1, \dots, a_p)$.

Preuve. Par induction sur $F(x_1, \dots, x_p) \in \Sigma$. □

On termine la preuve du point 3.

Preuve. On pose

$$\mathscr{P}_1 := \mathscr{P}_0 \cup \{\, F \to \exists x\, F_F(x_1, \sharp F) \mid F \text{ formule close de } \Sigma \,\}.$$

On a montré que $\mathbb{N} \models \mathscr{P}$. On admet que $\mathscr{P} \vdash \mathscr{P}_1$ donc $T \vdash \mathscr{P}_1$. □

Lemme 13.22 (Cœur du 2nd théorème d'incomplétude)**.** Soit T une théorie consistante, axiome-récursive, et contenant $\mathscr{P}_0$. Alors, $T \nvdash \mathrm{Coh}(T)$.

Preuve. ▷ Soit $g : \mathbb{N} \to \mathbb{N}$ définie par $n = \sharp F(x_0) \mapsto \sharp F(\textcircled{n}) = \sharp F(\textcircled{\sharp F(x_0)})$. C'est *la formule appliquée à elle-même*. La fonction g est primitive récursive, donc représentée par une formule $G(x, y)$ telle que

$$\forall n \in \mathbb{N}, \quad \mathscr{P}_0 \vdash \forall x\, G(x, \textcircled{n}) \leftrightarrow x = \textcircled{g(n)}.$$

- ▷ On considère la formule « il existe une preuve de x_0 appliquée à elle-même » :

$$\varepsilon(x_0) := \exists x_1\, \exists x_2\, F_T(x_1, x_2) \wedge G(x_2, x_0).$$

- ▷ On pose $a := \sharp(\neg\varepsilon(x_0))$, « il n'existe pas de preuve de x_0 appliquée à elle-même ».
- ▷ On pose $b := g(a) = \sharp(\neg\varepsilon(\textcircled{@}))$, « il n'existe pas du preuve du fait qu'il n'existe pas de preuve de x_0 appliquée à elle-même ».
- ▷ Dans $\mathscr{P}_0$, on a $\forall x_2\, G(x_2, \textcircled{@}) \leftrightarrow x_2 = \textcircled{b}$.
- ▷ Par définition, $\varepsilon(\textcircled{@})$ est $\exists x_1\, \exists x_2\, F_T(x_1, x_2) \wedge G(x_0, \textcircled{@})$. « Il existe une preuve du fait qu'il n'existe pas de preuve de nous-même ». Dans $\mathscr{P}_0$, $\varepsilon(\textcircled{@})$ est équivalent à $\exists x_1 F_T(x_1, \textcircled{b}) \wedge G(\textcircled{b})$, ce qui est équivalent à $\exists x_1\, F_T(x_1, \textcircled{b})$ car $b = g(a)$ $(\star)$. Ainsi, on a « $\varepsilon(\textcircled{@})$ ssi il y a une preuve de $\neg\varepsilon(\textcircled{@})$ »

Voici le paradoxe :

- ▷ Prouvons que $T \vdash \mathrm{Coh}(T) \to \neg\varepsilon(@)$. Il suffit de montrer que $\mathscr{P}_1 \vdash \varepsilon(@) \to \neg\mathrm{Coh}(T)$. Soit $T_1 := \mathscr{P}_1 \cup \{\varepsilon(@)\}$. Alors $T_1 \vdash \exists v_1\, F_T(v_1, \textcircled{b})$ et $b = \sharp(\neg\varepsilon(@))$ On a donc une preuve de $\varepsilon(@)$ et une preuve de $\neg\varepsilon(@)$, donc de $\neg\mathrm{Coh}(T)$.
- ▷ On va montrer que $T \vdash \neg\varepsilon(@)$ mène à un paradoxe. Si c'est vrai, soit C le numéro d'une preuve de $\neg\varepsilon(@)$ dans T. Alors, $\mathscr{P}_0 \vdash F_T(\textcircled{c}, \textcircled{b})$. D'où, avec $(\star)$, $\mathscr{P}_0 \vdash \varepsilon(@)$ impossible car T consistante. Donc $T \nvdash \neg\varepsilon(@)$ et donc $T \nvdash \mathrm{Coh}(T)$.

□

14. La théorie des ensembles.

On se place dans la logique du 1er ordre avec $\mathscr{L} = \{\in, =\}$. On se place dans un univers $\mathcal{U}$ non vide, le modèle, dont les éléments sont appelés des *ensembles*.

Il faudra faire la différence entre les ensembles « naïfs » (les ensembles habituels), et les ensembles « formels » (les éléments de $\mathcal{U}$).

On a le paradoxe de Russel. On peut l'écrire

> « On a un barbier qui rase tous les hommes qui ne se rasent pas eux-mêmes. Qui rase le barbier ? ».

Si $\mathcal{U}$ est l'ensemble de tous les ensembles, alors

$$a := \{\, x \in \mathcal{U} \mid x \notin x \,\}$$

vérifie $a \in a \iff a \notin a$, ***paradoxe***. Pour éviter ce paradoxe, on choisit donc de ne pas faire $\mathcal{U}$ un ensemble.

14.1 Les axiomes de la théorie de Zermelo-Fraenkel.

ZF 1. *Axiome d'extensionnalité* : deux ensembles sont égaux ssi ils ont les mêmes éléments

$$\forall x\, \forall y\, \Big(\forall z\, (z \in x \leftrightarrow z \in y) \leftrightarrow x = y\Big).$$

- *Axiome de la paire*[1] : il existe une paire $\{x, y\}$ pour tout élément x et y

$$\forall x\, \forall y\, \exists z\, \forall t\Big(t \in z \leftrightarrow (t = x \vee t = y)\Big).$$

1. On verra plus tard que cet axiome est une conséquence des autres (de ZF 3 et ZF 4).

[*continué plus tard...*]

Remarque 14.1. Cela nous donne l'existence du *singleton* $\{x\}$ si x est un ensemble. En effet, il suffit de faire la paire $\{x, x\}$ avec l'Axiome de la paire.

Définition 14.1. Si a et b sont des ensembles, alors (a, b) est l'ensemble $\{\{a\}, \{a, b\}\}$. Ainsi, (a, a) est l'ensemble $\{\{a\}\}$.

Lemme 14.1. Pour tous ensembles a, b, a', b', on a $(a, b) = (a', b')$ ssi $a = a'$ et $b = b'$.

Preuve. En exercice. □

Définition 14.2. On peut construire des 3-uplets (a_1, a_2, a_3) avec $(a_1, (a_2, a_3))$, et ainsi de suite pour les n-uplets.

Notation. On utilise les raccourcis

- $t = \{a\}$ pour $\forall x\,(x \in t \leftrightarrow x = a)$;
- $t = \{a, b\}$ pour $\forall x\,(x \in t \leftrightarrow (x = a \vee x = b))$;
- $t \subseteq a$ pour $\forall x\,(z \in t \to z \in a)$.

ZF 3. *Axiome des parties* : l'ensemble des parties $\wp(a)$ existe pour tout ensemble a

$$\forall a\, \exists b\, \forall t\, (t \in b \leftrightarrow t \subseteq a).$$

ZF 2. *Axiome de la réunion* : l'ensemble $y = \bigcup_{z \in x} z$ existe

$$\forall x\, \exists y\, \forall t (t \in y \leftrightarrow \exists z (t \in z \wedge z \in x)).$$

Remarque 14.2. Comment faire $a \cup b$? La paire $x = \{a, b\}$ existe par l'Axiome de la paire, et $\bigcup_{z \in x} z = a \cup b$ est un ensemble

par ZF 2.

ZF 4'. *Schéma de compréhension* : pour toute formule $\varphi(y, v_1, \dots, v_n)$, on a l'ensemble $x = \{ y \in v_{n+1} \mid \varphi(y, v_1, \dots, v_n) \}$

$$\forall v_1 \ \dots \ \forall v_n \ \exists x \, \forall y \left(y \in x \leftrightarrow (y \in v_{n+1} \wedge \varphi(y, v_1, \dots, v_n)) \right).$$

Remarque 14.3. Peut-on faire le paradoxe de Russel ? On ne peut pas faire $a := \{z \in \mathcal{U} \mid z \notin z\}$ car $\mathcal{U}$ n'est pas un ensemble ! Et, on ne peut pas avoir de paradoxe avec $b := \{z \in E \mid z \notin z\}$, car on a l'ajout de la condition $b \in E$.

Définition 14.3. Une *relation fonctionnelle* en w_0 est une formule $\varphi(w_1, w_2, a_1, \dots, w_n)$ *à paramètres* (où les a_i sont dans $\mathcal{U}$) telle que

$$\mathcal{U} \models \forall w_0 \, \forall w_1 \, \forall w_2 \left(\varphi(w_0, w_1, a_1, \dots, a_n) \wedge \varphi(w_0, w_2, a_1, \dots, w_n) \to w_1 = w_2 \right).$$

En termes naïfs, c'est une fonction partielle. On garde le terme *fonction* quand le domaine et la collection d'arrivée sont des *ensembles*, autrement dit, des éléments de $\mathcal{U}$.

ZF 4. *Schéma de substitution/de remplacement* : « la collection des images par une relation fonctionnelle des éléments d'un ensemble est aussi un ensemble ». Pour tout n-uplet $\bar{a}$, si la formule à paramètres $\varphi(w_0, w_1, \bar{a})$ définit une relation fonctionnelle $f_{\bar{a}}$ en w_0 et si a_0 est un ensemble alors la collection des images par $f_{\bar{a}}$ des éléments de a_0 est un ensemble nommé a_{n+1}

$$\forall a_0 \ \cdots \ \forall a_n$$
$$\left(\forall w_0 \, \forall w_1 \, \forall w_2 (\varphi(w_0, w_1, a_1, \dots, a_n) \wedge \varphi(w_0, w_2, a_1, \dots, a_n)) \to w_1 = w_2 \right)$$
$$\downarrow$$
$$\exists a_{n+1} \, \forall a_{n+2} (a_{n+2} \in a_{n+1} \leftrightarrow \exists w_0 \, w_0 \in a_0 \wedge \varphi(w_0, a_{n+2}, v_1, \dots, v_n)).$$

Théorème 14.1. Si ZF 1, ZF 2, ZF 3 et ZF 4 sont vrais dans $\mathcal{U}$, il existe (dans $\mathcal{U}$) un et un seul ensemble sans élément, que l'on notera $\emptyset$.

Preuve.
- ▷ *Unicité* par ZF 1.
- ▷ *Existence.* On procède par compréhension : l'univers $\mathcal{U}$ est non vide, donc a un élément x. On considère la formule $\varphi(w_0, w_1) := \bot$ qui est une relation fonctionnelle. Par ZF 4 (avec la formule φ et l'ensemble $a_0 := x$) un ensemble a_{n+1} qui est vide.

□

Proposition 14.1. Si ZF 1, ZF 2, ZF 3 et ZF 4 sont vrais dans $\mathcal{U}$, alors l'Axiome de la paire est vrai dans $\mathcal{U}$.

Preuve. On a $\emptyset$ dans $\mathcal{U}$ et également $\wp(\emptyset) = \{\emptyset\}$ et $\wp(\wp(\emptyset)) = \{\emptyset, \{\emptyset\}\}$ par ZF 3.

Étant donné deux ensemble a et b, on veut montrer que $\{a, b\}$ est un ensemble avec ZF 4

$$\varphi(w_0, w_1, a, b) := (w_0 = \emptyset \wedge w_1 = a) \vee (w_0 = \{\emptyset\} \wedge w_1 = b),$$

où

- ▷ $w_0 = \emptyset$ est un raccourci pour $\forall z\ (z \notin w_0)$;
- ▷ $w_0 = \{\emptyset\}$ est un raccourci pour $\forall z\ (z \in w_0 \leftrightarrow (\forall t\ t \notin z))$.

Ces notations sont compatibles avec celles données précédemment.

Comme φ est bien une relation fonctionnelle et $\{a, b\}$ est l'image de $\{\emptyset, \{\emptyset\}\}$. □

Proposition 14.2. Si ZF 1, ZF 2, ZF 3 et ZF 4 sont vrais dans $\mathcal{U}$, alors ZF 4' est vrai dans $\mathcal{U}$.

Preuve. On a la formule $\varphi(y, v_1, \dots, v_n)$ et on veut montrer que

$$\mathcal{U} \models \forall v_1 \ \cdots \ \forall v_{n+1} \ \exists x \ \forall y \ \Big(y \in x \leftrightarrow (y \in v_{n+1} \wedge \varphi(y, v_1, v_n)) \Big).$$

On considère la formule $\psi(w_0, w_1, \bar{v}) := w_0 = w_1 \wedge \varphi(w_0, \bar{v})$, qui est bien une relation fonctionnelle en w_0. La collection

$$\{\, y \in v_{n+1} \mid \varphi(y, v_1, \dots, v_n) \,\}$$

est l'image de v_{n+1} par ψ par ZF 4. □

Remarque 14.4. La réciproque du théorème précédent est fausse ! Les axiomes ZF 4 et ZF 4' ne sont pas équivalents. On le verra en TD (probablement).

Proposition 14.3. Le produit ensembliste de deux ensembles est un ensemble.

Preuve. Soient v_1 et v_2 deux ensembles. On considère

$$X := v_1 \times v_2 = \{\, (x, y) \mid x \in v_1 \text{ et } y \in v_2 \,\} \text{ (en naïf)} .$$

La notation (x, y) correspond à l'ensemble $\{\{x\}, \{x, y\}\} \in \wp(\wp(v_1 \cup v_2))$.

On applique ZF 4' dans l'ensemble ambiant $\wp(\wp(v_1 \cup v_2))$, on définit le produit comme la compréhension à l'aide de la formule

$$\varphi(z, v_1, v_2) := \exists x \ \exists y \ \Big(z = \{\{x\}, \{x, y\}\} \wedge x \in v_1 \wedge y \in v_2 \Big).$$

C'est bien un élément de $\mathcal{U}$. □

Définition 14.4. Une *fonction* (sous-entendu *totale*) d'un ensemble a dans un ensemble b est un sous-ensemble de $a \times b$ qui

vérifie la propriété

$$\varphi(f,a,b) := \left(\begin{array}{c} f \subseteq a \times b \\ \wedge \\ \forall x\, \forall y\, \forall y'\, (x,y) \in f \wedge (x,y') \in f \to y = y' \\ \wedge \\ \forall x\, x \in a \to \exists y\, y \in b \wedge (x,y) \in f \end{array}\right).$$

On identifie ainsi f et son graphe.

Une *fonction partielle* d'un ensemble a dans un ensemble b est un sous-ensemble de $a \times b$ qui vérifie la propriété

$$\varphi(f,a,b) := \left(\begin{array}{c} f \subseteq a \times b \\ \wedge \\ \forall x\, \forall y\, \forall y'\, (x,y) \in f \wedge (x,y') \in f \to y = y' \end{array}\right).$$

On note b^a la collection des fonctions partielles de a dans b.

Proposition 14.4. La collection b^a est un ensemble, *i.e.* si a et b sont dans $\mathcal{U}$ alors b^a aussi.

Preuve. En exercice. □

Remarque 14.5 (Réunion indexée)**.** Soit a une famille d'ensemble indexée par l'ensemble I, *i.e.* a est une fonction de domaine I. Si $i \in I$, on note a_i pour $a(i)$.

Proposition 14.5. Si I est un ensemble et a est une fonction de domaine I, alors $\bigcup_{i\in I} a_i$ est un ensemble. Autrement dit, si dans $\mathcal{U}$, ZF 1, ZF 2, ZF 3, ZF 4 sont vraies, et que I et a sont dans $\mathcal{U}$, et a est une fonction, alors la collection définie naïvement par $\bigcup_{i\in I} a_i$ appartient à $\mathcal{U}$.

Preuve. On pose $b := \{a_i \mid i \in I\}$. C'est bien un ensemble car b

est l'ensemble des images des éléments de I par a. On peut écrire a comme relation fonctionnelle :

$$\varphi(w_0, w_1, a) := (w_0, w_1) \in a.$$

On a donc que b est un ensemble avec ZF 4.

Et, $\bigcup_{i\in I} a_i = \bigcup_{z\in b} z$ donc on conclut par ZF 2. □

Proposition 14.6 (Propriété d'intersection)**.** Si I est un ensemble non vide et a est une fonction de domaine I alors $\bigcap_{i\in I} a_i$ est un ensemble.

Preuve. On pose $c := \bigcup_{i\in I} a_i$ qui est un ensemble par ZF 2. On considère

$$\varphi(x, a, I) := \forall i\, i \in I \to x \in a_i.$$

Par compréhension (ZF 4') on construit l'ensemble

$$\bigcap_{i\in I} a_i := \{x \in c \mid \varphi(x, a, I)\}.$$

□

Proposition 14.7. Si I est un ensemble et a une fonction de domaine i alors $\prod_{i\in I} a_i$ est un ensemble.

Preuve. La collection $\prod_{i\in I} a_i$ est l'ensemble des fonctions de I dans $\bigcup_{i\in I} a_i$ telles que $f(i) \in a_i$ pour tout i. □

ZF 5 *Axiome de l'infini* : il existe un ensemble ayant une infinité d'élément

$$\exists x\, (\emptyset \in x \wedge \forall y\, (y \in x \to y \cup \{y\} \in x)).$$

On encode les entiers avec des ensembles :

- $0 \rightsquigarrow \emptyset$
- $1 \rightsquigarrow \{\emptyset\}$

- $2 \rightsquigarrow \{\emptyset, \{\emptyset\}\}$
- $\vdots$
- $n+1 \rightsquigarrow n \cup \{n\}$
- $\vdots$

Ainsi, on a bien $n = \{0, 1, \ldots, n-1\}$.

Remarque 14.6. Si on retire $\emptyset \in x$, on peut avoir $x = \emptyset$ qui satisfait la version modifiée de ZF 5.

Cependant, sans retirer $\emptyset \in x$, on peut quand même avoir un ensemble fini s'il existe un ensemble fini y tel que $y \in y$. Ceci est impossible avec l'axiome de bonne fondation.

Remarque 14.7. Les français sont les seuls à considérer que l'axiome de bonne fondation ne fait pas partie de la théorie de ZF.

14.2 Ordinaux et induction transfinie.

Théorème 14.2 (Cantor).
1. Soient A et B deux ensembles et supposons qu'il existe des injections $A \to B$ et $B \to A$ alors il existe une bijection $A \to B$.
2. Il n'existe pas de surjection de $A \to \wp(A)$.

Preuve. En TD. □

Définition 14.5. Deux ensembles sont *équipotents* s'il existe une bijection entre-eux.

Définition 14.6. Soit A un ensemble. Un *ordre* (*partiel*, *strict*) sur A est une relation binaire $<$ (donnée par un sous-ensemble de $A \times A$) telle que

1. *transitivité* : $\forall x\, \forall y\, \forall z\, x < y \to y < z \to x < z$;
2. *anti-réflexif* : $\forall x, x \not< x$.

Notation. On note $x \leq y$ pour $x < y$ ou $x = y$.

Exemple 14.1. L'ordre $\subsetneq$ sur $\wp(\mathbb{N})$ est partiel. Les ordres $<_{\mathbb{N}}$, $<_{\mathbb{R}}$, $<_{\mathbb{Z}}$ sur $\mathbb{N}, \mathbb{R}, \mathbb{Z}$ sont totaux.

Définition 14.7. Soit $(A, <)$ ordonné. Soient $a, a' \in A$ et $B \subseteq A$. On dit que

- a est un plus petit élément de B si $a \in B$ et pour tout $b \in B$ et si $b \neq a$ alors $b > a$;
- a est un élément minimal de B si $a \in B$ et pour tout $b \in B$, $b \not< a$;
- a est un minorant de B si pour tout $b \in B$, $a \leq b$;
- de la même manière, on définit plus grand élément, élément maximal, majorant ;
- a est une borne inférieure de B si a est un plus grand élément de l'exemple des minorants de B ;
- a et a' sont incomparables si $a \neq a'$ et $a \not< a'$ et $a' \not< a$;
- un ordre est bien fondé si toute partie non vide de A a un élément minimal ;
- un bon ordre est un ordre total bien fondé.

Proposition 14.8. Un ordre total est bien fondé ssi il n'existe pas de suite infinie décroissante.

Preuve. En exercice. □

Exemple 14.2.
- L'ordre $<$ sur $\mathbb{N}$ est bien fondé.
- L'ordre $<$ sur $\mathbb{Z}$ n'est pas bien fondé.
- L'ordre $\subsetneq$ sur $\wp_{\text{finies}}(\mathbb{N})$ est bien fondé.
- L'ordre $\subsetneq$ sur $\wp(\mathbb{N})$ n'est pas bien fondé.

Définition 14.8.
- ▷ Deux ensembles ordonnés sont *isomorphes* s'il existe une bijection préservant l'ordre de l'un vers l'autre. On note $A \simeq B$.
- ▷ Soit X totalement ordonné. Un sous-ensemble $J \subseteq X$ est un *segment initial* si pour tous $a, b \in X$ avec $a < b$ alors $b \in J$ implique $a \in J$.
- ▷ Un ensemble X est *transitif* si pour tout $x \in X$ et $y \in x$ alors $y \in X$.
- ▷ Un ensemble X est un *ordinal* s'il est transitif et que $\in$ défini un bon ordre sur X.
- ▷ On note $\mathbb{O}$ la *classe des ordinaux*, et on note indifféremment les relations $\in$ et $<$.

Exemple 14.3. Les entiers de Von Neumann sont des ordinaux.

Proposition 14.9. Soient α et β des ordinaux. On a les propriétés suivantes :

1. $\emptyset$ est un ordinal ;
2. si $\alpha \neq \emptyset$ alors $\emptyset \in \alpha$;
3. $\alpha \notin \alpha$;
4. si $x \in \alpha$ alors $x = \{y \in \alpha \mid y < x\}$.
5. si $x \in \alpha$ alors x est un ordinal (on a l'abus de notation $x \in \mathbb{O}$) ;
6. $\beta \leq \alpha$ ssi $\beta = \alpha$ ou $\beta \in \alpha$;
7. $x = \alpha \cup \{\alpha\}$ est un ordinal noté $\alpha + 1$.

Preuve.
1. C'est vrai.
2. La relation $\in$ est un bon-ordre sur α, soit β le plus petit élément. Si $\beta \neq \emptyset$ alors il contient au moins un élément γ d'où $\gamma < \beta$ et $\gamma \in \alpha$ (par transitivité), *absurde* car β minimal.
3. Le reste sera vu en TD ou en exercice.

□

Proposition 14.10. ▷ Si α et β sont des ordinaux et que l'on a $\alpha \leq \beta \leq \alpha + 1$ alors $\beta = \alpha$ ou $\beta = \alpha + 1$.

▷ Si X est un ensemble non vide d'ordinaux alors $\bigcap_{\alpha \in X} \alpha$ est le plus petit élément de X.

□

Définition 14.9. Soit β un ordinal.

▷ S'il existe α tel que $\beta = \alpha + 1$ alors on dit que β est un ordinal successeur ;

▷ Sinon, c'est un ordinal limite.

Exemple 14.4. Quelques ordinaux limites :

$$\omega \qquad \omega \cdot 2 \qquad \omega \cdot 3 \qquad \omega \cdot \omega$$

$$\omega^2 + \omega \qquad \omega^\omega \qquad \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot^{\omega}}}}}} .$$

Lemme 14.2. Soit X un ensemble d'ordinaux. Le plus petit élément de X est $\bigcap_{\alpha \in X} \alpha$.

Théorème 14.3. Si α et β sont des ordinaux alors une et une seule de ces propriétés est vérifiée :

$$\alpha = \beta \qquad \alpha \in \beta \qquad \alpha \ni \beta.$$

Preuve. Soit $X = \{\alpha, \beta\}$, on sait que $\alpha \cap \beta$ est le plus petit élément de X.

▷ Si $\alpha \cap \beta = \alpha$ alors $a \subseteq \beta$ donc $\alpha = \beta$ ou $\alpha \in \beta$.

▷ Si $\alpha \cap \beta = \beta$ alors $a \supseteq \beta$ donc $\alpha = \beta$ ou $\alpha \ni \beta$.

□

Proposition 14.11. Soit X un ensemble d'ordinaux. Alors l'ensemble $b := \bigcup_{\alpha\in X} \alpha$ est un ordinal. On le note $b = \sup_{\alpha\in X} \alpha$. De plus si $\gamma \in b$ alors il existe un certain $\alpha \in X$ tel que $\gamma \in \alpha$.

Preuve. En exercice. □

Proposition 14.12. Soit λ un ordinal non vide. On a :

$$\overbrace{\lambda \text{ est limite}}^{(1)} \iff \overbrace{\lambda = \bigcup_{\alpha\in\lambda} \alpha}^{(2)}.$$

Preuve.

1. Par contraposée, si λ n'est pas limite, c'est donc un successeur d'un certain ordinal β et donc $\lambda = \beta \cup \{\beta\}$. On a
$$\bigcup_{\alpha\in\lambda} \alpha = \beta \cup \bigcup_{\alpha\in\beta} \alpha = \beta \neq \lambda.$$
2. Soit λ limite. Montrons qu'il n'a pas de plus grand élément β. Sinon, $\lambda = \beta\cup\{\beta\}$. Donc, pour tout $\alpha \in \lambda$ il existe un certain $\gamma \in \lambda$ tel que $\alpha < \gamma$, *i.e.* $\alpha \in \gamma$. On en conclut que $\lambda = \bigcup_{\gamma\in\lambda} \gamma$.

□

Théorème 14.4 (Induction transfinie). Soit $\mathscr{P}$ une propriété sur les ordinaux. On suppose que :

- ▷ $\emptyset$ satisfait $\mathscr{P}$;
- ▷ pour tout ordinal α tel que, pour tout $\beta < \alpha$ satisfait $\mathscr{P}$, alors α satisfait $\mathscr{P}$:

$$\forall\alpha,\ (\forall\beta < \alpha,\ \mathscr{P}(\beta)) \implies \mathscr{P}(\alpha)\ ;$$

alors tous les ordinaux satisfont $\mathscr{P}$.

Preuve. Par l'absurde, soit α ne satisfaisant pas $\mathscr{P}$. Soit β le plus

petit ordinal de $\alpha \cup \{\alpha\}$ ne satisfaisant pas $\mathcal{P}$. Tous les ordinaux plus petit que β satisfont $\mathcal{P}$, d'où $\mathcal{P}(\beta)$, ***absurde***. On en conclut que α n'existe pas. □

Remarque 14.8. En pratique on décompose :

- ▷ on montre pour $\emptyset$;
- ▷ on montre pour α successeur ;
- ▷ on montre pour α limite.

Définition 14.10. Un ordinal α est *fini* si $\alpha = \emptyset$ ou si α et sous ses éléments sont des successeurs.

Proposition 14.13. L'ensemble des ordinaux finis ω est un ordinal. C'est le plus petit ordinal limite.

Preuve. En exercice. □

Lemme 14.3. Soit $f : \alpha \to \alpha'$ une fonction strictement croissante entre deux ordinaux α et α'. Alors $f(\beta) \geq \beta$ pour tout $\beta \in \alpha$. De plus, on a $\alpha' \geq \alpha$. Aussi si f est un isomorphisme alors $\alpha = \alpha'$ et f est l'identité.

Preuve.

- ▷ Soit β_0 le plus petit élément tel que $f(\beta_0) < \beta_0$. Comme f strictement croissante, on a $f(f(\beta_0)) < f(\beta_0) < \beta_0$ absurde car β_0 est le plus petit.
- ▷ Soit $\beta \in \alpha$. On a $f(\beta) \in \alpha'$ et $\beta \leq f(\beta)$ donc $\beta \in \alpha'$, donc $\beta \in \alpha'$, d'où $\alpha \subseteq \alpha'$ et donc $\alpha \leq \alpha'$.
- ▷ Si f est un isomorphisme alors f^{-1} est strictement croissante. On applique le point précédent à f^{-1}, d'où $\alpha = \alpha'$.
- ▷ Montrons que f est l'identité. On sait que, pour tout $\beta \in \alpha$, on a $f_{|\beta}$ strictement croissante de β dans $f(\beta)$ et bijective, d'où $\beta = f(\beta)$ par le point précédent. D'où f est l'identité.

□

Théorème 14.5. Tout ensemble bien ordonné est isomorphe à un ordinal. Cet ordinal ainsi que l'isomorphisme sont uniques.

Preuve. Cette preuve ressemble à une induction sans en être une. On aura le droit d'en faire quand on aura le théorème.

Si l'isomorphisme existe, il est unique grâce au lemme précédent. En effet, s'il y en a deux f et g alors $f \circ g^{-1}$ est un isomorphisme entre deux ordinaux égaux, donc c'est l'identité.

Notons $\mathscr{P}(x)$ la propriété « il existe un ordinal α_x et un isomorphisme $f_x : S_{\leq x} \to \alpha_x$ » où $S_{\leq x} := \{y \in X \mid y \leq x\}$. Pour montrer $\mathscr{P}(x)$ pour tout $x \in X$, on pose

$$Y := \{x \in X \mid \mathscr{P}(x) \text{ est vraie}\}.$$

et on montre $Y = X$.

Supposons $Y \neq X$ et soit $a = \min(X \setminus Y)$.

- Si Y a un plus grand élément b, alors il existe un isomorphisme $f_b : S_{\leq b} \to \alpha_b$ (car $\mathscr{P}(b)$). Or, $S_{\leq a} = S_{\leq b} \cup \{a\}$ (il faudrait montrer que Y est un segment initial de X). Et, on construit $f_a : S_{\leq a} \to \alpha_b \cup \{\alpha\}_b$ qui est un isomorphisme, donc $a \in Y$, ***absurde***.
- Si Y n'a pas de plus grand élément, on considère $\alpha := \bigcup_{x \in Y} \alpha_x$ un ordinal. Pour tout $x < \alpha$ il existe un isomorphisme de $S_{\leq x}$ dans α_x. Si on prend $x < y < \alpha$ alors $(f_y)_{|S_{\leq x}}$ est un isomorphisme et, par unicité, on a donc que f_y prolonge f_x. On peut définir un isomorphisme $f_{\leq \alpha}$ comme limite des f_x pour $x < \alpha$. C'est un isomorphisme de $S_{<a}$ dans $\beta := \bigcup_{x<\alpha} \alpha_x$. On peut prolonger f en $f_X : S_{\leq \alpha} \to \beta + 1$ où $\alpha \mapsto \beta$, d'où $\alpha \leq Y$, ***absurde***.

□

Lemme 14.4 (Définition récursive transfinie des fonctions)**.** Soient

- α un ordinal ;

- ▷ S une collection ;
- ▷ $\mathscr{F}$ la collection des applications définies sur les ordinaux $\beta \leq \alpha$ et prenant leur valeurs dans S ;
- ▷ F une relation fonctionnelle de domaine $\mathscr{F}$ à valeur dans S.

Alors il existe une fonction f dans $\mathscr{F}$ (et une unique définie sur α) telle que :

$$(\star) \qquad \text{pour tout } \beta < \alpha \qquad f(\beta) = F(f_{|\beta}).$$

Preuve. **Unicité.** Soient f et g satisfaisant $(\star)$. Montrons $\mathscr{P}(\beta)$: « si $\beta < \alpha$ alors $f(\beta) = g(\beta)$ » par induction transfinie.

- ▷ On a directement $\mathscr{P}(\emptyset)$ car il y a une unique fonction de $\emptyset \to \emptyset$.
- ▷ Supposons que $f(\gamma) = g(\gamma)$ pour tout $\gamma < \beta$. Alors $f_{|\beta} = g_{|\beta}$ et donc par $(\star)$ on a $f(\beta) = g(\beta)$.

Existence. Soit τ l'ensemble des ordinaux $\gamma \in \alpha$ tels qu'il existe $f_\gamma \in \mathscr{F}$ définie sur γ et vérifiant $(\star)$. Alors τ est un segment initial de α donc un ordinal. Par unicité si $\gamma < \gamma'$ alors $f_{\gamma'}$ prolonge f_γ. On définit f_τ par $f_\tau(\gamma) := F(f_\gamma)$ si $\gamma < \tau$.

- ▷ Si $\tau \in \alpha$ alors f_τ prolonge tous les f_γ et donc $\tau \in \tau$, ***impossible***.
- ▷ D'où $\tau = \alpha$.

□

Exercice 14.1. Généraliser la preuve ci-dessus en replaçant α par la classe de tous les ordinaux $\mathfrak{O}$ (et remplacer $\mathscr{F}$ par autre chose).

Proposition 14.14. 1. La classe $\mathfrak{O}$ n'est pas un ensemble.
2. Il n'existe pas de relation fonctionnelle bijective entre $\mathfrak{O}$ et un ensemble a.

Preuve. 1. En effet, supposons $\mathfrak{O}$ un ensemble. On a que $\mathfrak{O}$ est transitif et $\in$ y définit un ordre total, donc $\mathfrak{O}$ est un ordinal.

D'où $\mathfrak{O} \in \mathfrak{O}$ ce qui est impossible pour un ordinal.

2. Sinon $\mathfrak{O}$ serait un ensemble.

□

14.3 Axiome de choix et variantes équivalentes.

Les axiomes du choix sont exprimable au premier ordre !

AC 1. Le produit d'une famille d'ensembles non vides est non vide.

AC 2. Pour tout ensemble a non vide, il existe une fonction de $\mathscr{P}(a)$ dans a tel que si $x \subseteq a$ est non vide alors $f(x) \in x$. (C'est une fonction de choix.)

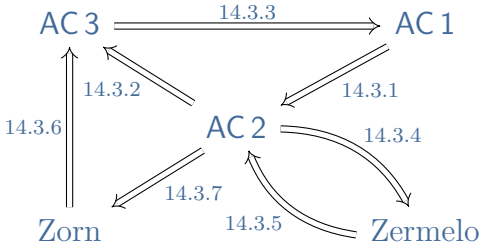
AC 3. Si a est un ensemble dont tous les éléments sont non vides et deux à deux disjoints alors il existe un ensemble c tel que, pour tout $x \in a$, $x \cap c$ a exactement un élément.

(Lemme de) Zorn. Tout ensemble non vide partiellement ordonné et inductif admet un élément maximal.

On rappelle qu'un ensemble partiellement ordonné X est inductif si $X \neq \emptyset$ et si tout sous-ensemble $Y \subseteq X$ totalement ordonné admet un majorant dans X.

(Lemme de) Zermelo. Tout ensemble non vide peut être mini d'un bon ordre (*i.e.* un ordre total où toute partie non vide a un plus petit élément).

En supposant les axiomes ZF 1, . . . , ZF 5, on va montrer les implications suivantes :



14.3.1 AC 1 implique AC 2.

On rappelle que l'ensemble $\prod_{i\in I} a_i$ est l'ensemble de fonctions f de la forme $f : I \to \bigcup_{i\in I} a_i$ tel que $f(i) \in a_i$ pour tout i.

Soit a non vide. On considère $\prod_{\emptyset\neq x\subseteq a} x$ qui est non vide par AC 1. Soit f un de ces éléments. On a, pour tout $\emptyset \neq x \subseteq a$, que $f(x) \in x$ donc f est une fonction de choix.

14.3.2 AC 2 implique AC 3.

Soit a un ensemble dont les éléments sont non vides et deux à deux disjoints. On considère $b = \bigcup_{x\in a} x$ qui est un ensemble. Par AC 2, on a une fonction de choix f sur $\wp(b)$. On prend $c = \{f(x) \mid x \in a\}$. Comme les x sont disjoints, on obtient la propriété recherchée.

14.3.3 AC 3 implique AC 1.

Soit $X = \prod_{i\in I} a_i$ un produit d'ensemble non vides. On considère $A := \{\, \{i\} \times a_i \mid i \in I \,\}$. Par AC 3, il existe c tel que, pour tout $x \in A$, $x \cap c$ a exactement un élément. D'où, c peut s'écrire

$$c = \{(i, d_i) \mid i \in I \text{ et } d_i \in a_i\}.$$

On a donc $c \in \prod_{i\in I} a_i$ (c'est le graphe d'une fonction) et donc $\prod_{i\in I} a_i \neq \emptyset$.

14.3.4 AC 2 implique Zermelo.

Soit a un ensemble non vide.

> **Remarque 14.9** (Idée). L'idée est qu'on utilise $f : \mathcal{P}(a) \to a$ une fonction de choix pour définir l'ordre. On peut imaginer définir $x \leq y$ ssi $f(\{x, y\}) = x$ (on traite donc f comme la fonction minimum), mais on n'a pas la transitivité. Il faut être plus futé.
>
> On va construire en partant du plus petit une bijection entre a et un ordinal.

- $h(0) := f(a)$
- $h(1) := f(a \setminus \{h(0)\})$
- $h(2) := f(a \setminus \{h(0), h(1)\})$
- $\vdots$
- $h(\omega) = f(a \setminus \{h(0), h(1), \ldots\})$
- $\vdots$

On s'arrête quand $a \setminus \{h(0), h(1), \ldots\}$ est vide.

Soit $\theta \notin a$ (pour « détecter » quand l'ensemble $a \setminus \{h(0), \ldots\}$ est vide). Il existe car a est un ensemble donc pas l'univers tout entier.

On définit

$$F(\alpha) := \begin{cases} f(a \setminus \{F(\beta) \mid \beta < \alpha\}) & \text{si } a \setminus \{F(\beta) \mid \beta < \alpha\} \neq \emptyset \\ \theta & \text{sinon} \end{cases}.$$

D'après le dernier lemme de la section précédente, on peut construire l'application F ainsi et elle est unique.

S'il n'existe pas d'ordinal α tel que $F(\alpha) = \theta$ alors F est une injection de $\mathbb{O}$ dans a. ***Absurde***. Il existe donc α tel que $F(\alpha) = \theta$.

Le sous-ensemble $\{\beta \in \alpha \mid F(\beta) = \theta\}$ a un plus petit élément β. Montrons que $F_{|\beta}$ est une bijection de β dans α.

- D'une part, on sait que c'est une injection.
- D'autre part, $F(\beta) = \theta$ implique $\{F(\gamma) \mid \gamma < \beta\} = a$ donc $F_{|\beta}$ est une surjection.

On définit le bon ordre $x \prec y$ ssi $F^{-1}(x) < F^{-1}(y)$.

14.3.5 Zermelo implique AC 2.

Soit a non vide. Il existe un bon-ordre $<$ sur a. Soit $\emptyset \neq x \subseteq a$. On définit $f(x) = \min x$, c'est bien une fonction de choix.

14.3.6 Zorn implique AC 3.

Soit a un ensemble dont les éléments sont disjoints et non vides. On pose :

$$b := \bigcup_{x \in a} x \quad \text{et} \quad X := \{c \subseteq b \mid \forall x \in a,\ |c \cap x| \leq 1\}.$$

Montrons que l'ensemble $(X, \subsetneq)$ est inductif.

Soit $Y \subseteq X$ est totalement ordonné. Montrons que Y a un majorant dans X. On pose $z = \bigcup_{y \in Y} y$ qui majore Y. On a bien $z \in X$ (on ne duplique pas les éléments). On en conclut que X est inductif. Soit d un élément maximal de X (il existe par Zorn).

- ▷ S'il existe $x \in a$ tel que $x \cap d = \emptyset$ alors prenons $u \in x$ et posons $d_1 := d \cup \{u\}$. D'où $d_1 \in X$ et $d \subsetneq d_1$ donc d non maximal, ***absurde***.
- ▷ Pour tout $x \in a$, on a $|d \cap x| = 1$, d'où d est l'ensemble recherché (appelé c dans AC 3).

14.3.7 AC 2 implique Zorn.

Soit a un ensemble inductif.

> **Remarque 14.10** (Idée). On construit une chaîne dans a (*i.e.* un ensemble totalement ordonné) de taille maximale (c'est ici qu'on utilise la fonction de choix de AC 2). Elle a un majorant car a est inductif. Ce majorant va être l'élément maximal.

Soit $f : \wp(a) \to a$ une fonction de choix donnée par AC 2. Si $x \subseteq a$ on appelle majorant strict de x un $y \in a$ tel que $z < y$ pour tout $z \in x$.

> **Remarque 14.11** (Idée – suite).
> - ▷ On part de $\emptyset$: tout élément de a a un majorant strict de $\emptyset$ en choisissant $a_1 = f(a)$.
> - ▷ Soit X_1 l'ensemble des majorants de $\{a_1\}$. On pose $a_2 := f(X_1)$.

▷ Soit X_2 l'ensemble des majorants de $\{a_1, a_2\}$. On pose $a_3 := f(X_2)$.

Formellement, soit $C := \{x \subseteq a \mid x \text{ a un majorant strict dans } a\}$. On a $\emptyset \in C$. On définit

$$\begin{aligned} m : C &\longrightarrow a \\ x &\longmapsto f(\{y \in a \mid y \text{ est un majorant strict de } x \text{ dans } a\}). \end{aligned}$$

On définit par induction la chaîne maximale (dernier lemme de la section précédente). Soit $\theta \notin a$. On définit :

$$F(\alpha) := \begin{cases} m(\{F(\beta) \mid \beta < \alpha\}) & \text{si } \{F(\beta) \mid \beta < \alpha\} \in C \\ \theta & \text{sinon} \end{cases}.$$

La fonction F n'est pas une injection de $\mathbb{O}$ dans a donc il existe un ordinal α tel que $F(\alpha) = \theta$. Comme $\alpha + 1$ est un ordinal, l'ensemble $\{\beta \in \alpha + 1 \mid F(\beta) = \theta\}$ a un plus petit élément α_0. D'où l'ensemble $\{F(\beta) \mid \beta < \alpha_0\}$ n'a pas de majorant strict mais a un majorant M car a inductif. Et, a n'a pas d'élément plus grand que a. Ainsi M est maximal dans a.

14.3.8 Indépendance de ZF et de l'axiome du choix.

On a les deux théorèmes suivants (que l'on admet).

Théorème 14.6 (Gödel, 1938)**.** S'il est cohérent, ZF ne réfute pas l'axiome du choix.

Théorème 14.7 (Cohen, 1963)**.** S'il est cohérent, ZF ne montre pas l'axiome du choix.

Ainsi l'axiome du choix est indépendant de ZF. Cependant, il existe des versions plus faibles : axiome du choix dépendant (ACD), axiome du choix dénombrable (AC_ω).

15. Exemple de théories décidables.

Dans ce chapitre, on traite de l'élimination des quantificateurs dans les corps réels clos (et les corps algébriquement clos).

15.1 De quoi on parle ?

15.1.1 L'élimination des quantificateurs.

Définition 15.1. Une théorie T (de la logique du 1er ordre) admet *l'élimination des quantificateurs* si pour toute formule $\varphi(\bar{y})$, il existe une formule sans quantificateurs $\psi(\bar{y})$ telle que $T \vdash \forall \bar{y}\, (\varphi(\bar{y}) \leftrightarrow \psi(\bar{y}))$.

Lemme 15.1. Une théorie T élimine les quantificateurs ssi pour toute formule $\varphi(x, \bar{y})$ sans quantificateurs, il existe une formule $\psi(\bar{y})$ sans quantificateurs et $T \vdash \forall \bar{y}\, (\exists x\, \varphi(x, \bar{y}) \leftrightarrow \psi(\bar{y}))$.

Preuve. Idée de la preuve :

- ▷ « $\Longrightarrow$ ». C'est un cas particulier.
- ▷ « $\Longleftarrow$ ». Toute formule est équivalente à une formule prénexe, c'est-à-dire une formule où les quantificateurs sont à la racine :

$$\mathsf{Q}_1 x_1\, \mathsf{Q}_2 x_2\, \ldots\, \mathsf{Q}_n x_n\, \varphi(x_1, \ldots, x_n),$$

où $\varphi(\cdots)$ est sans quantificateurs. Pour démontrer que

toute formule est équivalente à une formule prénexe, on procède par induction sur la formule, et on doit potentiellement procéder à des cas d'α-renommage au besoin.

Pour toute formule sous forme prénexe, le lemme est vrai.

□

Exemple 15.1. La théorie des booléens est la théorie

$$T_{\text{bool}} := \{\forall x\, x = 0 \vee x = 1, 0 \neq 1\},$$

sur le langage $\mathscr{L} = \{0, 1\}$. Cette théorie admet l'élimination des quantificateurs. En effet, par exemple, une formule

$$F := \exists x_1 \cdots \exists x_n\, (x_1 = 1 \vee x_2 = 0 \vee x_4 = 1) \wedge \cdots),$$

est équivalente à $\top$ ou $\bot$.

Exemple 15.2. Sur le langage $\mathscr{L}_{\text{co}} = \{0, 1, +, \times, \leq\}$, la théorie $T := \mathbf{Th}(\mathbb{R})$ admet l'élimination des quantificateurs. En effet, par exemple, la formule

$$\varphi(a, b, c) := \exists x\, (a \times x \times x + b \times x + c = 0)$$

est équivalente à la formule sans quantificateurs

$$\psi(a, b, c) := (a \neq 0 \wedge b^2 - 4ac \geq 0) \vee (a = 0 \wedge b \neq 0) \vee (a = 0 \wedge b = 0 \wedge c = 0)\ .$$

15.1.2 Les corps réels clos et le théorème de Tarski.

Définition 15.2. Un *corps réel clos* est un corps commutatif ordonné dans lequel on a le théorème des valeurs intermédiaires pour les polynômes à 1 variable.

La théorie T_{CRC} est la théorie du 1er ordre et ses axiomes sont :

- ▷ axiomes de corps commutatifs ;

- ▷ axiomes de relation d'ordre total ;
- ▷ $1 > 0$;
- ▷ axiomes de corps ordonné (compatibilité de $+$ et $\times$ avec $\leq$) :

$$\forall x\, \forall y\, \forall z \begin{pmatrix} x \leq y \to x + z \leq y + z \\ \wedge \\ (z \geq 0 \wedge x \leq y) \to x \times y \leq y \times z \\ \wedge \\ (z \leq 0 \wedge x \leq y) \to x \times y \geq y \times z \end{pmatrix} ;$$

- ▷ schéma d'axiomes pour le théorème des valeurs intermédiaires : pour $n \in \mathbb{N}$,

$$\begin{gathered} \forall a_0 \dots a_n\, \forall x\, \forall y \\ a_0 + a_1 x + \cdots + a_n x^n \geq 0 \wedge a_0 + a_1 y + \cdots + a_n y^n \leq 0 \\ \downarrow \\ \exists z\, (x \leq z \leq y \vee y \leq z \leq x) \wedge a_0 + a_1 z + \cdots + a_n z^n = 0. \end{gathered}$$

Exemple 15.3. Exemples de corps réels clos : $\mathbb{R}$ les réels, $\bar{\mathbb{Q}} \cap \mathbb{R}$ les nombres réels algébriquement clos.

Qu'en est-il de $\mathbb{C}$? Si on a $i \geq 0$ et on a $1 \leq 2$ donc $i \leq 2i$ et par multiplication par i on a $-1 \leq -2$, absurde ! Le même procédé fonctionne si l'on suppose $i \leq 0$. Il n'y a pas de manière d'ordonner $\mathbb{C}$ de telle sorte à ce qu'il soit un corps réel clos.

Proposition 15.1.
1. Un corps réel clos est de caractéristique 0.
2. Dans un corps réel clos, on a le théorème de Rolle (entre deux racines d'un polynôme, la dérivée s'annule).

Preuve. Idée de la preuve :

1. On a $1 > 0$ donc $2 > 1 > 0$ donc $3 > 0$, *etc.* On montre, par récurrence, pour tout n que $n > 0$ et donc $n \neq 0$.

2. On montre que si la dérivée est de signe constant alors le polynôme est monotone d'où le théorème de Rolle.

□

À quoi ressemblent les formules dans $\mathscr{L}_{\text{co}}$?

- Les termes représentent des polynômes à plusieurs variables et à coefficients dans $\mathbb{N}$.
- Les formules atomiques représentent des équations et inéquations entre polynômes :
$$P(X) \leq Q(X) \text{ ou } P(X) = Q(X),$$
et même $P(X) \geq 0$ ou $P(X) = 0$ avec P à coefficient dans $\mathbb{Z}$.
- Les formules sans quantificateur sont équivalentes à des formules de la forme
$$\bigvee_i \bigwedge_j (P_{i,j} \ \Delta_{i,j} \ 0),$$
où $\Delta_{i,j} \in \{<, >, =\}$.
- Les formules sont équivalentes à des formules sous forme prénexe de la forme
$$\mathsf{Q}_1 x_1 \dots \mathsf{Q}_n x_n \bigvee_i \bigwedge_j (P_{i,j} \ \Delta_{i,j} \ 0),$$
avec $\mathsf{Q}_i \in \{\forall, \exists\}$.

Théorème 15.1 (Tarski)**.** La théorie des corps réels clos admet l'élimination des quantificateurs. Elle est axiome-complète et décidable.

Preuve. En supposant que T_{CRC} admet l'élimination des quantificateurs, alors on a une théorie axiome-récursive ~~qui contient les entiers donc indécidable par Gödel~~. Non ! On ne contient pas $\mathscr{P}_0$! En effet, l'axiome A1 n'est pas vérifié : on n'a pas $\textcircled{S} x \neq 0$!

Soit F une formule close de $\mathscr{L}_{\text{co}}$. Montrer que $T_{\text{CRC}} \vdash F$ ou $T_{\text{CRC}} \vdash \neg F$. Il existe une formule sans quantificateurs G et $T_{\text{CRC}} \vdash F \leftrightarrow G$ et G n'a pas de variable. Ainsi G est équivalent à une conjonction

de disjonction de formules équivalentes à

$$\textcircled{n} > \textcircled{m} \text{ ou } \textcircled{n} = \textcircled{m}.$$

La valeur de vérité ne dépend pas du modèle, d'où $T_{\text{CRC}} \vdash G$ ou $T_{\text{CRC}} \vdash \neg G$, donc $T_{\text{CRC}} \vdash F$ ou $T_{\text{CRC}} \vdash \neg F$, et donc T_{CRC} est axiome-complète.

Comme T_{CRC} est axiome-récursive, pour décider si $T_{\text{CRC}} \vdash F$, il suffit d'énumérer toutes les preuves jusqu'à en trouver une de F ou de $\neg F$. □

15.2 La méthode d'élimination.

15.2.1 Rappels et exemples.

Il suffit de montrer le lemme ci-dessous.

Lemme 15.2. Si pour toute formule F de la forme $\exists x \bigvee_i \bigwedge_k P_{i,j} \ \Delta_{i,j} 0$ avec $P_{i,j}$ des polynômes et $\Delta_{i,j} \in \{<, >, =\}$, il existe une formule sans quantificateurs G telle que

$$T_{\text{CRC}} \vdash \forall \bar{y} \, G(\bar{y}) \leftrightarrow F(\bar{y})$$

alors T_{CRC} admet l'élimination des quantificateurs.

Idée de la méthode :

- ▷ On part d'un polynôme, par exemple $ax^2 + bx + 1$.
- ▷ On calcule des « quantités importantes » (des polynômes de degré 0 en x), ici a et $a^2 - 4a$.
- ▷ On trouve des « conditions de signe » qui permettent de satisfaire la formule, ici $a \neq 0 \wedge a^2 - 4a \geq 0$.

Définition 15.3. Avec $P \in \mathbb{Z}[\bar{Y}][X] = \mathbb{Z}[Y_1, \ldots, Y_n][X]$, les poly-

nômes s'écrivent comme

$$P(X) = a_n X^n + \cdots + a_0 \text{ où } n \geq 1,\ a_n \neq 0 \text{ et } a_i \in \mathbb{Z}[\bar{Y}],$$

et on définit les opérations :

- ▷ *dérivée* $\mathrm{D}(P) := \frac{\partial P(X)}{\partial X}$;
- ▷ *extraction du coefficient dominant* $\mathrm{E}(P) := a_n$;
- ▷ *omission du terme dominant* $\mathrm{O}(P) := a_{n-1}X^{n-1} + \cdots + a_0$;
- ▷ *reste modifié* $\mathrm{MR}(P, Q)$:
  si $P = a_n X^n + \cdots + a_0$ et $Q = b_n X^n + \cdots + b_0$ où

$$n = \deg P \geq m = \deg Q \geq 1$$

et $P \neq Q$ alors $\mathrm{MR}(P, Q)$ est l'unique polynôme de $\mathbb{Z}[\bar{Y}][X]$ de degré $r < m$ tel qu'il existe $L \in \mathbb{Z}[\bar{Y}][X]$ et

$$(b_n)^{nm+1} \times P = Q \times L + R.$$

Exemple 15.4. Si $P = X^4$ et $Q = 3X^2 + X + 1$ alors

$$\begin{array}{rl|l}
X^4 & & 3X^2 + X + 1 \\
\cline{3-3}
-X^4 - \frac{1}{3}X^3 - \frac{1}{3}X^2 & & \frac{1}{3}X^2 - \frac{1}{9}X - \frac{2}{27} \\
\cline{1-1}
-\frac{1}{3}X^3 - \frac{1}{3}X^2 & & \\
\frac{1}{3}X^3 + \frac{1}{9}X^2 + \frac{1}{9}X & & \\
\cline{1-1}
-\frac{2}{9}X^2 + \frac{1}{9}X & & \\
\frac{2}{9}X^2 + \frac{2}{27}X + \frac{2}{27} & & \\
\cline{1-1}
\frac{5}{27}X + \frac{2}{27} & &
\end{array}$$

et le reste modifié est $\mathrm{MR}(P, Q) = 3^3(\frac{5}{27}X + \frac{2}{27}) = 5X + 2$.

15.2.2 Énoncé comme lemme clé.

Lemme 15.3 (Informel). À partir d'un ensemble de polynômes S, on obtient en temps fini un ensemble fini de polynômes $\mathrm{BC}S$ de degré 0 en appliquant les quatre opérations D, E, O et MR. [1]

Exemple 15.5. À partir de $S = \{\overbrace{aX^2 + bX + 1}^{p_0}\}$, on a

- ▷ on commence par ajouter p_0 ;
- ▷ d'abord les dérivées, omissions et extractions : on ajoute les polynômes $2aX + a$, a et $aX + 1$, $2a$, 1 et 0 ;
- ▷ ensuite on calcule le reste modifié

$$\mathrm{MR}(aX^2 + aX + 1, 2aX + a) = 4a^2 - a^3,$$

et on l'ajoute ;

- ▷ on calcule le reste modifié

$$\mathrm{MR}(aX^2 + aX + 1, aX + 1) = a,$$

et on l'ajoute (il y est déjà) ;

- ▷ on calcule le reste modifié

$$\mathrm{MR}(3aX + a, aX + 1) = a^2 - 2a,$$

et on l'ajoute ;

- ▷ on ne conserve que les polynômes de degré 0.

Dans l'exemple on obtient (après suppression des termes inutiles pour les comparaisons à 0),

$$\mathrm{BC}S = \{a, 4a^2 - a^3, a^2 - 2a\}.$$

On a, en théorie, 27 conditions de signe possibles ($3^{|\mathrm{BC}S|}$) :

- ▷ $a > 0$ et $4a^2 - a^3 > 0$ et $A^2 - 2a < 0$,
- ▷ $a > 0$ et $4a^2 - a^3 < 0$ et $a^2 - 2a < 0$,
- ▷ $a = 0$ et $a^2 - a^3 > 0$ et $a^2 - 2a > 0$,
- ▷ *etc* pour les 24 autre cas.

1. I ♡ le lemme de König.

On traite deux cas : $a > 0$ et $4a^2 - a^3$ et $a^2 - 2a$.

X	$-\infty$		γ_2		γ_1		$+\infty$
a		$>$	$>$	$>$	$>$	$>$	
$4a^2 - a^3$		$>$	$>$	$>$	$>$	$>$	
$a^2 - 2a$		$<$	$<$	$<$	$<$	$<$	
$aX + 1$	$-\infty$	$<$	$<$	$<$	0	$>$	$+\infty$
$2aX + a$	$-\infty$	$<$	0	$>$	$>$	$>$	$+\infty$
$aX^2 + aX + 1$	$+\infty$	$>$	$>$	$>$	$>$	$>$	$+\infty$

15.3 Corps algébriquement clos.

Définition 15.4. Un *corps algébriquement clos* est un corps commutatif dans lequel tout polynôme a une racine.

Exemple 15.6. Le corps $\mathbb{C}$ est algébriquement clos. En effet, il s'agit du *théorème fondamental de l'algèbre*, *i.e.* un polynôme de degré n a n racines comptées avec multiplicité.

Tout polynôme est ainsi un produit de polynômes de degré 1.

Définition 15.5. La *théorie des corps algébriquement clos* est la théorie formée des :

- ▷ axiomes de corps ;
- ▷ du schémas d'axiomes, noté Clos_n, pour tout $n \in \mathbb{N}$,

$$\forall a_0 \dots \forall a_n \, (a_1 \neq 0 \vee \dots \vee a_n \neq 0 \to \exists b \, a_0 + a_1 b + \dots + a_n b^n = 0).$$

Définition 15.6. Un corps est de *caractéristique* $p \in \mathbb{N}^\star$ s'il est modèle de l'ensemble Car_p définie par

$$\{(1 \neq 0) \wedge (1+1 \neq 0) \wedge \dots \wedge (\underbrace{1 + \dots + 1}_{p-1} \neq 0) \wedge (\underbrace{1 + \dots + 1}_{p} = 0)\}.$$

Un corps est de *caractéristique* 0 s'il est modèle de l'ensemble Car_0 définie par

$$\{1 \neq 0, 1+1 \neq 0, 1+1+1 \neq 0, \ldots\}.$$

La *théorie des corps algébriquement clos de caractéristique* $p \in \mathbb{N}$ est :

$$\mathrm{ACF}_p := \{\text{Axiomes des corps}\} \cup \{\, \mathrm{Clos}_n \mid n \in \mathbb{N} \,\} \cup \mathrm{Car}_p.$$

Exemple 15.7. Les corps $\mathbb{C}$ et $\bar{\mathbb{Q}}$ sont modèles de cette théorie. ***Attention***, $\mathbb{F}_p$ ne l'est pas (et $\mathbb{F}_{p^n}$ non plus), il faut prendre sa clôture algébrique $\bar{\mathbb{F}}_p$ et $\bar{\mathbb{F}}_{p^n}$.

Remarque 15.1.
- ▷ Tous les corps finis sont de la forme $\mathbb{F}_{p^n}$ avec p premier.
- ▷ Un élément a est dit *algébrique* sur le corps $\Bbbk$ si c'est la racine d'un polynôme à coefficient dans $\Bbbk$. On dit que a est *algébrique de degré* q si le polynôme minimal dont a est racine est de degré q.

Exemple 15.8.
- ▷ Le nombre $\sqrt{3}$ est algébrique sur $\mathbb{Q}$ de degré 2.
- ▷ Le nombre i est algébrique sur $\mathbb{Q}$ de degré 2.
- ▷ Le nombre $\sqrt[3]{2}$ est algébrique sur $\mathbb{Q}$ de degré 3.
- ▷ Le nombre π n'est pas algébrique sur $\mathbb{Q}$.

Remarque 15.2. Si a est algébrique de degré q sur $\Bbbk$ alors $\Bbbk(a)$ est le corps engendré par $\Bbbk$ et a. C'est l'ensemble des polynômes de degré $\leq q-1$ sur $\Bbbk$, et on définie le produit modulo un polynôme minimal de a.

Exemple 15.9. On a $\mathbb{R}(i) = \mathbb{R}[X]/(X^2-1) \cong \mathbb{C}$. Le produit est :

$$\begin{aligned}(aX+b)(cX+d) &= acX^2 + X(ad+bc) + bd \\ &= (ad+bc)X + bd - ac.\end{aligned}$$

En particulier, si a est de degré q sur $\mathbb{F}_{p^n}$ alors $\mathbb{F}_{p^n}(a) = \mathbb{F}_{p^{qn}}$.

Théorème 15.2 (Tarski–bis)**.** Pour tout p, la théorie des corps algébriquement clos de caractéristique p admet l'élimination des quantificateurs. Elle est complète et décidable.

Preuve. Comme la dernière fois, il suffit de montrer pour toute formule de la forme

$$\exists x\, (P_1(x) = 0 \wedge \cdots \wedge P_n(x) = 0 \wedge Q(x) \neq 0),$$

il existe une formule sans quantificateurs équivalente dans ACF_p. On continue la preuve sur un exemple. □

Exemple 15.10. On élimine les quantificateurs sur

$$\exists x\, (ax^2 + ax + 1 = 0 \wedge ax + 1 \neq 0),$$

avec la caractéristique $p = 0$. On a les polynômes suivants :

- $p_0(X) = aX^2 + aX + 1$
- $p_1(X) = \mathrm{D}p_0(X) = 2aX + a$
- $p_2(X) = \mathrm{E}p_0 = a$
- $p_3(X) = aX + 1$
- $p_4(X) = \mathrm{MR}(p_0, p_1) = 4a^2 - a^3$
- $p_2(X) = \mathrm{MR}(p_0, p_3) = a$
- $p_5(X) = \mathrm{MR}(p_1, p_3) = a^2 - 2a.$

Les « conditions de signe » sont $= 0$ ou $\neq 0$ (notés 0 et $\neq$).

On se place dans un cas exemple :

	autres	γ_1	γ_2	γ_3	γ_4
a	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$
$4a^2 + a^3$	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$
$a^2 - 2a$	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$
$aX + 1$	$\neq$	0	$\neq$	$\neq$	$\neq$
$2aX + a$	$\neq$	$\neq$	0	$\neq$	$\neq$
$aX^2 + aX + 1$	$\neq$	$\neq$	$\neq$	0	0

.

Ainsi, pour $a \neq 0$, $4a^2 - a^3 \neq 0$, $a^2 - 2a \neq 0$ alors on a

$$\exists x\ (ax^2 + ax + 1 = 0 \wedge ax + 1 \neq 0).$$

Avec les autres cas, on peut en déduire que

$$\exists x\ (ax^2 + ax + 1 = 0 \wedge ax + 1 \neq 0)$$

est équivalente à

$$\bigvee_{\substack{\text{tableau de la condition de signe} \\ \text{a une colonne qui convient}}} \text{(conditions de signe)}.$$

Exercice 15.1. En déduire que ACF_p est complète et décidable.

Remarque 15.3. En 2010, une preuve ~~Coq~~ Rocq de l'élimination des quantificateurs de cette théorie a été publiée par Cyril Cohen et Assia Mahboubi.

15.3.1 Applications aux mathématiques.

Théorème d'Ax–Grothendieck.

Théorème 15.3 (Ax–Grothendieck)**.** Si P est un polynôme de $\mathbb{C}^n$ dans $\mathbb{C}^n$ injectif alors il est bijectif (et son inverse est un polynôme !).

On va prouver ce théorème en trois lemmes.

Lemme 15.4. Si φ est une formule qui admet comme modèle un corps algébriquement clos de caractéristique arbitrairement grande, alors φ admet comme modèle un corps algébriquement clos de caractéristique 0.

Preuve. On utilise le théorème de compacité de la logique du 1er ordre. Soit $T := \mathrm{ACF}_0 \cup \{\varphi\}$. Montrons que T a un modèle. Pour cela, on montre que T est finiment satisfiable. Soit $T' \subseteq_{\text{fini}} T$. Soit n le plus grand entier tel que

$$(\underbrace{1+1+\cdots+1}_{n} \neq 0) \in T'.$$

Soit $p > n$ un nombre premier tel que φ admet comme modèle un corps algébriquement clos $\Bbbk$ de caractéristique p (qui existe par hypothèse). D'où $\Bbbk \models \varphi$, et

$$\Bbbk \models \{\text{Axiomes des corps}\} \cup \{\,\mathrm{Clos}_n \mid n \in \mathbb{N}\,\}.$$

D'où, $\Bbbk \models \mathrm{ACF}_p$, et donc $\Bbbk \models T'$. Ainsi T finiment satisfiable donc T satisfiable. On en déduit que φ admet un modèle de caractéristique 0. □

Lemme 15.5. Soit $\Bbbk$ un corps fini et soient $n \in \mathbb{N}^\star$ et $P : \Bbbk^n \to \Bbbk^n$ un polynôme injectif. Alors P est bijectif.

Preuve. Comme $\Bbbk^n$ est fini alors P est bijectif. □

Lemme 15.6. Soit $\Bbbk$ un corps fini et soient $n \in \mathbb{N}^\star$ et $\bar{\Bbbk}$ la clôture

algébrique de $\Bbbk$. Soit $P : \bar{\Bbbk}^n \to \bar{\Bbbk}^n$ un polynôme injectif. Alors P est bijectif.

Preuve. On suppose P non surjectif, il existe donc $\bar{b} = (b_1, \dots, b_n) \in \bar{\Bbbk}^n \setminus P(\bar{\Bbbk}^n)$ des nombres algébriques dans $\Bbbk$. Ils sont raciles de polynômes minimaux à coefficients dans $\Bbbk$. Soient $\bar{a} = (a_1, \dots, a_m)$ les coefficients de ces polynômes, ce sont des éléments de $\bar{\Bbbk}$. Soient $\bar{c}$ les coefficients de P.

Soit $\Bbbk' := \Bbbk(\bar{a}, \bar{b}, \bar{c})$, c'est un corps fini. On a $P : \Bbbk'^n \to \Bbbk'^n$ injectif pas surjectif, qui est impossible d'après le lemme précédent. □

On peut donc montrer le théorème d'Ax–Grothendieck.

Pour un degré d fini et un entier n fixé, on va construire la formule $\phi_{n,d}$ qui exprime qu'un polynôme de degré $\leq d$ de $\Bbbk^n$ dans $\Bbbk^n$ qui est injectif et surjectif. Soit $M(n,d)$ l'ensemble fini des monômes unitaires de degré $\leq d$ avec n variables $x_1, \dots, x_n$:

$$M(n,d) := \{1, x_1, x_2, x_1x_2, \dots, x_1^d, x_1^{d-1}x_2, \dots\}.$$

On pose la formule, notée $\varphi_{n,d}$.

$$\forall (a_{m,i})_{m \in M(n,d), i \in [\![1,n]\!]}$$

$$\left(\forall x_1 \dots x_n \forall y_1 \dots y_n \bigwedge_{i=1}^{n} \sum_{m \in M(n,d)} a_{m,i} m(x_i) = \sum_{m \in M(n,d)} a_{m,i} m(y_i) \to \bigwedge_{i=1}^{n} x_i = y_i \right)$$

$$\downarrow$$

$$\forall y_1 \dots y_n \exists x_1 \dots x_n \bigwedge_{i=1}^{n} y_i = \sum_{m \in M(n,d)} a_{m,i} m(x_i).$$

Par le troisième lemme, pour tout corps fini $\Bbbk$, on a $\bar{\Bbbk} \models \varphi_{n,d}$ donc pour tout p premier, on a $\bar{\mathbb{F}}_p \models \varphi_{n,d}$. Par le premier lemme, il existe donc $\Bbbk$ de caractéristique 0 telle que $\Bbbk \models \varphi_{n,d}$. Par la complétude de la théorie des corps algébriquement clos, on a que $\mathbb{C} \models \varphi_{n,d}$.

Conjecture de la Jacobienne (1939).

C'est une question encore ouverte. On reçoit plein de preuves fausses.

Définition 15.7. Soit $P : \mathbb{C}^n \to \mathbb{C}^n$ un polynôme. Son *jacobien* est le déterminant de la matrice jacobienne

$$\operatorname{Jac} P = \left| \left(\frac{\partial P_i}{\partial x_j} \right)_{1 \leq i \leq n, 1 \leq j \leq n} \right|.$$

C'est un polynôme.

Proposition 15.2. Si P est injectif sur $\mathbb{C}^n$ alors P est localement injectif. Et donc, pour tout x (théorème des fonctions implicites), $\operatorname{Jac}(P)$ n'est jamais nul, d'où $\operatorname{Jac} P$ est un polynôme constant non nul.

Remarque 15.4 (Conjecture (problème 16 de la liste de Steve Smale))**.** En caractéristique 0, on a $\operatorname{Jac} P$ non nul implique P injectif.

Remarque 15.5. En caractéristique p, c'est faux : $P(x) := x - x^p$ est non-inversible et $P'(x) = 1 - px = 1$.

Exemple 15.11. ▷ Avec $n = 1$ et $d = 1$, on considère

$$\begin{aligned} P : \mathbb{C} &\longrightarrow \mathbb{C} \\ x &\longmapsto P(x) := ax + b. \end{aligned}$$

On a $\operatorname{Jac} P = a$ et, $a \neq 0$ implique P injectif.

▷ Avec $n = 1$ et $d = 2$, on considère

$$\begin{aligned} P : \mathbb{C} &\longrightarrow \mathbb{C} \\ x &\longmapsto P(x) := ax^2 + bx + c. \end{aligned}$$

On a, si $\operatorname{Jac} P = 2ax + b$ non nul, alors $a = 0$ et $b \neq 0$. C'est le cas précédent !

▷ Avec $n = 2$ et $d = 1$, on considère

$$\begin{aligned} P : \mathbb{C}^2 &\longrightarrow \mathbb{C}^2 \\ x &\longmapsto P(x, y) := (ax + by + c, dx + ey + f). \end{aligned}$$

On a $\operatorname{Jac} P = \begin{vmatrix} a & b \\ d & e \end{vmatrix} = ae - bd$. On a $\operatorname{Jac} P$ non nul implique $ae - bd \neq 0$ ce qui implique que le système

$$\begin{cases} ax + bj + c = 0 \\ dx + ej + f = 0 \end{cases}$$

est inversible, donc la conjecture est vrai.

On a montré quelques résultats partiels :

- ▷ pour $d \leq 2$ en 1980 ;
- ▷ pour $d \leq 3$ dans le cas général.

Troisième partie

Projet fonctionnel.

Introduction.

Le coups se décompose en deux parties :

- une partie *pratique* ($\sim 2/3$ du cours) avec de la programmation OCaml en binôme (fouine, puis pieuvre) ;
- une partie *théorique* : avec du λ-calcul, typage et lien avec la logique.

Ce document contiendra les notes de cours pour la partie théorique. D'autres documents pour la partie projet sont en ligne sur mon site (transformation CPS, et optimisations associées à la β-réduction).

Parfois, des références aux chapitres du cours de théorie de la programmation seront fait, ce sont des liens cliquables qui mènent vers le PDF du chapitre en question, par exemple :

THPROG [Chapitre 0].

16. Le λ-calcul pur.

Le λ-calcul a trois domaines proches :

- ▷ la *calculabilité*, avec l'équivalence entre machines de Turing et λ-expression (vue en FDI) ;
- ▷ la *programmation fonctionnelle* (vue en **THPROG** [Chapitre 6] avec le petit langage FUN) ;
- ▷ la *théorie de la démonstration* (vue dans la suite de ce cours).

On se donne un ensemble infini $\mathcal{V}$ de variables notées $x, y, z, \ldots$ Les *termes* (du λ-calcul) ou λ-termes sont définis par la grammaire

$$M, N, \ldots ::= \lambda x.\, M \mid M\ N \mid x.$$

La construction $\lambda x.\, M$ s'appelle l' *abstraction* ou λ-*abstraction*. Elle était notée $\texttt{fun}\, x \to M$ en cours de théorie de la programmation.

Notation.
- ▷ On notera $M\ N\ P$ pour $(M\ N)\ P$.
- ▷ On notera $\lambda xyz.M$ pour $\lambda x.\lambda y.\lambda z M$ (il n'y a pas lieu de mettre des parenthèses ici, vu qu'il n'y a pas d'ambigüités).
- ▷ On notera $\lambda x.\, M\ N$ pour $\lambda x.\, (M\ N)$. ***Attention***, c'est différent de $(\lambda x.\, M)\ N$.

16.1 Liaison et α-conversion.

> **Remarque 16.1** (Liaison). Le « λ » est un lieur. Dans $\lambda y.\, x\ y$, la variable y est *liée* mais pas x (la variable x libre). On note $\mathcal{VL}(M)$ l'ensemble des variables libres de M, définie par induction sur M (il y a *3* cas).

Remarque 16.2 (α-conversion).
On note $=_\alpha$ la relation d' *α-conversion*. C'est une relation binaire sur les λ-termes fondée sur l'idée de renommage des abstractions ***en évitant la capture de variables libres*** :

$$\lambda x.\, x\, y =_\alpha \lambda t.\, x\, t \neq_\alpha \lambda x.x\, x.$$

Ainsi $\lambda x.\, M =_\alpha \lambda z.\, M'$ où M' est obtenu en remplaçant x par z ***là où il apparaît libre*** et ***à condition que*** $z \notin \mathcal{V}\ell(M)$. Ceci, on peut le faire partout.

Lemme 16.1. La relation $=_\alpha$ est une relation d'équivalence. Si $M =_\alpha N$ alors $\mathcal{V}\ell(M) = \mathcal{V}\ell(N)$.

Par convention, on peut identifier les termes modulo $=_\alpha$. On pourra donc toujours dire

« considérons $\lambda x.\, M$ où $x \notin E$ [...] »

avec E un ensemble ***fini*** de variables.

Ceci veut dire qu'on notera

$$M = N \text{ pour signifier que } M =_\alpha N.$$

16.2 La β-réduction.

Définition 16.1 (β-réduction). On définit la relation de β-réduction sur les λ-termes, notée $\to_\beta$ ou $\to$, définie par les règles d'inférences :

$$\frac{}{(\lambda x.\, M)\, N \to_\beta M[N/x]} \qquad \frac{M \to_\beta M'}{\lambda x.\, M \to_\beta \lambda x.\, M'}$$

$$\frac{M \to_\beta M'}{M\, N \to_\beta M'\, N} \qquad \frac{N \to_\beta N'}{M\, N \to_\beta M\, N'}$$

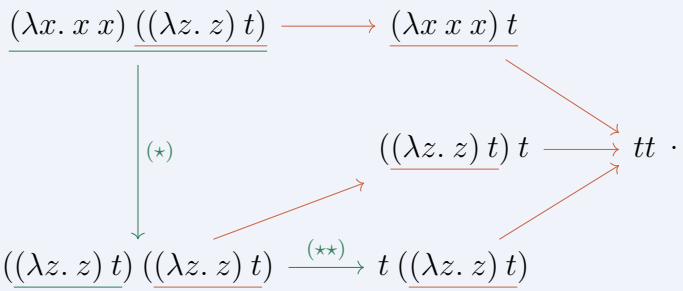
où $M[^N/_x]$ est la substitution de x par N dans M (on le défini ci-après).

Définition 16.2. Un terme de la forme $(\lambda x.\, M)\, N$ est appelé un *redex* (pour *reducible expression*) ou *β-redex*. Un terme M est une *forme normale* s'il n'existe pas de N tel que $M \to_\beta N$.

Remarque 16.3. La relation $\to_\beta$ n'est pas terminante :

$$\Omega := (\lambda x.\, x\, x)\, (\lambda y.\, y\, y) \to_\beta (\lambda y.\, y\, y)\, (\lambda y.\, y\, y) =_\alpha \Omega.$$

Exemple 16.1.



Un pas de β-réduction peut :

- ▷ dupliquer un terme (*c.f.* $(\star)$) ;
- ▷ laisser un redex inchangé (*c.f.* $(\star\star)$) ;
- ▷ faire disparaître un redex (qui n'est pas celui que l'on contracte) :

$$(\lambda x.\, u)((\lambda z.\, z)\, t) \ \to_\beta u\ ;$$

- ▷ créer de nouveaux redex :

$$(\lambda x.\, x\, y)\, (\lambda z.\, z) \to_\beta (\lambda z.\, z)\, y.$$

16.3 Substitutions.

Exemple 16.2. Le terme $\lambda xy.\ x$ c'est une « fonction fabriquant des fonctions constantes » au sens où

$$(\lambda xy.\ x)M \to_\beta \lambda y.\ M,$$

à condition que $y \notin \mathcal{VL}(M)$. On doit cependant α-renommer pour éviter la capture :

$$\begin{array}{c} (\lambda xy.x)\ (\lambda t.\ y) \not\to_\beta \lambda y.\ (\lambda t.\ y) \\ \qquad\quad \wr\| \\ (\lambda xy'.x)\ (\lambda t.\ y) \to_\beta \lambda y'.\ (\lambda t.\ y). \end{array}$$

Définition 16.3. On procède par induction, il y a trois cas :

- $y[N/x] := \begin{cases} N & \text{si } y = x \\ y & \text{si } y \neq x \end{cases}$
- $(M_1\ M_2)[N/x] := (M_1[N/x])\ (M_2[N/x])$
- $(\lambda y.\ M)[N/x] := \lambda y.\ (M[N/x])$ ***à condition que*** $y \notin \mathcal{VL}(N)$ ***et*** $y \neq x$.

Lemme 16.2 (Gymnastique des substitutions)**.** Pour $y \notin \mathcal{VL}(R)$,

$$(P[Q/y])[R/x] = (P[R/x])[Q[R/x]/y].$$

Lemme 16.3. Si $M \to_\beta M'$ alors $\mathcal{VL}(M') \subseteq \mathcal{VL}(M)$.

16.4 Comparaison λ-calcul et FUN.

En λ-calcul, on a une règle

$$\frac{M \to_\beta M'}{\lambda x.\ M \to_\beta \lambda x.\ M'.}$$

Cette règle n'existe pas en FUN (ni en fouine) car on traite les fonctions comme des valeurs. Et, en FUN, les trois règles suivantes sont

mutuellement exclusives :

$$\frac{}{(\lambda x.\, M)\, N \to_\beta M[N/x]} \qquad \frac{M \to_\beta M'}{M\, N \to_\beta M'\, N} \qquad \frac{N \to_\beta N'}{M\, N \to_\beta M\, N'}$$

car on attend que N soit une ***valeur*** avant de substituer.

En FUN (comme en fouine), pour l'exemple 16.1, on se limite à n'utiliser que les flèches rouges.

La relation $\to_\beta$ est donc « plus riche » que $\to_{\mathsf{FUN}}$. En FUN, on a une *stratégie de réduction* : on a au plus un redex qui peut être contracté. On n'a pas de notion de valeur en λ-calcul pur. Le « *résultat d'un calcul* » est une forme normale.

16.5 Exercice : les booléens.

On définit

$$\mathbf{T} := \lambda xy.\, x \qquad \mathbf{F} := \lambda xy.\, y.$$

Ainsi, pour tout M (si $y \notin \mathcal{V}\ell(M)$),

$$\mathbf{T}\, M \to \lambda y.\, M \qquad \mathbf{F}\, M \to \lambda y.\, y =: \mathbf{I}.$$

La construction `if` b `then` M `else` N se traduit en $b\, M\, N$.

Le « non » booléen peut se définir par :

- ▷ $\mathbf{not} := \lambda b.\, b\, \mathbf{F}\, \mathbf{T} = \lambda b.\, b\, (\lambda xy.\, y)\, (\lambda tu.\, t)$;
- ▷ $\mathbf{not}' := \lambda b.\, \lambda xy.\, byx$.

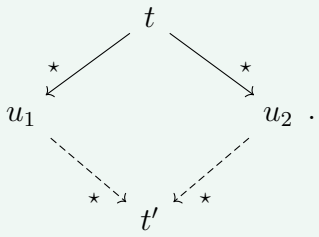
La première version est plus abstraite, la seconde est « plus électricien ». On a deux formes normales ***différentes***.

De même, on peut définir le « et » booléen :

- ▷ $\mathbf{and} := \lambda b_1.\, \lambda b_2.\, b_1\, (b_2\, \mathbf{T}\, \mathbf{F})\, \mathbf{F}$;
- ▷ $\mathbf{and}' := \lambda b_1.\, \lambda b_2.\, \lambda xy.\, b_1\, (b_2\, x\, y)\, y$.

16.6 Confluence de la β-réduction.

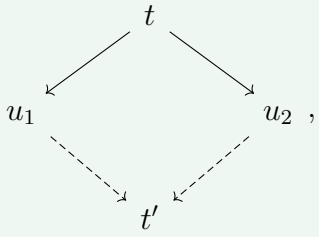
Définition 16.4 (Rappel, *c.f.* **THPROG** [Chapitre 10])**.** On dit que $\to$ est *confluente* en $t \in A$ si, dès que $t \to^\star u_1$ et $t \to^\star u_2$ il existe t' tel que $u_1 \to^\star t'$ et $u_2 \to^\star t'$.



Les flèches en pointillés représentent l'existence.

On dit que $\to$ est *confluente* si $\to$ est confluente en tout $a \in A$.

La propriété du diamant correspond au diagramme ci-dessous :



c'est-à-dire si $t \to u_1$ et $t \to u_2$ alors il existe t' tel que $u_1 \to t'$ et $u_2 \to t'$.

La confluence pour $\to$, c'est la propriété du diamant pour $\to^\star$. On sait déjà que la β-réduction n'a pas la propriété du diamant (certains chemins de l'exécution sont plus longs), mais on va montrer qu'elle est confluente.

Définition 16.5. On définit la relation de *réduction parallèle*, notée $\rightrightarrows$, par les règles d'inférences suivantes :

$$\frac{}{x \rightrightarrows x} \qquad \frac{M \rightrightarrows M'}{\lambda x.\, M \rightrightarrows \lambda x.\, M'}$$

$$\frac{M \rightrightarrows M' \quad N \rightrightarrows N'}{M\, N \rightrightarrows M'\, N'} \qquad \frac{M \rightrightarrows M' \quad N \rightrightarrows N'}{(\lambda x.\, M)\, N \rightrightarrows M'[N'/x]}$$

Lemme 16.4. La relation $\rightrightarrows$ est réflexive.

Lemme 16.5. Si $\mathscr{R} \subseteq \mathscr{S}$ alors $\mathscr{R}^\star \subseteq \mathscr{S}^\star$. De plus, $(\mathscr{R}^\star)^\star = \mathscr{R}^\star$.

Lemme 16.6. Les relations $\to^\star$ et $\rightrightarrows^\star$ coïncident.

Preuve.
- ▷ On a $\to^\star \subseteq \rightrightarrows^\star$ car cela découle de $\to \subseteq \rightrightarrows$ par induction sur $\to$ en utilisant la réflexivité de $\rightrightarrows$.
- ▷ On a $\rightrightarrows^\star \subseteq \to^\star$ car cela découle de $\rightrightarrows \subseteq \to^\star$. En effet, on montre que pour tout M, M' si $M \rightrightarrows M'$ alors $M \to^\star M'$, par induction sur $\rightrightarrows$. Il y a 4 cas.
  - Pour $x \rightrightarrows x$, c'est immédiat.
  - Pour l'abstraction, on suppose $M \rightrightarrows M'$ alors par induction $M \to^\star M'$, et donc $\lambda x.\, M \to^\star \lambda x.\, M'$ par induction sur $M \to^\star M'$.
  - Pour l'application, c'est plus simple que pour la précédente.
  - Pour la substitution, supposons $M \rightrightarrows M'$ et $N \rightrightarrows N$. On déduit par hypothèse d'induction $M \to^\star M'$ et $N \to^\star N'$. Et, par induction sur $M \to^\star M'$, on peut montrer que $(\lambda x.\, M)\, NN \to^\star (\lambda x.\, M')\, N$. Puis, par induction sur $N \to^\star N'$, on montre $(\lambda x.\, M')\, N \to^\star (\lambda x.\, M')\, N'$. Enfin, par la règle de β-réduction, on a $(\lambda x.\, M')N' \to M'[N'/x]$. On rassemble tout pour

obtenir :

$$(\lambda x.\, M)\, N \to^\star M'[N'/x].$$

□

On est donc ramené à montrer que $\rightrightarrows^\star$ a la propriété du diamant. Or $\rightrightarrows$ a la propriété du diamant, ce que l'on va montrer en TD.

Lemme 16.7. Si $M \rightrightarrows M'$ alors $N \rightrightarrows N'$ implique $M[N/x] \rightrightarrows M'[N'/x]$.

Preuve. Par induction sur $M \rightrightarrows M'$, il y a 4 cas. On ne traite que le cas de la 4ème règle. On suppose donc $M = (\lambda y.\, P)\, Q$ avec $y \notin \mathcal{VL}(N)$ et $y \neq x$. On suppose aussi $P \rightrightarrows P'$, $Q \rightrightarrows Q'$ et $M' = P'[Q'/y]$. On suppose de plus $N \rightrightarrows N'$. Par hypothèse d'induction, on a $P[N/x] \rightrightarrows P'[N'/x]$ et $Q[N/x] \rightrightarrows Q'[N'/x]$. On applique la 4ème règle d'inférence définissant $\rightrightarrows$ pour déduire

$$\underbrace{(\lambda y.\, (P[N/x]))}_{\substack{\| \\ (\lambda y.\, P)[N/x] \\ \text{car } x \neq y}}(Q[N/x]) \rightrightarrows (P'[N'/x])[Q'[N'/x]/y] = (P'[Q'/y])[N'/x]$$

par le lemme de gymnastique des substitutions et car $y \notin \mathcal{VL}(N') \subseteq \mathcal{VL}(N)$ et car $N \to^\star N'$. □

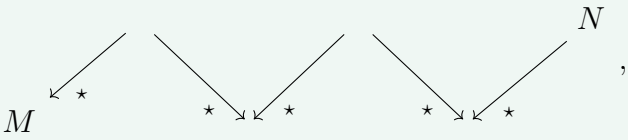
Proposition 16.1. La relation $\rightrightarrows$ a la propriété du diamant.

Preuve. Vu en TD. □

Corollaire 16.1. On a la confluence de $\to_\beta$.

Définition 16.6. La *β-équivalence*, ou *β-convertibilité* est la plus petite relation d'équivalence contenant $\to_\beta$. On la note $=_\beta$.

Si l'on a



alors $M =_\beta N$.

Proposition 16.2. Tout λ-terme est β-équivalent à au plus une forme normale.

Preuve. Si $M =_\beta N$ et M, N sont des formes normales, alors par confluence il existe P tel que $M \to^\star P$ et $N \to^\star P$. On a donc que $M = N = P$. □

Remarque 16.4 (Conséquences).
- ▷ Deux normales distinctes (au sens de $=_\alpha$) ne sont pas β-convertibles.
- ▷ Si on a un λ-terme qui diverge et qui a une forme normale, par exemple $(\lambda x.\ y)\ \Omega$, alors on peut toujours « revenir » sur la forme normale.

17. Le λ-calcul simplement typé.

Dans ce chapitre, on va parler de *typage*. Ceci permet de « stratifier » les λ-termes. En effet, pour l'instant, tous les termes se ressemblent.

17.1 Définition du système de types.

Définition 17.1. On se donne un ensemble de *types de base*, notés X, Y, Z, ... Les types simples sont donnés par la grammaire suivante :

$$A, B, C ::= \boldsymbol{X} \mid A \to B.$$

Il n'y a donc que deux « types » de types : les types de base, et les types fonctions. Il n'y a donc pas de type `unit`, `bool`, ... En effet, ceci demanderai d'ajouter des constantes `()`, `true`, `false`, *etc* dans la grammaire du λ-calcul (et ceci demanderai ensuite d'ajouter des règles de typage supplémentaire). On verra en TD comment typer $\mathbf{T}$ et $\mathbf{F}$ comme défini au chapitre précédent.

Par convention, on notera $A \to B \to C$ pour $A \to (B \to C)$.

Définition 17.2. On définit une *hypothèse de typage* comme un couple variable-type (x, A) noté $x : A$.

Définition 17.3. Un *environnent de typage*, noté Γ, Γ', *etc* est un dictionnaire sur $(\mathscr{V}, \mathsf{Types})$, *c.f.* cours de Théorie de la Programmation. On notera $\Gamma(x) = A$ lorsque Γ associe x à A. On définit

le *domaine* de Γ comme

$$\mathrm{dom}(\Gamma) := \{x \mid \exists A,\ \Gamma(x) = A\}.$$

On note aussi $\Gamma, x : A$ l'extension de Γ avec $x : A$ si $x \notin \mathrm{dom}(\Gamma)$.

Définition 17.4. On définit la *relation de typage*, notée $\Gamma \vdash M : A$ (« sous les hypothèses Γ, le λ-terme M a le type A ») par les règles d'inférences suivantes :

$$\Gamma(x) = A \ \frac{}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$

$$x \notin \mathrm{dom}(\Gamma) \ \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\ M : A \to B}$$

Dans cette dernière règle, on peut toujours l'appliquer modulo α-conversion (il suffit d'α-renommer x dans $\lambda x.\ M$).

Exemple 17.1. On peut omettre le « $\emptyset$ » avant « $\vdash$ ».

$$\dfrac{\dfrac{\dfrac{\dfrac{}{f : \boldsymbol{X} \to \boldsymbol{X}, z : \boldsymbol{X} \vdash f : \boldsymbol{X} \to \boldsymbol{X}} \quad \dfrac{\dfrac{}{f : \boldsymbol{X} \to \boldsymbol{X}, z : \boldsymbol{X} \vdash f : \boldsymbol{X} \to \boldsymbol{X}} \quad \dfrac{}{f : \boldsymbol{X} \to \boldsymbol{X}, z : \boldsymbol{X} \vdash z : \boldsymbol{X}}}{f : \boldsymbol{X} \to \boldsymbol{X}, z : \boldsymbol{X} \vdash f\ z : \boldsymbol{X}}}{f : \boldsymbol{X} \to \boldsymbol{X}, z : \boldsymbol{X} \vdash f\ (f\ z) : \boldsymbol{X}}}{f : \boldsymbol{X} \to \boldsymbol{X} \vdash \lambda z.\ f\ (f\ z) : \boldsymbol{X} \to \boldsymbol{X}}}{\vdash \lambda f.\ \lambda z.\ f\ (f\ z) : (\boldsymbol{X} \to \boldsymbol{X}) \to \boldsymbol{X} \to \boldsymbol{X}}$$

Exemple 17.2.

$$\dfrac{\dfrac{\dfrac{}{a, X, t : X \to Y \vdash t : X \to Y} \quad \dfrac{}{a, X, t : X \to Y \vdash a : X}}{a, X, t : X \to Y \vdash t\ a : Y}}{a : X \vdash \lambda t.\ t\ a : (X \to Y) \to Y}\ .$$

17.2 Propriétés de la relation de typage.

Lemme 17.1 (Lemme administratif).

- ▷ Si $\Gamma \vdash M : A$ alors $\mathcal{V}\ell(M) \subseteq \mathrm{dom}(\Gamma)$.
- ▷ *Renforcement* : si $\Gamma, x : B \vdash M : A$ et $x \notin \mathcal{V}\ell(M)$ alors $\Gamma \vdash M : A$.
- ▷ *Affaiblissement* : si $\Gamma \vdash M : A$ alors, pour tout B et tout $x \notin \mathrm{dom}(\Gamma)$ alors $\Gamma, x : B \vdash A$.
- ▷ *Contraction* : si $\Gamma, x : B, y : B \vdash M : A$ alors $\Gamma, x : B \vdash M[x/y] : A$ □

Proposition 17.1 (Préservation du typage). Si $\Gamma \vdash M : A$ et $M \to_\beta M'$ alors $\Gamma \vdash M' : A$.

Preuve. On procède comme en **THPROG** [Chapitre 7] avec le lemme suivant.

> **Lemme 17.2.** Si $\Gamma, x : A \vdash M : B$ et $\Gamma \vdash N : A$ alors $\Gamma \vdash M[N/x] : B$.

□

17.3 Normalisation forte.

Définition 17.5. Un λ-terme M est dit *fortement normalisant* ou *terminant* si toute suite de β-réductions issue de M conduit à une forme normale. Autrement dit, il n'y a pas de divergence issue de M.

Théorème 17.1. Si M est typage (il existe Γ, A tels que $\Gamma \vdash M : A$) alors M est fortement normalisant.

Remarque 17.1 (Quelques tentatives de preuves ratées.). ▷
Par induction sur M ? Non.

- ▷ Par induction sur la relation de typage $\Gamma \vdash M : A$? Non (le cas de l'application pose problème car deux cas de β-réductions).

Pour démontrer cela, on utilise une méthode historique : les *candidats de réductibilité*.

Définition 17.6 (Candidat de réductibilité)**.** Soit A un type simple. On associe à A un ensemble de λ-termes, noté $\mathscr{R}_A$ appelé *candidats de réductibilité* (ou simplement *candidats*) associé à A, défini par induction sur A de la manière suivante :

- ▷ $\mathscr{R}_{\boldsymbol{X}} := \{M \mid M \text{ est fortement normalisant}\}$;
- ▷ $\mathscr{R}_{A\to B} := \{M \mid \forall N \in \mathscr{R}_A, M\ N \in \mathscr{R}_B\}$.

L'idée est la suivante :

$$\begin{array}{c} M \text{ typable} \\ \Gamma \vdash M : A \end{array} \quad \rightsquigarrow \quad M \in \mathscr{R}_A \quad \rightsquigarrow \quad M \text{ fortement normalisant.}$$

Remarque 17.2 (Rappel sur le PIBF, *c.f.* **THPROG** [Chapitre 10])**.** Le principe d'induction bien fondé nous dit qu'une relation $\mathscr{R}$ est terminante ssi pour tout prédicat $\mathscr{P}$ sur E vérifie que si

$$\forall x \in E \left((\forall y, x\ \mathscr{R}\ y \implies \mathscr{P}(y)) \implies \mathscr{P}(x) \right)$$

alors $\forall x \in E, \mathscr{P}(x)$.

Proposition 17.2. Soit A un type simple. On a :

CR 1. Pour tout $M \in \mathscr{R}_A$, M est fortement normalisant.

CR 2. Pour tout $M \in \mathscr{R}_A$, si $M \to_\beta M'$ alors $M' \in \mathscr{R}_A$.

CR 3. Pour tout M *neutre* (c-à-d, M n'est pas une λ-abstraction), si $\forall M', M \to_\beta M' \implies M' \in \mathscr{R}_A$ alors $M \in \mathscr{R}_A$.

Preuve. On montre la conjonction de **CR 1**, **CR 2** et **CR 3** par induction sur A. Il y a deux cas.

- Cas $\boldsymbol{X}$ un type simple.

  CR 1. C'est vrai par définition.

  CR 2. Si M est fortement normalisant, et $M \to_\beta M'$ alors $M' \in \mathscr{R}_{\boldsymbol{X}}$.

  CR 3. Si M est neutre et si on a que « pour tout M' tel que $M \to_\beta M'$ alors $M' \in \mathscr{R}_{\boldsymbol{X}}$ » alors c'est l'induction bien fondée pour $\to_\beta$ sur $\mathscr{R}_{\boldsymbol{X}}$.

- Cas $A \to B$ un type flèche.

  CR 1. Soit $M \in \mathscr{R}_{A\to B}$. Supposons que M diverge :

  $$M \to_\beta M_1 \to_\beta M_2 \to_\beta \cdots .$$

  On a observé que $x \in \mathscr{R}_A$ pour une variable x arbitraire (conséquence de **CR 3** pour A). Par définition de $\mathscr{R}_{A\to B}$, $M\,x \in \mathscr{R}_B$. Par **CR 1** pour B, on a que $M\,x$ est fortement normalisant. Or, $M\,x \to_\beta M_1\,x$ car $M \to_\beta M_1$. On construit ainsi une divergence dans $\mathscr{R}_B$ à partir de $M\,x$:

  $$M\,x \to_\beta M_1\,x \to_\beta M_2\,x \to_\beta \cdots .$$

  C'est absurde car cela contredit que $M\,x$ fortement normalisant.

  CR 2. Soit $M \in \mathscr{R}_{A\to B}$ et $M \to M'$. Montrons que $M' \in \mathscr{R}_{A\to B}$, *i.e.* pour tout $N \in \mathscr{R}_A$ alors $M'\,N \in \mathscr{R}_B$. Soit donc $N \in \mathscr{R}_A$. On sait que $M\,N \in \mathscr{R}_B$ (car $M \in \mathscr{R}_{A\to B}$). Et comme $M \to_\beta M'$ alors $M\,N \to_\beta M'\,N$ et, par **CR 2** pour B, on a $M'\,N \in \mathscr{R}_B$. On a donc montré $\forall N \in \mathscr{R}_{A\to B}, M'N \in \mathscr{R}_B$ autrement dit, $M' \in \mathscr{R}_{A\to B}$.

  CR 3. Soit M neutre tel que $\forall M', M \to_\beta M' \implies M' \in \mathscr{R}_{A\to B}$. Montrons que $M \in \mathscr{R}_{A\to B}$. On sait que $\to_\beta$ est

terminante sur $\mathscr{R}_A$ (par **CR 1** pour A). On peut donc montrer que $\forall N \in \mathscr{R}_A$, $M\,N \in \mathscr{R}_B$ par induction bien fondée sur $\to_\beta$. On a les hypothèses suivantes :

- hypothèse 1 : pour tout M' tel que $M \to_\beta M'$ alors $M' \in \mathscr{R}_{A\to B}$;
- hypothèse d'induction bien fondée : pour tout N' tel que $N \to_\beta N'$ que $M\,N' \in \mathscr{R}_B$.

On veut montrer $M\,N \in \mathscr{R}_B$. On s'appuie sur **CR 3** pour B et cela nous ramène à montrer que, pour tout P tel que $M\,N \to_\beta P$ est $P \in \mathscr{R}_B$. On a trois cas possibles pour $M\,N \to_\beta P$.

- Si $M = \lambda x.\, M_0$ et $P = M_0[{}^N/_x]$ qui est exclu car M est neutre.
- Si $P = M'\,N$ alors par hypothèse 1 $M' \in \mathscr{R}_{A\to B}$ et donc $M'\,N \in \mathscr{R}_B$.
- Si $P = M\,N'$ alors, par par hypothèse d'induction bien fondée, $M\,N' \in \mathscr{R}_B$.

□

Lemme 17.3. Soit M tel que $\forall N \in \mathscr{R}_A, M[{}^N/_x] \in \mathscr{R}_B$. Alors $\lambda x.\, M \in \mathscr{R}_{A\to B}$.

Preuve. On procède comme pour **CR 3** pour $A \to B$. □

Lemme 17.4. Supposons $x_1 : A_1, \dots, x_k : A_k \vdash M : A$. Alors, pour tout $N_1, \dots, N_k$ tel que $N_i \in \mathscr{R}_{A_i}$, on a

$$M[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] \in \mathscr{R}_A.$$

On note ici la *substitution simultanée* des x_i par des N_i dans M. C'est ***n'est pas*** la composition des substitutions.

Preuve. Par induction sur la relation de typage, il y a trois cas.

- Si on a utilisé la règle de l'axiome, c'est que M est une variable : $M = x_i$ et $A = A_i$. Soit $N_i \in \mathscr{R}_{A_i}$ alors $M[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] = N_i \in \mathscr{R}_A$.
- Si on a utilisé la règle de l'application, c'est que M est une application : $M = M_1\, M_2$ et $M_1 : B \to A$ et $M_2 : B$. On a :

  $$M[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] = M_1[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}]M_2[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}].$$

  On conclut en appliquant les hypothèses d'inductions : $M_1[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] \in \mathscr{R}_{B\to A}$ et $M_2[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] : \mathscr{R}_B$.
- Si on a utilisé la règle de l'abstraction, c'est que $M = \lambda y.M_0$ avec $y \notin \{x_1, \ldots, x_k\} \cup \mathscr{V}\ell(N_1) \cup \cdots \cup \mathscr{V}\ell(N_k)$. Supposons que $x_1 : A_1, \ldots, x_k : A_k \vdash \lambda y.\ M_0 : A \to B$. Alors nécessairement $x_1 : A_1, \ldots, x_k : A_k, y : A \vdash M_0 : B$. Par hypothèse d'induction, on a que pour tout $N_i \in \mathscr{R}_{A_i}$ on a

  $$M_0[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}][{}^{N}/_{y}] = M_0[{}^{N_1\ \cdots\ N_k\ N}/_{x_1\ \cdots\ x_k\ y}] \in \mathscr{R}_B.$$

  Par le lemme précédent, on déduit que

  $$(\lambda y.M_0)[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] = \lambda y.(M_0[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}]) \in \mathscr{R}_{A\to B}.$$

□

Corollaire 17.1. Si $\Gamma \vdash M : A$ alors $M \in \mathscr{R}_A$.

17.4 Extension : le λ-calcul typé avec $\times$ et 1.

En ajoutant les couples et *unit*, il faut modifier quatre points.

Syntaxe. $M, N ::= \lambda x.\ M \mid M\ N \mid x \mid (M, N) \mid \star \mid \pi_1\ M \mid \pi_2\ M$

β-réduction.

$$\frac{M \to_\beta M'}{(M, N) \to_\beta (M', N)} \qquad \frac{N \to_\beta N'}{(M, N) \to_\beta (M, N')}$$

$$\frac{}{\pi_1\ (M, N) \to_\beta M} \quad \frac{}{\pi_2\ (M, N) \to_\beta N}.$$

Types. $A, B ::= X \mid A \to B \mid A \times B \mid \mathbf{1}$

Typage.

$$\frac{}{\star : \mathbf{1}} \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A \times B}$$

$$\frac{\Gamma \vdash P : M \times N}{\Gamma \vdash \pi_1\ P : M} \qquad \frac{\Gamma \vdash P : M \times N}{\Gamma \vdash \pi_2\ P : N}\ .$$

18. Introduction à la théorie de la démonstration.

18.1 Formules et preuves.

Définition 18.1. On se donne un ensemble de *variables propositionnelles*, qui seront notées X, Y, Z, *etc.* L'ensemble des *formules* est défini par la grammaire :

$$A, B ::= X \mid A \Rightarrow B.$$

Cet ensemble de formules s'appelle le « *fragment implicatif de la logique propositionnelle intuitionniste* ».

Cela peut sembler inhabituel car, généralement, on commence par introduire $\neg$, $\vee$ et $\wedge$, car on a en tête les booléens.

Définition 18.2. Les *séquents*, notés $\Gamma \vdash A$, un couple formé de Γ une ***liste*** de formules, et A une formule. La liste Γ est une *liste d'hypothèses*. On notera Γ, A la notation pour l'extension de la liste.

Définition 18.3. On *prouve* (*dérive*) les séquents à l'aides des *règles de déduction* (*d'inférence*) :

$$_{A \in \Gamma}\ \frac{}{\Gamma \vdash A}\ \mathsf{Ax} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow_{\mathsf{I}} \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \Rightarrow_{\mathsf{E}}.$$

On les appelle, dans l'ordre, règle de l'*axiome*, règle de l'*introduction de* $\Rightarrow$ et règle de l'*élimination de* $\Rightarrow$. Il s'agit des *règles de déduction naturelle pour le fragment implicatif de la logique propositionnelle intuitionniste*.

Définition 18.4. Le séquent $\Gamma \vdash A$ est *prouvable* s'il existe une *preuve* (*dérivation*) ayant $\Gamma \vdash A$ à la racine et des axiomes aux feuilles. La formule A est *prouvable* si $\vdash A$ l'est.

Exemple 18.1.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{X \Rightarrow Y, X \vdash X \Rightarrow Y}\,\mathsf{Ax} \quad \dfrac{}{X \Rightarrow Y, X \vdash X}\,\mathsf{Ax}}{X \Rightarrow Y, X \vdash Y}\,\Rightarrow_{\mathsf{E}}}{X \Rightarrow Y \vdash X \Rightarrow Y}\,\Rightarrow_{\mathsf{I}}}{\vdash (X \Rightarrow Y) \Rightarrow (X \Rightarrow Y)}\,\Rightarrow_{\mathsf{I}} .$$

On écrit généralement des « preuves génériques », en utilisant A, B plutôt que X, Y.

Exemple 18.2.

$$\dfrac{\dfrac{\dfrac{}{(A \Rightarrow A) \Rightarrow B \vdash (A \Rightarrow A) \Rightarrow B}\,\mathsf{Ax} \quad \dfrac{\dfrac{}{(A \Rightarrow A) \Rightarrow B, A \vdash A}\,\mathsf{Ax}}{(A \Rightarrow A) \Rightarrow B \vdash A \Rightarrow A}\,\Rightarrow_{\mathsf{I}}}{(A \Rightarrow A) \Rightarrow B \vdash B}\,\Rightarrow_{\mathsf{E}}}{\vdash ((A \Rightarrow A) \Rightarrow B) \Rightarrow B}\,\Rightarrow_{\mathsf{I}} .$$

18.2 Et en Rocq ?

En Rocq, un objectif de preuve

$$\left.\begin{array}{l} H_1 : A_1 \\ H_2 : A_2 \\ H_3 : A_3 \\ \vdots \\ \hline\hline \end{array}\right\} \Gamma \qquad A$$

correspond au séquent

$$\Gamma \vdash A.$$

Chaque tactique correspond à des opérations sur l'arbre de preuve. On construit « au fur et à mesure » l'arbre de preuve montrant $\Gamma \vdash A$. Voici ce que quelques tactiques Rocq font.

$$\frac{??}{\Gamma, A, B, A \vdash A} \xrightarrow[\texttt{assumption}]{} \frac{}{\Gamma, A, B, A \vdash A}\,\mathsf{Ax}$$

$$\frac{??}{\Gamma \vdash C} \xrightarrow[\texttt{assert } A]{} \frac{\dfrac{??}{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B} \quad \frac{??}{\Gamma \vdash A}\,\Rightarrow_{\mathsf{E}}$$

$$\frac{??}{\Gamma \vdash C} \xrightarrow[\texttt{cut } A]{} \frac{??}{\Gamma \vdash A \Rightarrow B} \quad \frac{??}{\Gamma \vdash A}\,\Rightarrow_{\mathsf{E}}$$

$$\frac{??}{\Gamma \vdash C} \xrightarrow[\texttt{apply } H]{} \frac{\dfrac{}{\Gamma, A \Rightarrow B \vdash A \Rightarrow B}\,\mathsf{Ax}}{\Gamma, \underbrace{A \Rightarrow B}_{H} \vdash B} \quad \frac{??}{\Gamma, A \Rightarrow B \vdash A}\,\Rightarrow_{\mathsf{E}}$$

$$\frac{??}{\Gamma \vdash B \Rightarrow C} \xrightarrow[\texttt{intro}]{} \frac{\dfrac{??}{\Gamma, B \vdash C}}{\Gamma \vdash B \Rightarrow C}\,\Rightarrow_{\mathsf{I}}$$

18.3 Liens avec le λ-calcul simplement typé : *correspondance de Curry-Howard.*

Les règles de typage du λ-calcul correspondent au règles d'inférences du fragment implicatif :

$$\frac{}{\Gamma \vdash x : A}\ {\scriptstyle x : A \in \Gamma} \quad \longleftrightarrow \quad \frac{}{\Gamma \vdash A}\ {\scriptstyle A \in \Gamma}\ \mathsf{Ax}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \to B} \quad \longleftrightarrow \quad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow_{\mathsf{I}}$$

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \quad \longleftrightarrow \quad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow_{\mathsf{I}}$$

.

En retirant les λ-termes en bleu (incluant les « : »), et en changeant $\to$ en $\Rightarrow$, on obtient exactement les mêmes règles.

Si on sait que $\Gamma \vdash x : A$ alors, en effaçant les parties en bleu, on obtient une preuve de $\tilde{\Gamma} \vdash A$.

Inversement, on se donne une preuve de $\Gamma \vdash A$. On se donne des variables x_i pour transformer $\Gamma = A_1, \dots, A_k$ en $\hat{\Gamma} = x_1 : A_1, \dots, x_k : A_k$. Par induction sur $\Gamma \vdash A$, on montre qu'il existe un λ-terme tel que $\hat{\Gamma} \vdash M : A$. On a trois cas.

- Pour $\Rightarrow_{\mathsf{I}}$, par induction, si $\hat{\Gamma}, x = A \vdash M : B$, on déduit $\hat{\Gamma} \vdash \lambda x.\, M : A \to B$.
- Pour $\Rightarrow_{\mathsf{I}}$, par induction, si $\hat{\Gamma} \vdash M : A \to B$ et $\hat{\Gamma} \vdash N : A$, on déduit $\hat{\Gamma} \vdash M\ N : B$.
- Pour Ax, on sait $A \in \Gamma$ donc il existe x tel que $x : A \in \hat{\Gamma}$, et on conclut $\hat{\Gamma} \vdash x_i : A$.

On a les propriétés suivantes pour la relation de déduction :

- *affaiblissement* : si $\Gamma \vdash B$ (implicitement « est prouvable ») alors $\Gamma, A \vdash B$;
- *contraction* : si $\Gamma, A, A \vdash B$ alors $\Gamma, A \vdash B$;
- *renforcement* si $\Gamma, A \vdash B$ alors $\Gamma \vdash B$ *à condition qu'on n'utilise pas l'axiome avec l'hypothèse A (celle là uniquement, les A intermédiaires ne posent pas de problèmes) pour déduire B.*

- ▷ *échange* ; su $\Gamma, A, B, \Gamma' \vdash C$ alors $\Gamma, B, A, \Gamma' \vdash C$.

C'est analogue aux propriétés du typage en λ-calcul.

En effet, la propriété de renforcement, très imprécise dans sa formulation logique, est simplement : si $\hat{\Gamma}, x : A \vdash M : B$ alors $\hat{\Gamma} \vdash M : B$ *à condition que* $x \notin \mathcal{V}\ell(M)$.

Si on veut prouver ces propriétés (au lieu d'utiliser la correspondance de Curry-Howard), on ferait une induction sur la preuve du séquent qui est donné.

La règle

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ \mathsf{aff}$$

est *admissible*. En effet, si on sait prouver les prémisses (ici, $\Gamma \vdash B$) alors on sait prouver la conclusion (ici, $\Gamma, A \vdash B$). Ceci dépend fortement de la logique que l'on utilise.

18.4 Curry-Howard du côté calcul : les coupures.

Typons un redex :

$$\frac{\dfrac{\Gamma, x : A \vdash M : B}{\lambda x.\, M : A \to B}\Rightarrow_{\mathsf{I}} \qquad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x.\, M)\, N : M} \Rightarrow_{\mathsf{E}}.$$

Oui, c'est exactement la même chose que la tactique `assert` en Rocq.

Définition 18.5. Une *coupure* est un endroit dans la preuve où il y a un usage d'une règle d'élimination ($\Rightarrow_{\mathsf{E}}$) dont la prémisse principale est déduite à l'aide d'une règle d'introduction ($\Rightarrow_{\mathsf{I}}$) pour le même connecteur logique.

Remarque 18.1. Ici, on n'a qu'un seul connecteur logique, $\Rightarrow$, mais cela s'étend aux autres connecteurs que l'on pourrait ajouter. La *prémisse principale* est, par convention, la première.

On peut *éliminer une coupure* pour $\Rightarrow$, c'est-à-dire transformer une preuve (*c.f.* contracter un β-redex) en passant de

$$\dfrac{\dfrac{\dfrac{\delta}{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B}\Rightarrow_{\mathsf{I}} \qquad \dfrac{\delta'}{\Gamma \vdash A}}{\Gamma \vdash B}\Rightarrow_{\mathsf{E}}$$

à

$$\dfrac{\delta[\delta'/A]}{\Gamma \vdash B}$$

où l'on note $\delta[\delta'/A]$ la preuve obtenue en remplaçant dans δ chaque usage de l'axiome avec A par δ'.

On a le même séquent en conclusion (*c.f.* préservation du typage en λ-calcul simplement typé).

La correspondance de Curry-Howard c'est donc :

Types	$\longleftrightarrow$	Formules
Programmes	$\longleftrightarrow$	Preuves
β-réduction	$\longleftrightarrow$	Élimination d'une coupure
Programmation	$\longleftrightarrow$	***Logique***

18.5 Faux, négation, consistance.

On modifie nos formules :

$$A, B ::= X \mid A \Rightarrow B \mid \bot$$

et on ajoute la règle d'élimination du $\bot$ (il n'y a pas de règle d'introduction) :

$$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}\bot_{\mathsf{E}}.$$

La négation $\neg A$ est une notation pour $A \Rightarrow \bot$. On peut donc prouver le séquent $\vdash A \Rightarrow \neg\neg A$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A, \neg A \vdash \neg A}\,\mathsf{Ax} \qquad \dfrac{}{A, \neg A \vdash A}\,\mathsf{Ax}}{A, \neg A \vdash \bot}\,\Rightarrow_{\mathsf{E}}}{A \vdash \neg\neg A}\,\Rightarrow_{\mathsf{I}}}{\vdash A \Rightarrow \neg\neg A}\,\Rightarrow_{\mathsf{I}} .$$

Théorème 18.1 (Élimination des coupures)**.** Si $\Gamma \vdash A$ (est prouvable) alors il existe une preuve ***sans coupure*** de $\Gamma \vdash A$.

Preuve. *c.f.* TD. □

Remarque 18.2 (Lien avec normalisation forte en λ-calcul simplement typé)**.** Ici, on veut la normalisation faible (« il existe une forme normale . . . »). On ne peut pas appliquer *stricto sensu* la normalisation forte pour le λ-calcul simplement typé car le système de type contient $\bot$.

Lemme 18.1. Une preuve sans coupure de $\vdash A$ en logique intuitionniste se termine (à la racine) nécessairement par une règle d'introduction.

Preuve. Par induction sur $\vdash A$. Il y a 4 cas.

- ▷ Ax : Absurde car $\Gamma = \emptyset$.
- ▷ $\Rightarrow_{\mathsf{I}}$: OK
- ▷ $\Rightarrow_{\mathsf{E}}$: On récupère une preuve de $\vdash B \Rightarrow A$ qui termine (par induction) par une introduction $\Rightarrow_{\mathsf{I}}$. Absurde car c'est une coupure.
- ▷ $\bot_{\mathsf{E}}$: On récupère une preuve de $\bot$ qui termine par une règle d'induction : impossible.

□

Corollaire 18.1 (Consistance de la logique)**.** Il n'y a pas de preuve de $\vdash$ en logique propositionnelle intuitionniste dans le fragment avec $\Rightarrow$ et $\bot$.

Preuve. S'il y en avait une, il y en aurait une sans coupure, qui se termine par une règle d'introduction, impossible. □

18.6 Et en Rocq ? (partie 2)

On étend les formules avec $\forall$, $\exists$, $\neg$, $\vee$, $\wedge$, *etc.* Les preuves sont des λ-termes. En effet, dans une preuve de $\vdash X \to X \to X$ on peut écrire

$$\texttt{exact (fun}\, x\, y \texttt{ => }\, x\texttt{)},$$

pour démontrer le séquent.

Le mot clé `Qed` prend le λ-terme construit par la preuve et calcule M' sous forme normale tel que $M \to_\beta^\star M'$. La logique de Rocq est *constructive*. C'est-à-dire qu'une preuve de $A \Rightarrow B$ c'est une fonction qui transforme une preuve de A en une preuve de B. Après avoir appelé `Qed`, il est possible d'extraire le λ-terme construit en un programme OCaml, Haskell, *etc.*

18.7 Logique intuitionniste *vs* logique classique.

Dans la logique que l'on a considérée (TD), on a deux règles d'introduction pour $\vee$:

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee_{\mathsf{I}}^{\mathsf{g}} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee_{\mathsf{I}}^{\mathsf{g}}.$$

Lorsqu'on a une preuve de $A \vee B$, on a, soit une preuve de A, soit une preuve de B. Ce n'est pas une preuve « il est impossible de ne pas avoir A et B ». La logique est *constructive*.

On rappelle que $\neg A := A \to \bot$, et que l'on se donne la règle d'élimination de $\bot$:

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot_{\mathsf{E}}.$$

La *logique classique* est la logique obtenue à l'aide de l'ajout d'une des règles suivantes :

Tiers exclu. $\dfrac{}{\Gamma \vdash A \vee \neg A}$ tiers exclu

Ce n'est pas constructif : on ne sait pas si l'on a une preuve de $\Gamma \vdash A$ ou de $\Gamma \vdash \neg A$.

Absurde. $\dfrac{}{\Gamma \vdash (\neg\neg A) \Rightarrow A}$ absurde

C'est mieux : ici, on n'a pas de $\vee$.

Loi de Peirce. $\dfrac{}{\Gamma \vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$ peirce

C'est encore mieux : ici, on n'a pas de $\vee$, ni de $\bot$, mais c'est plus subtil.

En choisissant un de ces axiomes, on a la même notion de prouvabilité.

Exercice 18.1. Montrons que absurde implique tiers exclu (au sens de « on peut dériver l'un dans le système incluant l'autre »).

$$\dfrac{\dfrac{}{\Gamma \vdash (\neg\neg(A \implies \neg A)) \Rightarrow (A \vee \neg A)}\ \text{absurde} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \neg(A \vee \neg A) \vdash \neg(A \vee \neg A)}\ \text{ax}}{\Gamma, \neg(A \vee \neg A) \vdash A \vee \neg A}\ \vee_{\mathsf{I}}^{\mathsf{d}}}{\Gamma, \neg(A \vee \neg A) \vdash A \vee \neg A}\ \Rightarrow_{\mathsf{I}}}{\Gamma \vdash \neg\neg(A \vee \neg A)}\ \Rightarrow_{\mathsf{E}}}{\Gamma \vdash A \vee \neg A}$$

où la branche droite est

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \neg(A \vee \neg A), A \vdash A}\ \text{ax}}{\Gamma, \neg(A \vee \neg A), A \vdash A \vee \neg A}\ \vee_{\mathsf{I}}^{\mathsf{g}}}{\Gamma, \neg(A \vee \neg A), A \vdash \bot}\ \Rightarrow_{\mathsf{E}}}{\Gamma, \neg(A \vee \neg A) \vdash \neg A}\ \Rightarrow_{\mathsf{I}}$$

Théorème 18.2 (Glivenko). Une formule A est prouvable en logique classique si et seulement si $\neg\neg A$ est prouvable en logique intuitionniste.

Preuve. Ressemble un peu à la traduction par continuation des programmes fouine. □

Corollaire 18.2. La logique classique est consistante ssi la logique intuitionniste est consistante.

Preuve. Si $\vdash \bot$ en logique intuitionniste alors $\vdash \bot$ en logique classique. Si $\vdash \bot$ en logique classique, alors $\neg\neg\bot$ en intuitionniste, et on peut en déduire une preuve de $\vdash \bot$ en intuitionniste :

$$\dfrac{\dfrac{\text{par hyp.}}{\vdash (\bot \to \bot) \to \bot} \qquad \dfrac{\dfrac{}{\bot \vdash \bot}\,\mathsf{ax}}{\vdash \bot \to \bot}\,\Rightarrow_{\mathsf{I}}}{\vdash \bot}\,\Rightarrow_{\mathsf{E}}.$$

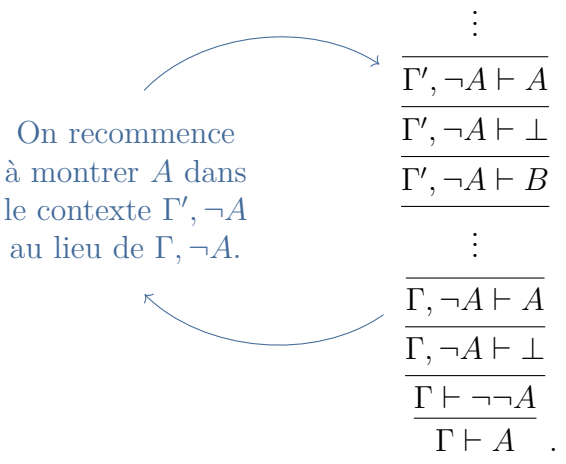
□

18.8 Logique classique et Curry-Howard : intuition opérationnelle.

On cherche à compléter la correspondance de Curry-Howard :

Types	$\longleftrightarrow$	Formules
Programmes	$\longleftrightarrow$	Preuves
β-réduction	$\longleftrightarrow$	Élimination d'une coupure
Principes classiques	$\longleftrightarrow$	???
Programmation	$\longleftrightarrow$	***Logique***

Avec la preuve par l'absurde, on peut « recommencer dans un contexte différent ».



$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma', \neg A \vdash A}}{\Gamma', \neg A \vdash \bot}}{\Gamma', \neg A \vdash B} \qquad \dfrac{\dfrac{\dfrac{\vdots}{\Gamma, \neg A \vdash A}}{\Gamma, \neg A \vdash \bot}}{\dfrac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}}\ .$$

S'autoriser les principes classiques, c'est savoir utiliser les exceptions : si ça explose, je peux le rattraper. En effet, l'élimination du $\bot$ fait penser à un opérateur comme `raise : exn -> 'a`, et la construction `try...with...` pour pouvoir « sauter » à des endroits du programme, et dévier le flot du programme.

19. Le λ-calcul polymorphe.

On étend la grammaire des types :

$$A, B ::= X \mid A \to B \mid \forall X\ A.$$

Ici, les X ne sont plus les types de base (noté $\boldsymbol{X}$ précédemment), mais ce sont des *variables de types*.

Pour mieux refléter les notations de la littérature, on aura plutôt :

$$A, B ::= \alpha \mid A \to B \mid \forall \alpha\ A.$$

Le « $\forall \alpha\ A$ » est une structure qui lie, $\forall$ est un lieur. Il y a donc une notion de variables libres d'un type $\mathscr{Vl}(A)$, d'α-conversion, et de substitution. Par exemple,

$$\mathscr{Vl}(\forall \alpha\, \forall \beta\, (\alpha \to \beta \to \gamma \to \alpha)) = \{\gamma\}$$

et

$$\forall \alpha\ \alpha \to \alpha =_\alpha \forall \beta\ \beta \to \beta.$$

19.1 Typage, version implicite.

Aux trois règles des types simples, on ajoute les deux règles ci-dessous :

$$\alpha \notin \mathscr{Vl}(\Gamma)\ \frac{\Gamma \vdash M : A}{\Gamma \vdash M : \forall \alpha\ A}\ \mathscr{T}_{\mathsf{g}} \qquad \frac{\Gamma \vdash M : \forall \alpha\ A}{\Gamma \vdash M : A[{}^{B}/_{\alpha}]}\ \mathscr{T}_{\mathsf{i}}.$$

Les règles correspondent à la *généralisation* et à l'*instantiation*.

Et, ***on ne change pas les λ-termes***.

19.2 Typage, version explicite.

On change les λ-termes :

$$M, N ::= x \mid M\ N \mid \lambda x : A.\ M \mid \Lambda\alpha.\ M \mid M\ A.$$

La construction $\Lambda\alpha. M$ est une *abstraction de types*, et la construction $M\ A$ est l'*application d'un terme à un type*. On note ***explicitement*** le type « d'entrée » de l'abstraction.

On change les règles de β-réduction :

$$\frac{}{(\Lambda\alpha.\ M)\ A \to_\beta M[{}^{A}/_{\alpha}]} \qquad \frac{M \to_\beta M'}{\Lambda\alpha.\ M \to_\beta \Lambda\alpha.\ M'}$$

$$\frac{M \to_\beta M'}{M\ A \to_\beta M'\ A.}$$

On change également les règles de typage :

$$\Gamma(x) = A\ \frac{}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$

$$x \notin \mathrm{dom}(\Gamma)\ \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.\ M : A \to B}$$

$$\alpha \notin \mathcal{V}\ell(\Gamma)\ \frac{\Gamma \vdash M : A}{\Gamma \vdash \Lambda\alpha.\ M : \forall\alpha\ A} \qquad \frac{\Gamma \vdash M : \forall\alpha\ A}{\Gamma \vdash M\ B : A[{}^{B}/_{\alpha}]}$$

Lemme 19.1. On a $\Gamma \vdash M : A$ dans le système explicite ssi il existe M' dans le système explicite avec $\Gamma \vdash M' : A$ (explicite) et M est « l'effacement » de M'.

- ▷ La représentation implicite est plus utilisée dans les langages comme OCaml, où l'on doit inférer les types.
- ▷ La représentation explicite correspond plus aux langages comme Rocq, avec un lien plus naturel avec la logique.

Exemple 19.1. Soit le λ-terme non typé $\underline{2} := \lambda f.\, \lambda z.\, f\ (f\ z)$. Comment le typer en version explicite ?

1.
$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{f : \alpha \to \alpha, z : \alpha \vdash f\ (f\ z) : \alpha}}{f : \alpha \to \alpha \vdash \lambda z : \alpha.\, f\ (f\ z) : \alpha \to \alpha}}{\emptyset \vdash \lambda f : \alpha \to \alpha.\, \lambda z : \alpha.\, f\ (f\ z) : (\alpha \to \alpha) \to \alpha \to \alpha}}{\emptyset \vdash \Lambda\alpha.\, \lambda f : \alpha \to \alpha.\, \lambda z : \alpha.\, f\ (f\ z) : \forall\alpha\ (\alpha \to \alpha) \to \alpha \to \alpha.}$$

2.
$$\dfrac{\dfrac{\dfrac{\vdots}{f : \forall\alpha\ \alpha \to \alpha, z : \beta \vdash f\ \beta\ (f\ \beta\ z) : \beta}}{f : \forall\alpha\ \alpha \to \alpha \vdash \lambda z : \beta.\, f\ \beta\ (f\ \beta\ z) : \beta \to \beta}}{\emptyset \vdash \lambda f : \forall\alpha\ \alpha \to \alpha.\, \lambda z : \beta.\, f\ \beta(f\ \beta\ z) : (\forall\alpha\ \alpha \to \alpha) \to \beta \to \beta.}$$

Exemple 19.2. On suppose que l'on a les couples. Comment compléter le séquent

$$y : \beta, z : \gamma \vdash \lambda f : \boxed{?}.\, (f\ y, f\ z) : \boxed{?}.$$

- ▷ Pour les types simples, on ne peut pas si $\beta \neq \gamma$.
- ▷ Mais, avec polymorphisme, on a :

$$y : \beta, z : \gamma \vdash \lambda f : \forall\alpha\ \alpha \to \delta.\, (f\ y, f\ z) : (\forall\alpha\ \alpha \to \delta) \to \delta \times \delta.$$

Exemple 19.3.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}}{\emptyset \vdash \lambda x.\, x : \alpha \to \alpha}}{\emptyset \vdash \lambda x.\, x : \forall \alpha\, \alpha \to \alpha}}{\emptyset \vdash \lambda x.\, x : (\forall \beta\, \beta) \to (\forall \delta\, \delta).}$$

En effet, $(\forall \beta\, \beta) \to (\forall \beta\, \beta) =_\alpha (\forall \beta\, \beta) \to (\forall \delta\, \delta)$.

Exemple 19.4 (Ouch !).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}}{x : \alpha \vdash x : \forall \alpha\, \alpha}}{x : \alpha \vdash x : \beta}}{\emptyset \vdash \lambda x.\, x : \alpha \to \beta}}{\emptyset \vdash \lambda x.\, x : \forall \alpha\, \forall \beta\, \alpha \to \beta.}$$

NON ! On a $\alpha \in \mathcal{V}\ell(x : \alpha)$!

19.3 Point de vue logique sur le polymorphisme, système F.

On se place du point de vue Curry-Howard. On ajoute deux règles de déduction supplémentaire :

$$\alpha \notin \mathcal{V}\ell(\Gamma)\ \dfrac{\Gamma \vdash A}{\Gamma \vdash \forall \alpha\, A}\ \forall_{\mathsf{I}} \qquad \dfrac{\Gamma \vdash \forall \alpha\, A}{\Gamma \vdash A[^B/_\alpha]}\ \forall_{\mathsf{E}}.$$

On a, encore, un possibilité d'éliminer les coupures : si $\alpha \notin \mathcal{V}\ell(\Gamma)$,

$$\dfrac{\dfrac{\dfrac{\delta}{\Gamma \vdash A}\ \forall_{\mathsf{I}}}{\Gamma \vdash \forall \alpha\, A}\ \forall_{\mathsf{E}}}{\Gamma \vdash A[^B/_\alpha]} \quad \rightsquigarrow \quad \dfrac{\delta[^B/_\alpha]}{\Gamma \vdash A[^B/_\alpha]}$$

Ceci correspond, par correspondance de Curry-Howard, a une β-réduction :

$$(\Lambda\alpha\, M)\, B \to_\beta M[^B/_\alpha].$$

On a aussi un théorème de normalisation forte.

Théorème 19.1. Si $\Gamma \vdash M : A$ en λ-calcul polymorphe, alors M est fortement normalisant.

Pour démontrer ce théorème, on utilise encore les candidats de réductibilité (il ne faut définir que $\mathscr{R}_{\forall\alpha\, A}$), mais la preuve est bien plus complexe. En effet, les types polymorphes ont un grand pouvoir expressif (*c.f.* la section suivante).

Le système que l'on a s'appelle *système F*.[1] C'est de la logique du second ordre : on peut quantifier sur les variables propositionnelles. On quantifie donc sur des ensembles de valeurs au lieu de se limiter uniquement à quantifier sur les valeurs.

19.4 Représentation des connecteurs logiques.

On peut représenter les connecteurs logiques différemment, sans les ajouter explicitement, mais uniquement avec le fragment contenant $\to$ et $\forall$.

- On peut représenter $\bot := \forall\alpha\,\alpha$. En effet, on a la correspondance suivante :
$$\frac{\Gamma \vdash \forall\alpha\,\alpha}{\Gamma \vdash A}\,\forall_{\mathsf{I}} \quad \rightsquigarrow \quad \frac{\Gamma \vdash \bot}{\Gamma \vdash A}\,\bot_{\mathsf{E}}.$$
- On peut représenter $\top := \forall\alpha\,\alpha \to \alpha$.
- On peut représenter $A \wedge B := \forall\alpha\,(A \to B \to \alpha) \to \alpha$:

1. À ne pas confondre avec *Station F*.

$$\dfrac{\dfrac{\dfrac{}{\Gamma, c : A \to B \to \alpha \vdash c : A \to B \to \alpha} \quad \dfrac{\Gamma \vdash M : A}{\Gamma, c : A \to B \to \alpha \vdash M : A}}{\Gamma, c : A \to B \to \alpha \vdash c\, M : B \to \alpha} \quad \dfrac{\Gamma \vdash N : B}{\Gamma, c : A \to B \to \alpha \vdash M : A}}{\dfrac{\Gamma, c : A \to B \to \alpha \vdash c\, M\, N\, \alpha}{\dfrac{\Gamma \vdash \lambda c.\, c\, M\, N : (A \to B \to \alpha) \to \alpha}{\Gamma \vdash \lambda c.\, c\, M\, N : \forall \alpha\, (A \to B \to \alpha) \to \alpha}}}$$

d'où l'introduction du $\wedge$. Pour l'élimination, si

$$\Gamma \vdash M : \forall \alpha\, (A \to B \to \alpha) \to \alpha,$$

alors $\Gamma \vdash M\ (\lambda x.\, \lambda y.\, x) : A$ et $\Gamma \vdash M\ (\lambda x.\, \lambda y.\, y) : B$.

19.5 Le λ-calcul polymorphe, côté programmation.

« Généricité ». On peut avoir plusieurs types en même temps. Par exemple, une fonction de tri a un type

$$\forall \alpha\, (\alpha \to \alpha \to \texttt{bool}) \to \texttt{list}\ \alpha \to \texttt{list}\ \alpha.$$

« Paramétricité ». Lorsque l'on a une fonction $f : (\forall \alpha : \alpha) \to A$, la fonction n'inspecte pas son argument. Elle ne fait que le dupliquer, le passer à d'autres fonctions, le rejeter, mais elle ne peut pas voir les données sous-jacentes. (On exclue ici les fonctions types `Obj.magic`).

Quelques conséquences sur les formes normales :

- ▷ Il n'y a qu'une seule forme normale ayant un type $\forall \alpha\, \alpha \to \alpha$.
- ▷ Il n'y a que deux formes normales ayant un type $\forall \alpha \alpha \to \alpha \to \alpha$: $\lambda u.\, \lambda v.\, u$ et $\lambda u.\, \lambda v.\, v$.

On a aussi quelques « théorèmes gratuits », comme par exemple : si $\texttt{f} : \alpha \to \beta$ est croissante vis à vis de $\texttt{ordre}_\alpha : \alpha \to \alpha \to \texttt{bool}$ et $\texttt{ordre}_\beta : \beta \to \beta \to \texttt{bool}$, alors

$$\texttt{map f (sort ordre}_\alpha\ \texttt{l)} = \texttt{sort ordre}_\beta\ \texttt{(map f l)}.$$

19.6 Polymorphisme et inférence de types.

Pour l'inférence de types, on se place dans la représentation implicite du typage polymorphique. On s'intéresse à la question suivante : pour M un λ-terme donné, existe-t-il Γ et A tels que $\Gamma \vdash M : A$?

Cependant, il y a un problème :

Théorème 19.2 (1992). L'inférence de types pour le typage polymorphique implicite est ***indécidable***.

Ceci conclut l'étude du Système F. On va maintenant s'intéresser à une variant de système F développée en 1978, le λ-*calcul avec schéma de types*. On note ρ le schéma de types définis par :

$$A ::= \alpha \mid A \to B \qquad \rho ::= A \mid \forall \alpha\, \rho.$$

Remarque 19.1. Il n'y a pas de types de la forme $(\forall \alpha\, \alpha \to \alpha) \to (\forall \beta\, A)$. La quantification est ***prénexe***.

Les termes sont donnés par la grammaire :

$$M ::= x \mid \lambda x.\, M \mid M\ N \mid \texttt{let}\, x = M\ \texttt{in}\ N.$$

Pour le système de types, on considère Γ un dictionnaire sur (variables du λ-calcul x, schémas de types ρ). La relation de typage est notée $\Gamma \vdash M : A$ où A est un type ***simple***. On la définit par induction à l'aide des règles :

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \to B}$$

$$\vec{\alpha} \cap \mathcal{V}\ell(\Gamma) = \emptyset \ \frac{\Gamma \vdash M : A \qquad \Gamma, x : \forall \vec{\alpha}\, A \vdash N : B}{\Gamma \vdash \texttt{let}\, x = M\ \texttt{in}\ N : B} \qquad \Gamma(x) = \forall \vec{\alpha}\, A \ \frac{}{\Gamma \vdash x : A[\vec{B}/\vec{\alpha}]}$$

Ici, on note $\forall \vec{\alpha}\, A$ pour $\forall \alpha_1 \cdots \forall \alpha_n\, A$ avec possiblement aucun $\forall$. De même, la substitution $A[\vec{B}/\vec{\alpha}]$ se fait simultanément.

La règle d'***instantiation*** de système F se retrouve partiellement dans la règle de l'axiome. La règle d'***généralisation*** de système F se retrouve uniquement dans la règle du `let ... = ... in ...`, et uniquement après le `in`.

Exemple 19.5. Le terme

$$\texttt{let}\ f = \lambda x.\, x\ \texttt{in}\ f$$

a pour type $\beta \to \beta$, sans généralisation !

$$\dfrac{\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}}{\emptyset \vdash \lambda x.\, x : \alpha \to \alpha} \qquad \dfrac{}{f : \forall \alpha\ \alpha \to \alpha \vdash f : \beta \to \beta}}{\emptyset \vdash \texttt{let}\ f = \lambda x.\, x\ \texttt{in}\ f : \beta \to \beta}\ .$$

En OCaml, lorsqu'on essaie de typer

```
(fun z -> z) (fun x -> x)
```

on se rend compte que l'on n'a pas un type `'a -> 'a`, mais

```
'_weak1 -> '_weak1.
```

Théorème 19.3. Il existe un algorithme correct et complet pour l'inférence de types pour les schémas de types.

Pour ce faire, on fait comme en **THPROG** [Chapitre 7], où l'on part de M, on génère des contraintes, et à l'aide de l'algorithme d'unification, on trouve le type « le plus général ».

20. Types linéaires.

Le but des types linéaires est de ***contrôler l'usage des ressources***. Avec un tel système de typage, on pourrait :

- ▷ désallouer une zone mémoire, fermer un canal/un fichier ; on se limite à un seul usage : les deux termes suivants ne sont pas typables
  - $(\texttt{free}\ x, \texttt{free}\ x)$
  - $(\lambda y.\ \lambda z.\ (\texttt{free}\ y, \texttt{free}\ z))\ x\ x$
- ▷ l'initialisation d'une zone mémoire ; au moins un usage
- ▷ si on sait au plus un usage, alors on peut désallouer après un usage.

On se donne la syntaxe suivante.

- ▷ Les booléens sont :

$$b ::= \texttt{true} \mid \texttt{false}.$$

- ▷ Les termes sont :

$$\begin{aligned} M, N ::= {} & \eta b \mid \eta(M, N) \mid \eta\lambda x.\ M \\ & \mid \texttt{if}\ b\ \texttt{then}\ M\ \texttt{else}\ N \mid \texttt{let}(x, y) = M\ \texttt{in}\ N \mid M\ N \\ & \mid x \mid \texttt{let}\ x = M\ \texttt{in}\ N. \end{aligned}$$

- ▷ Les usages sont :

$$\eta ::= \mathsf{lin} \mid \mathsf{un}.$$

Les *usages* représentent une utilisation *linéaire* (lin) ou non restrictive (un*restricted*). Un usage lin est utilisé exactement ***une*** fois.

On s'autorise le filtrage sur les couples au lieu de `fst` M et `snd` M car, dans le cas linéaire, on ne pourrait utiliser qu'une des deux composantes… zut !

On accepte, par exemple, $\mathsf{lin}(\texttt{lintrue}, \mathsf{lin}\texttt{false})$ ou $\mathsf{un}(\texttt{lintrue}, \texttt{unfalse})$.

Les types sont définis par la grammaire

$$T, U ::= \texttt{bool} \mid A \to B \mid A * B \qquad A, B ::= \eta\, T.$$

Les T, U sont des *prétypes*, les A, B sont les *types*.

La séparation dans la grammaire permet de rejeter une expression comme $\mathsf{un}(\mathsf{lin}\,\texttt{bool} * \mathsf{un}\,\texttt{bool})$.

On se donne les règles de typages pour cette version linéaire du λ-calcul.

Première tentative.

On propose la règle pour les couples linéaires :

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : \mathsf{lin}(A * B)}.$$

Cependant, on ne mets pas de restrictions sur A : on peut avoir $A = \mathsf{lin}T$ ou $A = \mathsf{un}T$… Zut !

Seconde tentative.

On propose un règle pour les couples

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B \qquad \eta(A) \qquad \eta(B)}{\Gamma \vdash (M, N) : \eta(A * B)},$$

où l'on définit le prédicat $\eta(A)$ par :

- ▷ si $\eta = \mathsf{un}$ alors A ne peut pas être $\mathsf{lin}T$;
- ▷ si $\eta = \mathsf{lin}$ alors A peut être quelconque.

Puis, on se donne une règle pour le filtrage

$$\frac{\Gamma \vdash M : \eta(A_1 * A_2) \qquad \Gamma, x : A_1, y : A_2 \vdash N : B}{\Gamma \vdash \texttt{let}(x, y) = M \ \texttt{in} \ N : B} .$$

Ici, η n'a pas d'importance. On peut donc typer

$$\texttt{let}(x, y) = \mathsf{un}(\mathsf{lintrue}, \mathsf{unfalse}) \ \texttt{in} \ \mathsf{lin}(x, x),$$

mais on ne veut pas pouvoir le faire !

Tentative finale.

Définition 20.1. On définit $\Gamma_1 \circ \Gamma_2$ est défini comme « $\Gamma_1 \cup \Gamma_2$ » à ceci près que :

- si $\Gamma_1(x) = \mathsf{lin}T$ alors $x \notin \mathrm{dom}(\Gamma_2)$;
- si $\Gamma_2(x) = \mathsf{lin}T$ alors $x \notin \mathrm{dom}(\Gamma_1)$;
- si $x \in \mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2)$ alors il existe T tel que $\Gamma_1(x) = \Gamma_2(x) = \mathsf{un}T$.

Les vraies règles de typage sont :

$$\frac{\Gamma_1 \vdash M : A \qquad \Gamma_2 \vdash N : B \qquad \eta(A) \qquad \eta(B)}{\Gamma_1 \circ \Gamma_2 \vdash (M, N) : \eta(A * B)}$$

$$\frac{\Gamma_1 \vdash M : \eta(A * B) \qquad \Gamma_2, x : A, y : B \vdash N : C}{\Gamma_1 \circ \Gamma_2 \vdash \texttt{let}(x, y) = M \ \texttt{in} \ N : C}$$

$$\frac{\mathsf{un}(\Gamma)}{\Gamma, x : A \vdash x : A} \qquad \frac{\mathsf{un}(\Gamma)}{\Gamma \vdash \eta b : \eta\texttt{bool}}$$

$$\frac{\Gamma_1 \vdash M : \eta\texttt{bool} \qquad \Gamma_2 \vdash N_1 : A \qquad \Gamma_2 \vdash N_2 : A}{\Gamma_1 \circ \Gamma_2 \vdash \texttt{if} \ M \ \texttt{then} \ N_1 \ \texttt{else} \ N_2 : A}$$

$$\frac{\Gamma, x : A \vdash M : B \qquad \eta(\Gamma)}{\Gamma \vdash \eta\lambda x.\, M : \eta(A \to B)}$$

$$\frac{\Gamma_1 \vdash M : \eta(A \to B) \qquad \Gamma_2 \vdash N : A}{\Gamma_1 \circ \Gamma_2 \vdash M\, N : B}$$

On note ici $\mathsf{un}(\Gamma)$ (et éventuellement $\eta(\Gamma)$) le prédicat disant que Γ le contient des types que de la forme $\mathsf{un}T$.

Dans le cas de l'axiome, on assure que l'on n'a pas de variables inutilisées. Dans le cas de l'abstraction, on contrôle l'usage des variables qui se retrouvent dans la clôture associée à $\lambda x.\, M$.

Exercice 20.1. Peut-on trouver un terme ayant pour type

1. $\mathsf{un}(\mathtt{unbool} \to \mathtt{linbool})$?
2. $\mathsf{un}(\mathtt{linbool} \to \mathtt{unbool})$?

On ne peut pas proposer :

1. $\mathsf{un}(\lambda x.\, x)$ car on n'a pas de lin
2. $\mathsf{un}(\lambda x.\, \mathtt{unfalse})$ car on n'utilise pas le x.

Oui, on peut :

1. $\mathsf{un}(\lambda x.\, \mathtt{linfalse})$
2. $\mathsf{un}(\lambda x.\, \mathtt{if}\ x\ \mathtt{then\ unfalse\ else\ untrue})$

Remarque 20.1 (Obligation de linéarité)**.** On a des règles de « bonne formation » : $\eta(A)$ et $\eta(\Gamma)$. La décomposition $\Gamma_1 \circ \Gamma_2$ correspond à un aiguillage.

Les types linéaires correspondent à des *types substructurels*. Ceci est relié à la *logique linéaire*.

Lemme 20.1. ▷ *Affaiblissement.* Si $\Gamma \vdash M : A$ alors pour $x \notin \mathrm{dom}(\Gamma)$ on a $\Gamma, x : \mathsf{un}T \vdash M : A$.

▷ *Contraction.* Si $\Gamma, x : \mathsf{un}T, y : \mathsf{un}T \vdash M : B$ alors on a $\Gamma, x : \mathsf{un}T \vdash M[x/y] : B$. Ceci n'est pas vrai avec lin.