Fonctions récursives primitives

Nous définissons et étudions les fonctions qui expriment les calculs « faciles » de l'arithmétique.

Dans ce chapitre, notre point de vue est *sémantique*, c'est-à-dire qu'on va étudier des objets de la « réalité » : les entiers, les nombres premiers, l'addition, ...
Dans le chapitre suivant, nous ferons le lien avec le langage $\mathcal{L}_0$ de l'arithmétique de Peano, qui sera le point de vue *syntaxique* où tout s'exprime par des suites de symboles. Le lien entre les deux sera le théorème de représentabilité : tout ce qui est vrai pour les fonctions récursives primitives peut être prouvé dans le langage de l'arithmétique de Peano.

1. Les fonctions récursives primitives

Les fonctions récursives primitives sont les fonctions qui sont « faciles à calculer » au sens du chapitre précédent, c'est-à-dire qu'on peut en calculer les valeurs à l'aide d'un algorithme n'utilisant que des boucles « pour ».
Ce sont des fonctions $f : \mathbb{N}^p \to \mathbb{N}$. Elles sont définies à partir de trois fonctions de base et deux opérations élémentaires. Cependant, cet ensemble contient toutes les fonctions et opérations usuelles sur les entiers : addition, multiplication, puissance, ...

1.1. Définition

On note $\mathcal{F}_p = \{f : \mathbb{N}^p \to \mathbb{N}\}$ et $\mathcal{F} = \bigcup_{p \geqslant 1} \mathcal{F}_p$. On rappelle que $\mathbb{N}^p = \mathbb{N} \times \cdots \times \mathbb{N}$. On notera souvent le p-uplet $(x_1, \ldots, x_p) \in \mathbb{N}^p$ par $\mathbf{x}$. Les éléments de $\mathcal{F}$ sont donc des fonctions de p variables (où p peut varier d'une fonction à l'autre).
La définition des fonctions récursives repose sur trois fonctions de base et deux opérations.
Fonctions de base

- Z : la fonction nulle

$$\begin{aligned} Z : \mathbb{N} &\longrightarrow \mathbb{N} \\ x &\longmapsto 0 \end{aligned}$$

- S : la fonction successeur

$$\begin{aligned} S : \mathbb{N} &\longrightarrow \mathbb{N} \\ x &\longmapsto x + 1 \end{aligned}$$

- P_i^p : les projections (pour $1 \leqslant i \leqslant p$).

$$\begin{aligned} P_i^p : \mathbb{N}^p &\longrightarrow \mathbb{N} \\ (x_1, \ldots, x_p) &\longmapsto x_i \end{aligned}$$

Opération de composition

- Données : $f_1, \dots, f_q : \mathbb{N}^p \to \mathbb{N}$ et $g : \mathbb{N}^q \to \mathbb{N}$.
- Composition : la fonction $g(f_1, \dots, f_q) : \mathbb{N}^p \to \mathbb{N}$ définie par

$$\boxed{\mathbf{x} \mapsto g(f_1(\mathbf{x}), \dots, f_q(\mathbf{x}))}$$

Opération de récurrence

- Données :
  — $g : \mathbb{N}^p \to \mathbb{N}$ (initialisation)
  — $h : \mathbb{N}^{p+2} \to \mathbb{N}$ (étape de récurrence)
- Nouvelle fonction $f : \mathbb{N}^{p+1} \to \mathbb{N}$ définie par :

$$\boxed{f(\mathbf{x}, 0) = g(\mathbf{x}) \quad \text{et} \quad f(\mathbf{x}, y+1) = h(\mathbf{x}, y, f(\mathbf{x}, y))}$$

Il faut y voir une récurrence comme pour toute suite définie par une initialisation u_0 et une relation $u_{n+1} = h(n, u_n)$. La récurrence se fait sur la dernière variable (ici y).

Définition.
L'ensemble PR des ***fonctions récursives primitives*** est le plus petit ensemble de $\mathcal{F}$ qui contient Z, S, P_i^p (pour tout $p \geqslant 1$, $1 \leqslant i \leqslant p$) et qui est stable pour les opérations de composition et de récurrence.

Que signifie « être stable »? Par exemple, pour la composition, si $f_1, \dots, f_q \in PR$ et $g \in PR$, alors la composée $g(f_1, \dots, f_q)$ est encore dans PR. Idem pour l'opération de récurrence.
Note. Le terme « récursive primitive » est mal choisi, mais c'est la terminologie usuelle. Un nom plus évocateur serait celui de fonction « facilement calculable », mais ce terme est utilisé dans d'autres situations.

1.2. Premiers exemples

(a) Fonctions constantes $\mathbb{N} \to \mathbb{N}$
Les fonctions constantes $f : \mathbb{N} \to \mathbb{N}$ sont récursives primitives. On le sait pour la fonction nulle Z. Pour la constante 1, on calcule $S(Z(x)) = S(0) = 1$, donc $f = S \circ Z$ est récursive primitive comme composée de deux fonctions de base. Plus généralement, $x \mapsto k$ est dans PR (par k compositions de S appliquées à Z).

(b) Fonctions constantes $\mathbb{N}^2 \to \mathbb{N}$
Soit $k \in \mathbb{N}$ un entier fixé. On définit :

- initialisation : $g : \mathbb{N} \to \mathbb{N}$, la fonction constante égale à k, qui est dans PR.
- étape de récurrence : $h : \mathbb{N}^3 \to \mathbb{N}$ définie par $h(x, y, z) = z$. C'est la projection P_3^3, donc elle est récursive primitive.

L'opération de récurrence construit la fonction $f : \mathbb{N}^2 \to \mathbb{N}$ appartenant à PR telle que $f(x, 0) = k$ et $f(x, y+1) = f(x, y)$. Par récurrence sur y, $f(x, y) = k$ pour tout y. Donc $f : \mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto k$ est récursive primitive.

On prouverait, par récurrence sur p, que toute fonction constante $f : \mathbb{N}^p \to \mathbb{N}$ est récursive primitive.

(c) Opérations arithmétiques élémentaires

Proposition 1.
Les fonctions suivantes sont récursives primitives :

- *l'addition :* $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x + y$
- *la multiplication :* $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x \times y$
- *l'exponentiation :* $\mathbb{N}^2 \to \mathbb{N}, (x, y) \mapsto x^y$

Démonstration.

- L'addition vérifie la relation de récurrence : $x + (y+1) = (x + y) + 1$. Ainsi on pose l'initialisation $g : \mathbb{N} \to \mathbb{N}$ définie par $g(x) = x$ (en fait $g(x) = P_1^1(x) = x$ est donc récursive primitive). Ensuite on

pose $h : \mathbb{N}^3 \to \mathbb{N}$ définie par $h(x,y,z) = S(z)$ (en fait $h = S \circ P_3^3$, donc h est récursive primitive). Par l'opération de récurrence on obtient $f : \mathbb{N}^2 \to \mathbb{N}$ récursive primitive.
Vérifions que $f(x,y) = x + y$. On fixe x et on fait une récurrence sur y.
— Pour $y = 0$, $f(x,0) = g(x) = x = x + 0$.
— Supposons $f(x,y) = x + y$. Alors :

$$f(x,y+1) = h(x,y,f(x,y)) = S(f(x,y)) = f(x,y) + 1 = (x+y) + 1 = x + (y+1)$$

On a prouvé par récurrence sur y que pour tout x, y, $f(x,y) = x + y$. On savait que f était récursive primitive, donc l'addition l'est aussi.

- Pour la multiplication, on utilise la relation $x \times (y+1) = x \times y + x$. Ce sera à faire dans les exercices.
- L'exponentiation est aussi à faire dans les exercices.

□

(d) La soustraction entière

Proposition 2.

- *La fonction prédécesseur* prec : $\mathbb{N} \to \mathbb{N}$,

$$\text{prec}(x) = \begin{cases} x-1 & si\ x \geqslant 1 \\ 0 & si\ x = 0 \end{cases}$$

est récursive primitive.
- *La* **soustraction entière** $\mathbb{N}^2 \to \mathbb{N}$, $(x,y) \mapsto x \dot{-} y$ *est aussi récursive primitive.*

$$\boxed{x \dot{-} y = \begin{cases} x-y & si\ x \geqslant y \\ 0 & sinon \end{cases}}$$

Démonstration.

- La fonction prec : $\mathbb{N} \to \mathbb{N}$ vérifie la relation de récurrence suivante : $\text{prec}(0) = 0$, et pour $y \geqslant 0$, $\text{prec}(y+1) = y$. Cela en fait intuitivement une fonction récursive primitive. Mais techniquement c'est un peu plus compliqué car l'opération de récurrence fournit une fonction $f : \mathbb{N}^{p+1} \to \mathbb{N}$. On procède ainsi :
— $g(x) = Z(x) = 0$ est dans PR.
— $h(x,y,z) = P_2^3(x,y,z) = y$ est aussi dans PR.
Donc $f : \mathbb{N}^2 \to \mathbb{N}$ définie par $f(x,0) = 0$ et qui vérifie $f(x,y+1) = h(x,y,f(x,y)) = y$ est dans PR. Ainsi $f(x,y)$ est récursive primitive.
Note : $f(x,y)$ est une fonction de deux variables où la variable x ne sert à rien. On s'en débarrasse par la composition suivante : prec : $\mathbb{N} \to \mathbb{N}$, définie par $\text{prec}(y) = f(P_1^1(y), P_1^1(y)) = f(y,y)$ qui est donc bien récursive primitive.
- La soustraction entière vérifie la relation de récurrence : $x \dot{-} 0 = x$ et $x \dot{-} (y+1) = \text{prec}(x \dot{-} y)$. On pose $g(x) = P_1^1(x) = x$ et $h(x,y,z) = \text{prec}(P_3^3(x,y,z)) = \text{prec}(z)$. Puisque g et h sont récursives primitives, par l'opération de récurrence, on obtient une fonction $f : \mathbb{N}^2 \to \mathbb{N}$ récursive primitive qui vérifie $f(x,y) = x \dot{-} y$.

□

1.3. Remarques

- **Autre définition de** PR**.** On peut aussi définir l'ensemble des fonctions récursives primitives « par le bas ».

$$PR_0 = \{Z, S\} \cup \{P_i^p\}_{p \geqslant 1, 1 \leqslant i \leqslant p}$$

$$PR_{n+1} = PR_n \cup \{\text{fonctions obtenues par opération de composition à partir de } PR_n\} \cup \{\text{fonctions obtenues par opération de récurrence à partir de } PR_n\}$$

Et alors $PR = \bigcup_{n\geqslant 0} PR_n$.

- **Variables.** On peut permuter les variables comme on veut. Par exemple, si $f : \mathbb{N}^2 \to \mathbb{N}, (x,y) \mapsto f(x,y)$ est récursive primitive, alors $(x,y) \mapsto f(y,x)$ est aussi récursive primitive. En effet, $f(y,x) = f(P_2^2(x,y), P_1^2(x,y))$. Ainsi, on utilise souvent les variables x, y, z directement et dans l'ordre que l'on veut.
- **Fonctions $f : \mathbb{N}^0 \to \mathbb{N}$.** Les éléments de $\mathbb{N}^p$ sont des p-uplets $(x_1, \dots, x_p)$. Pour $p = 0$, $\mathbb{N}^0$ est constitué d'un seul 0-uplet noté $()$. Ainsi $\mathbb{N}^0 = \{()\}$ et une fonction $f : \mathbb{N}^0 \to \mathbb{N}$ n'a qu'une seule valeur (car il y a un seul élément dans son ensemble de départ), c'est donc une fonction constante.
  Pour l'instant nous avions exclu ce cas $p = 0$. Mais cela peut être utile de l'autoriser et d'ajouter par convention que les fonctions $f : \mathbb{N}^0 \to \mathbb{N}$ sont récursives primitives (elles sont considérées comme des constantes).
  Cette convention permet d'utiliser l'opération de récurrence avec $p = 0$. Cela simplifie la preuve que la fonction prec est récursive primitive (en effet, dans la preuve, la variable x était inutile). On renvoie aux exercices pour montrer que $n!$ est récursive primitive avec cette méthode.

1.4. Algorithmes

Faisons le lien avec le chapitre précédent. Si $f : \mathbb{N}^p \to \mathbb{N}$ est une fonction récursive primitive, alors quel que soit $\mathbf{x} \in \mathbb{N}^p$, on peut calculer $f(\mathbf{x})$ par un algorithme n'utilisant que des boucles « pour » (avec un nombre d'itérations bien contrôlé).
Si on veut démontrer que les fonctions de PR vérifient une certaine propriété, il suffit de la vérifier pour les trois fonctions de base et de montrer que cette propriété est conservée par application des opérations de composition et de récurrence.
Il est clair que $Z(n) = 0$, $S(x) = x + 1$ et $P_i^p(\mathbf{x}) = x_i$ sont calculables facilement. Si f est la composée $g(f_1, \dots, f_q)$ et qu'il existe des algorithmes faciles pour $f_1, \dots, f_q$ et g, alors on calcule $y_1 = f_1(\mathbf{x}), \dots, y_q = f_q(\mathbf{x})$ par ces algorithmes, puis $g(y_1, \dots, y_q)$ qui donne ainsi $f(\mathbf{x})$ par une suite finie d'algorithmes faciles.
Si f est obtenue par opération de récurrence : $f(\mathbf{x}, 0) = g(\mathbf{x})$ est supposé se calculer par un algorithme facile, ce qui permet de calculer $f(\mathbf{x}, 1) = h(\mathbf{x}, 0, f(\mathbf{x}, 0))$ où h se calcule par un algorithme facile. De proche en proche, on calcule $f(\mathbf{x}, 2)$, $f(\mathbf{x}, 3)$,... jusqu'à la valeur voulue.

2. Ensembles et prédicats

2.1. Ensembles

Définition.
Soit $A \subset \mathbb{N}^p$. La *fonction caractéristique* $\chi_A : \mathbb{N}^p \to \mathbb{N}$ est définie par :

$$\chi_A(\mathbf{x}) = \begin{cases} 1 & \text{si } \mathbf{x} \in A \\ 0 & \text{sinon} \end{cases}$$

On la note aussi parfois $\mathbb{1}_A$.

Définition.
Un ensemble $A \subset \mathbb{N}^p$ est un *ensemble récursif primitif* si sa fonction caractéristique χ_A est une fonction récursive primitive.

Par exemple, l'ensemble $A \subset \mathbb{N}$ des entiers pairs est un ensemble récursif primitif. En effet, on peut définir sa fonction caractéristique par la relation de récurrence suivante :

$$\chi_A(0) = 1 \quad \text{et} \quad \chi_A(n+1) = 1 \dot{-} \chi_A(n)$$

Ceci fait de χ_A une fonction récursive primitive.

Proposition 3.
Si A et B sont des ensembles récursifs primitifs de $\mathbb{N}^p$, alors les ensembles $A \cup B$, $A \cap B$ et $\mathbb{N}^p \setminus A$ sont aussi des ensembles récursifs primitifs.

Pour la preuve, on a besoin de la ***fonction signe*** $\mathrm{sgn} : \mathbb{N} \to \mathbb{N}$ définie par :

$$\mathrm{sgn}(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n > 0 \end{cases}$$

Elle est récursive primitive car elle s'écrit aussi $\mathrm{sgn}(n) = 1 \dot{-} (1 \dot{-} n)$.

Démonstration. Les fonctions caractéristiques de ces ensembles s'expriment à l'aide des fonctions récursives primitives χ_A et χ_B :

- $\chi_{A \cap B} = \chi_A \times \chi_B$
- $\chi_{A \cup B} = \mathrm{sgn}(\chi_A + \chi_B)$
- $\chi_{\mathbb{N}^p \setminus A} = 1 \dot{-} \chi_A$

□

Proposition 4.
Si f est une fonction récursive primitive et $A = \{\mathbf{x} \in \mathbb{N}^p \mid f(\mathbf{x}) = 0\}$, alors A est un ensemble récursif primitif.

Démonstration. On a $\chi_A(\mathbf{x}) = 1 \dot{-} f(\mathbf{x})$. □

Proposition 5 (Définition par cas)**.**
Soient $g, h : \mathbb{N}^p \to \mathbb{N}$ des fonctions récursives primitives et $A \subset \mathbb{N}^p$ un ensemble récursif primitif. La fonction $f : \mathbb{N}^p \to \mathbb{N}$ définie par :

$$f(\mathbf{x}) = \begin{cases} g(\mathbf{x}) & \textit{si } \mathbf{x} \in A \\ h(\mathbf{x}) & \textit{sinon} \end{cases}$$

est une fonction récursive primitive.

Démonstration. On peut écrire $f(\mathbf{x}) = g(\mathbf{x}) \times \chi_A(\mathbf{x}) + h(\mathbf{x}) \times \chi_{\mathbb{N}^p \setminus A}(\mathbf{x})$. Elle s'exprime comme somme et produits de fonctions récursives primitives, donc l'est également. □

2.2. Prédicats

Définition.
Un prédicat $P : \mathbb{N}^p \to \{V, F\}$ à p places est un ***prédicat récursif primitif*** si l'ensemble

$$A = \{\mathbf{x} \in \mathbb{N}^p \mid P(\mathbf{x}) \text{ est vrai}\}$$

est un ensemble récursif primitif.
Autrement dit, cela signifie que la fonction caractéristique :

$$\chi_P(\mathbf{x}) = \begin{cases} 1 & \text{si } P(\mathbf{x}) \text{ est vrai} \\ 0 & \text{sinon} \end{cases}$$

est une fonction récursive primitive.

Proposition 6.
Si P et Q sont des prédicats à p places récursifs primitifs, alors les prédicats $\neg P$, $P \wedge Q$, $P \vee Q$, $P \to Q$, $P \leftrightarrow Q$ le sont aussi.

Démonstration. Notons A et B les ensembles récursifs primitifs associés respectivement à P et Q.

- L'ensemble $\mathbb{N}^p \setminus A$ est un ensemble récursif primitif, donc $\neg P$ est un prédicat récursif primitif. (Autre démonstration : $\chi_{\neg P}(\mathbf{x}) = 1 \dot{-} \chi_P(\mathbf{x})$.)

- Le prédicat $P \wedge Q$ a pour ensemble associé $A \cap B$ qui est bien un ensemble récursif primitif (ou bien $\chi_{P\wedge Q}(\mathbf{x}) = \chi_P(\mathbf{x}) \times \chi_Q(\mathbf{x})$).
- Le prédicat $P \vee Q$ a pour ensemble associé $A \cup B$ (ou bien $\chi_{P\vee Q}(\mathbf{x}) = \mathrm{sgn}(\chi_P(\mathbf{x}) + \chi_Q(\mathbf{x}))$).
- Pour les connecteurs $\to$ et $\leftrightarrow$, ils peuvent s'exprimer à l'aide des connecteurs précédents.

□

Proposition 7.
Les prédicats « $x > y$ » et « $x \geqslant y$ » sont récursifs primitifs.

Démonstration. Leurs fonctions caractéristiques s'écrivent :
- $\chi_>(x, y) = \mathrm{sgn}(x \dot{-} y)$
- $\chi_\geqslant(x, y) = \mathrm{sgn}((x+1) \dot{-} y)$

□

2.3. Schémas bornés

Rechercher des éléments définis par un minimum, ou par les quantificateurs $\forall$ ou $\exists$, conserve le caractère récursif primitif à condition qu'ils soient bornés. Ce sera fondamental pour les applications à l'arithmétique que l'on verra juste après.

Schéma du minimum borné
Soit $P(\mathbf{x}, t)$ un prédicat. Fixons $\mathbf{x}$ et y. On peut considérer :

$$\min\{t \leqslant y \mid P(\mathbf{x}, t) \text{ est vrai}\}.$$

Si l'ensemble $\{t \leqslant y \mid P(\mathbf{x}, t) \text{ est vrai}\}$ est vide, on pose par convention $\min = y + 1$.
Si A est un ensemble, on définit de même :

$$\min\{t \leqslant y \mid (\mathbf{x}, t) \in A\}$$

Proposition 8.
Si P est un prédicat récursif primitif, alors la fonction $f(\mathbf{x}, y)$ définie par :

$$f(\mathbf{x}, y) = \min\{t \leqslant y \mid P(\mathbf{x}, t) \textit{ est vrai}\}$$

est une fonction récursive primitive.
Il en est de même pour un ensemble A récursif primitif.

> **Exemple.**
> Considérons la fonction racine carrée entière $\mathbb{N} \to \mathbb{N}$, qui à y associe $\lfloor \sqrt{y} \rfloor$ (par exemple $\lfloor \sqrt{10} \rfloor = 3$).
> En fait, $\lfloor \sqrt{y} \rfloor$ est le plus petit entier dont le carré de son successeur dépasse y, c'est-à-dire :
>
> $$\lfloor \sqrt{y} \rfloor = \min\{t \leqslant y \mid (t+1)^2 > y\}$$
>
> C'est donc une fonction récursive primitive.

Démonstration. Intuitivement, pour $\mathbf{x}$ fixé et y fixé, il y a un nombre fini de valeurs $t \leqslant y$ à tester, c'est donc facile à calculer au sens algorithmique. Fixons $\mathbf{x}$ et trouvons la formule de récurrence.
- Initialisation : $(y = 0)$. Si $P(\mathbf{x}, 0)$ est vrai, on pose $g(\mathbf{x}) = 0$, sinon $g(\mathbf{x}) = 1$ (par la convention $\min = y + 1$).
- Formule de récurrence. On cherche $f(\mathbf{x}, y+1) = \min\{t \leqslant y+1 \mid P(\mathbf{x}, t)\}$. On trouve une formule par disjonction de cas (voir la Proposition 5) :

$$f(\mathbf{x}, y+1) = \begin{cases} f(\mathbf{x}, y) & \text{si } f(\mathbf{x}, y) \leqslant y \\ y+1 & \text{si } f(\mathbf{x}, y) = y+1 \text{ et } P(\mathbf{x}, y+1) \text{ est vrai} \\ y+2 & \text{si } f(\mathbf{x}, y) = y+1 \text{ et } P(\mathbf{x}, y+1) \text{ est faux} \end{cases}$$

□

Quantification bornée

Proposition 9.
Soit $A \subset \mathbb{N}^{p+1}$ un ensemble récursif primitif. Alors les ensembles :
- $B = \{(\mathbf{x}, y) \mid \forall t \leqslant y, (\mathbf{x}, t) \in A\}$
- $C = \{(\mathbf{x}, y) \mid \exists t \leqslant y, (\mathbf{x}, t) \in A\}$

sont aussi des ensembles récursifs primitifs.

Démonstration. Notons $A_t = \{(\mathbf{x}, t) \in A\}$. Alors $\chi_B = \chi_{A_0} \times \chi_{A_1} \times \cdots \times \chi_{A_y} = \prod_{t=0}^{y} \chi_{A_t}$ et $\chi_C = \operatorname{sgn}\left(\sum_{t=0}^{y} \chi_{A_t}\right)$. La somme et le produit bornés se définissent facilement par l'opération de récurrence. □

Cela fonctionne aussi sous la forme de prédicat. Par exemple, « être un carré parfait » est un prédicat récursif primitif. On rappelle que y est un carré parfait s'il existe t tel que $y = t^2$ ($0, 1, 4, 9, 16, 25, \ldots$ sont des carrés parfaits). Ce prédicat se définit par la quantification bornée :

$$\exists t \leqslant y,\ y = t^2.$$

2.4. Arithmétique

Proposition 10.
Les fonctions suivantes sont récursives primitives :
- $q(x, y)$ *le quotient de la division euclidienne de x par y (avec la convention $q(x, 0) = x + 1$).*
- $\pi(n)$ *qui calcule le n-ième nombre premier.*
- exposant(n, i) *qui calcule l'exposant de $\pi(i)$ dans la décomposition de n en facteurs premiers.*

Proposition 11.
L'ensemble $\mathcal{P}$ des nombres premiers est un ensemble récursif primitif.

Nous allons prouver ces deux propositions ensemble.

Démonstration.
- $q(x, y) = \min\{q \leqslant x \mid (q + 1) \times y > x\}$ est donc récursive primitive.
- Le reste $r(x, y) = x \dot{-} (q(x, y) \times y)$ l'est aussi.
- Le prédicat est_divisible(x, y) (qui vaut vrai si y divise x) s'obtient par $r(x, y) = 0$. C'est donc un prédicat récursif primitif.
- Être un nombre premier, c'est être un nombre strictement plus grand que 1 et ne pas avoir d'autres diviseurs que 1 et lui-même. Ainsi $\mathcal{P}$ est un ensemble récursif primitif :
$$\mathcal{P} = \{x \mid x > 1\} \cap \{x \mid \forall y \leqslant x \quad (y = 1 \vee y = x \vee \neg\text{est\_divisible}(x, y))\}$$
De façon équivalente, le prédicat est_premier(n) est récursif primitif.
- La fonction $\pi(n)$ qui renvoie le n-ième nombre premier (noté π_n, avec $\pi_1 = 2$) vérifie la relation de récurrence suivante, qui en fait une fonction récursive primitive :
$$\pi(n + 1) = \min\{t \leqslant \pi(n)! + 1 \mid t > \pi(n) \wedge \text{est\_premier}(t)\}$$
On rappelle qu'il existe toujours un nombre premier entre π_n et $\pi_n! + 1$ (ou même entre m et $2m$).
- L'exposant de π_i dans la décomposition de n en facteurs premiers est défini par :
$$\text{exposant}(n, i) = \min\{k \leqslant n \mid \neg\text{est\_divisible}(n, \pi_i^{k+1})\}$$
C'est donc bien une fonction récursive primitive.

□

Terminons par un contre-exemple célèbre.

Exemple.
La fonction d'Ackermann $A : \mathbb{N}^2 \to \mathbb{N}$ n'est pas récursive primitive. Elle est définie ainsi :
- $A(0,n) = n+1$
- $A(m+1,0) = A(m,1)$
- $A(m+1,n+1) = A(m,A(m+1,n))$

Pour m fixé, la fonction $A_m : \mathbb{N} \to \mathbb{N}, n \mapsto A(m,n)$ est récursive primitive. Mais la fonction $A : \mathbb{N}^2 \to \mathbb{N}$ ne l'est pas à cause de la récurrence imbriquée qui la fait croître plus vite que n'importe quelle fonction récursive primitive.

3. Application au codage de Gödel

Nous allons justifier, dans le monde des entiers, que toutes les opérations que l'on souhaite faire sur les nombres de Gödel se font par des fonctions récursives primitives, c'est-à-dire comme on l'avait présenté avant, que les calculs se font par des algorithmes avec des boucles « pour ».

3.1. Longueur, exposant, concaténation, substitution

On considère un nombre de Gödel $n = \pi_1^{c_1} \times \cdots \times \pi_l^{c_l}$ (avec les $c_i \geqslant 1$).
- On veut d'abord pouvoir retrouver l à partir de n de façon récursive primitive :
$$\text{longueur}(n) = \min\{i \leqslant n \mid \pi_{i+1} \text{ ne divise pas } n\}$$
- On a déjà étudié la fonction exposant(n,i) qui renvoie l'exposant c_i. On sait ainsi décoder un nombre de Gödel par une fonction récursive primitive.
- On a vu au chapitre précédent la formule de la concaténation. Si $n = \prod_{i=1}^{l} \pi_i^{\alpha_i}$ et $m = \prod_{j=1}^{l'} \pi_j^{\beta_j}$ alors :
$$n * m = n \times \prod_{j=1}^{l'} \pi_{j+l}^{\beta_j}$$
Ce sont des produits finis, donc c'est une opération récursive primitive.
- Considérons la fonction substitution(p,v,n) : pour une formule $\mathcal{P}$ (donnée par son nombre de Gödel p), de variable libre x (donnée par son code v), elle renvoie le code de Gödel de $\mathcal{P}(n)$ (c'est-à-dire $\mathcal{P}(n/x)$).
- On commence par extraire chaque code c_i de $\mathcal{P}$, on compare ce code à celui de la variable x, si c'est le même on remplace ce symbole par le terme n en utilisant la concaténation. C'est donc une succession de fonctions récursives primitives, donc la substitution est une fonction récursive primitive.

3.2. Variables, termes, formules

Décider si un entier n représente une variable, un terme, ou une formule valide sont des prédicats récursifs primitifs. On renvoie au chapitre précédent sans s'étendre davantage.
- est_variable(n)
- est_terme(n)
- est_formule(n)

3.3. Axiomes de Peano

On nous donne un entier n (un nombre de Gödel) et on doit décider si la formule qu'il encode est un des axiomes de Peano. Prenons l'exemple du premier axiome de Peano « $\forall x\ S(x) \neq 0$ » que l'on écrirait dans $\mathcal{L}_0$ par « $\forall x \neg Sx = 0$ », on note n_1 son nombre de Gödel.
Pour savoir si n code la formule de cet axiome, il suffit de tester si $n = n_1$. On fait cela pour les sept axiomes.
La difficulté vient des axiomes de récurrence :

$$\mathcal{R}ec_{\mathcal{P}(x),y} : \quad \mathcal{P}(0) \wedge \big[\forall y \quad \mathcal{P}(y) \to \mathcal{P}(S(y))\big] \quad \to \quad \forall x \ \ \mathcal{P}(x)$$

Il s'agit d'un schéma d'axiomes, il y a un axiome pour chaque formule $\mathcal{P}(x)$. Il y a donc à priori une infinité de formules et de variables à tester. Mais heureusement, pour un nombre de Gödel n donné, si x est une variable présente dans la formule de n, alors son code v est nécessairement inférieur à n. De même, si $\mathcal{P}$ est une sous-formule de la formule n, alors son code p vérifie aussi $p \leqslant n$.
On peut ainsi construire un prédicat récursif primitif sous la forme :

$$\begin{aligned}\text{est\_axiome\_recurrence}(n) = \exists \mathcal{P} \text{ de code } p \leqslant n \quad \exists x \text{ de code } v \leqslant n \quad \exists y \text{ de code } w \leqslant n \\ \text{tel que est\_formule}(\mathcal{P}) \text{ et est\_variable}(x) \text{ et est\_variable}(y) \\ \text{et } n \text{ code la formule } \mathcal{R}ec_{\mathcal{P}(x),y}\end{aligned}$$

Ainsi on sait détecter tous les axiomes de Peano de façon récursive primitive.

3.4. Preuves à la Hilbert

Dans les deux premières parties de ce livre, nous avons présenté les preuves avec leurs règles d'inférence dans leur écriture en déduction naturelle. Comme son nom l'indique, les preuves en déduction naturelle ont une écriture proche de l'écriture d'une preuve mathématique. Cependant il y a des hypothèses et des sous-hypothèses, et cette présentation n'est pas adaptée à un traitement algorithmique.
Nous allons voir l'écriture dans le système de Hilbert qui est plus adapté au codage de Gödel.

- Tout d'abord on se passe d'hypothèses. Si on devait prouver $\mathcal{P} \vdash \mathcal{Q}$, on prouverait $\vdash (\mathcal{P} \to \mathcal{Q})$.
- Il n'y a qu'une seule règle d'inférence pour la logique propositionnelle :

$$\mathcal{P}, \mathcal{P} \to \mathcal{Q} \quad \vdash \quad \mathcal{Q} \qquad (\textit{modus ponens})$$

- Par contre on a l'apparition d'axiomes qui sont des formules supposées toujours vraies (quel que soit ce que l'on souhaite démontrer).
- Pour la logique propositionnelle il y a trois axiomes (il existe des variantes, ici la variante de Lukasiewicz) :
  1. $\mathcal{P} \to (\mathcal{Q} \to \mathcal{P})$ (ajout d'hypothèse)
  2. $(\mathcal{P} \to (\mathcal{Q} \to \mathcal{R})) \to ((\mathcal{P} \to \mathcal{Q}) \to (\mathcal{P} \to \mathcal{R}))$ (distribution de l'implication)
  3. $(\neg\mathcal{Q} \to \neg\mathcal{P}) \to (\mathcal{P} \to \mathcal{Q})$ (contraposée)
- Ainsi pour vérifier si une suite $\mathcal{P}_1, \ldots, \mathcal{P}_l$ de formules est bien l'écriture d'une preuve de $\mathcal{Q}$, il suffit que pour tout i :
  — soit $\mathcal{P}_i$ est un des axiomes,
  — soit $\mathcal{P}_i$ est obtenu par *modus ponens* à partir des lignes précédentes, c'est-à-dire qu'il existe $j, k < i$ tels que $\mathcal{P}_k$ soit $\mathcal{P}_j \to \mathcal{P}_i$,
  et enfin on vérifie que $\mathcal{P}_l = \mathcal{Q}$.
- Pour la logique du premier ordre, on ajoute les axiomes suivants :
  — $(\forall x\ \mathcal{P}(x)) \to \mathcal{P}(t)$
  — $\big[\forall x\ (\mathcal{P} \to \mathcal{Q}(x))\big] \to \big[(\mathcal{P} \to \forall x\ \mathcal{Q}(x))\big]$ si $\mathcal{P}$ ne contient pas de variable libre x.
  — $x = x$

— $x = y \to (\mathcal{P}(x) \to \mathcal{P}(y))$

- Et on ajoute une seconde règle d'inférence, qui est une variante de la règle d'introduction de $\forall$:

$$\text{si} \quad \vdash \mathcal{P} \quad \text{alors} \quad \vdash \forall x\ \mathcal{P}$$

3.5. Être une preuve

Ainsi on comprend que vérifier si un super-nombre de Gödel N est une preuve de n se fait facilement à l'aide du système de Hilbert. On rappelle qu'on n'a pas d'hypothèses (pour prouver $\Gamma \vdash \mathcal{Q}$, on prouve $\vdash (\Gamma \to \mathcal{Q})$), que les règles d'inférence sont très simples (*modus ponens* et introduction de $\forall$) et qu'une preuve est juste une suite linéaire de formules.
On construit ainsi un prédicat est_preuve(N, n) récursif primitif qui vérifie si la suite de formules $\mathcal{P}_i$ codée par le super-nombre de Gödel N est bien une preuve valide de la formule codée par le nombre de Gödel n.
La construction de ce prédicat se fait ainsi (pour la logique propositionnelle) :

- extraire de N les expressions $\mathcal{P}_i$, $i = 1, \ldots, l$ et de n l'expression $\mathcal{Q}$,
- vérifier que ces expressions sont des formules valides,
- pour chaque $i = 1, \ldots, l$:
  — vérifier si $\mathcal{P}_i$ est un axiome,
  — ou si $\mathcal{P}_i$ se déduit par *modus ponens* des lignes précédentes (c'est-à-dire qu'il existe $j, k < i$ tels que $\mathcal{P}_k$ soit $\mathcal{P}_j \to \mathcal{P}_i$).
- vérifier si la dernière formule $\mathcal{P}_l$ est bien $\mathcal{Q}$.