Décidabilité de l'arithmétique de Presburger

Léo Gayral

2017-2018

ref : Carton – Langages formels, Calculabilité et complexité – p.178

Remarque 1. L'arithmétique de Presburger est la théorie $T = \mathrm{Th}\,(\mathbb{N}, +, 0, 1)$. Elle est donc complète. On peut montrer que cette théorie est axiomatisée par :

- $\forall x,\ \neg(0 = x + 1)$,
- $\forall x, y,\ (x + 1 = y + 1) \to x = y$,
- $\forall x,\ x + 0 = x$,
- $\forall x, y,\ (x + y) + 1 = x + (y + 1)$,
- $\forall \overline{x},\ (P(0, \overline{x}) \wedge [\forall y,\ P(y, \overline{x}) \to P(y + 1, \overline{x})]) \to [\forall y,\ P(y, \overline{x})]$ pour toute formule $P\,(y, x_1, \ldots, x_n)$.

Théorème 1. L'arithmétique de Presburger est décidable.

Démonstration.
On peut directement se restreindre aux formules $Q_n x_n \ldots Q_1 x_1 \varphi\,(x_1, \ldots, x_n)$ sous forme prénexe, et φ sous forme normale conjonctive. En outre, quitte à ajouter des variables *intermédaires*, de sorte que :

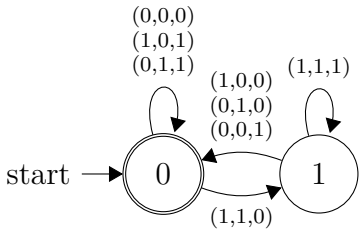
$$T \vdash (x + y) + z = t \leftrightarrow \exists k,\ x + y = k \wedge k + z = t$$

on peut supposer que toutes les formules atomiques sont du type $x + y = z$, avec x, y et z des variables ou des constantes.

Sur ces formules, on va montrer, par récurrence sur k, que le langage accepté par la formule $\psi_k := Q_{k+1} x_{k+1} \ldots Q_n x_n \varphi$ – l'ensemble des codages de k-uplets tels que $\mathbb{N} \vDash \psi_k\,(x_1, \ldots, x_k)$ – est rationnel.

On considère le codage en binaire de $(x_1, \ldots, x_k) \in \mathbb{N}^k$ dans l'alphabet $\{0, 1\}^k$, écrit de gauche à droite, quitte à ajouter des 0 à droite de certaines décompositions pour garantir la même longueur.

Étant donné un triplet (x, y, z), l'automate suivant accepte le langage de la formule $x + y = z$:



où l'état i correspond à la valeur du reste sur la puissance de 2 sommée à ce moment-là. On peut en outre adapter cet automate dans le cas où x, y et·ou z est en fait constante, égale à 0 ou 1.

Quitte à ajouter des arêtes pour chaque valeur possible sur les autres coordonnées dans $\{0,1\}^n$, on peut plus généralement accepter $x_i + x_j = x_k$ sur les codages de $(x_1, \ldots, x_n)$. Comme l'ensemble des langages rationnels est stable par union et intersection, on peut en déduit un automate A_n qui reconnait le langage de $\psi_n = \varphi$.

Supposons que $Q_{k+1} = \exists$. Soit A_{k+1} un automate qui reconnait le langage de ψ_{k+1}. On définit la projection sur les k première coordonnées par :

$$\pi_k : \begin{array}{ccc} \{0,1\}^{k+1} & \to & \{0,1\}^k \\ (x_1, \ldots, x_{k+1}) & \mapsto & (x_1, \ldots, x_k) \end{array} .$$

On considère alors l'automate A_k qui a les mêmes sommets que A_{k+1}, et dont les transitions sont les projections de celles de A_{k+1} : si $u \xrightarrow{c} v$ dans A_{k+1}, alors $u \xrightarrow{\pi_k(c)} v$ dans A_k. A_k accepte un mot ssi ce mot admet un relèvement accepté dans A_{k+1}, autrement dit ssi le mot code un $(n-k)$-uplet accepté par ψ_k.

Dans le cas où $\psi_k = \forall x_{k+1} \psi_{k+1} \equiv \neg\left(\exists x_{k+1}\left[\neg\psi_{k+1}\right]\right)$, on peut commencer par déterminiser A_{k+1}, puis complémenter l'ensemble des états acceptants (ce qui revient à accepter le langage complémentaire, celui de $\neg\psi_{k+1}$), suivre la méthode précédente, puis à nouveau déterminiser et complémenter pour enfin obtenif A_k qui accepte le langage de ψ_k.

On obtient enfin un automate A_1 qui reconnait le langage de ψ_1. Si $Q_1 = \exists$, il faut vérifier $L(A_1) \neq \emptyset$. Si $Q_1 = \forall$, il faut vérifier $L(A_1) = \{0,1\}^*$. Dans les deux cas, on peut appliquer l'algorithme de minimisation de Hopcroft, puis vérifier si l'automate obtenu correspond à celui de $\emptyset$ ou $\{0,1\}^*$. □

Remarque 2. En pratique, la mise sous forme prénexe de la formule initiale, la construction de l'automate A_n, ainsi que les diverses déterminisations nécessaires, induisent une explosion de la taille des données.