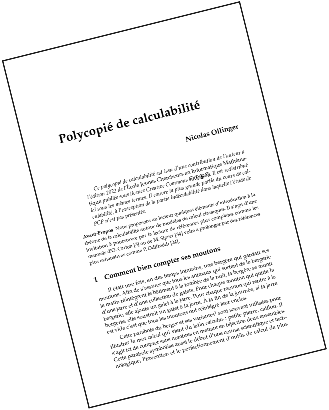
Calculabilité & Complexité

Calculabilité (3/5) : la machine universelle

Nicolas Ollinger (LIFO, Université d'Orléans)

M1 info, Université d'Orléans — S2 2025/2026

Rappel de méthode



Polycopié de calculabilité

Nicolas Ollinger

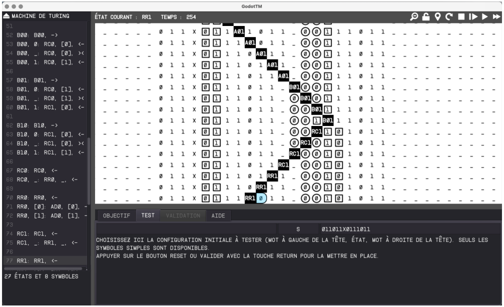
*Ce polycopié de calculabilité est issu d'une contribution de l'auteur à l'édition 2022 de l'*École Jeunes Chercheurs en Informatique Mathématique *publiée sous licence Creative Commons* 🅭🅯🄏🄎*. Il est redistribué ici sous les mêmes termes. Il couvre la plus grande partie du cours de calculabilité, à l'exeception de la partie indécidabilité dans laquelle l'étude de PCP n'est pas présentée.*

Avant-Propos Nous proposons au lecteur quelques éléments d'introduction à la théorie de la calculabilité autour de modèles de calcul classiques. Il s'agit d'une invitation à poursuivre par la lecture de références plus complètes comme les manuels d'O. Carton [3] ou de M. Sipser [34] voire à prolonger par des références plus exhaustives comme P. Odifreddi [24].

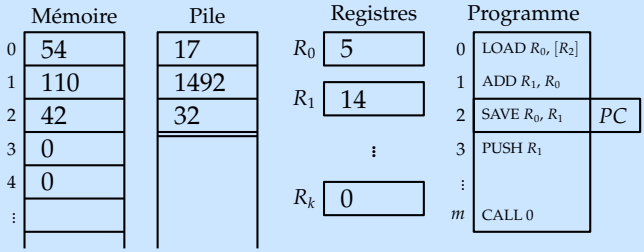
1 Comment bien compter ses moutons

Il était une fois, en des temps lointains, une bergère qui gardait ses moutons. Afin de s'assurer que tous les animaux qui sortent de la bergerie le matin réintègrent le bâtiment à la tombée de la nuit, la bergère se munit d'une jarre et d'une collection de galets. Pour chaque mouton qui quitte la bergerie, elle ajoute un galet à la jarre. Pour chaque mouton qui rentre à la bergerie, elle soustrait un galet à la jarre. À la fin de la journée, si la jarre est vide c'est que tous les moutons ont réintégré leur enclos.

Cette parabole du berger et ses variantes[1] sont souvent utilisées pour illustrer le mot *calcul* qui vient du latin *calculus* : petite pierre, caillou. Il s'agit ici de compter sans nombres en mettant en bijection deux ensembles. Cette parabole symbolise aussi le début d'une course scientifique et technologique, l'invention et le perfectionnement d'outils de calcul de plus



- Lire et relire le polycopié disponible sur Celene ;
- S'entraîner avec le simulateur de MT.



1. Dans l'épisode précédent...

Turing-complétude

Un **modèle de calcul** décrit une **famille dénombrable** d'objets et la manière de les utiliser pour calculer.

Une **fonction de codage acceptable** décrit une transformation raisonnable entre modèles pour coder les entrées et les sorties.

Définition Deux modèles de calcul se **simulent** entre eux s'ils calculent les même fonctions à un codage acceptable près.

Définition Un modèle de calcul est **Turing-complet** s'il est équivalent au modèle des machines de Turing.

IMP : l'archétype du langage impératif

Variables	$\ni$	X, Y		
Expressions	$\ni$	E, F	$::=$	X
			$\mid$	Constante entière
			$\mid$	$E \odot F$ avec $\odot \in \{+, -, *, \mathtt{DIV}, \mathtt{MOD}\}$
			$\mid$	$E \odot F$ avec $\odot \in \{=, <, >, \mathtt{AND}, \mathtt{OR}\}$
			$\mid$	$\odot E$ avec $\odot \in \{\mathtt{NOT}, -\}$
Instructions	$\ni$	S, T	$::=$	$X = E$
			$\mid$	S ; T
			$\mid$	IF E THEN S ELSE T
			$\mid$	WHILE E DO S
Programme	$\ni$	P	$::=$	READ X; S; WRITE Y

Calcule des fonctions partielles $f : \mathbb{N} \to \mathbb{N}$.

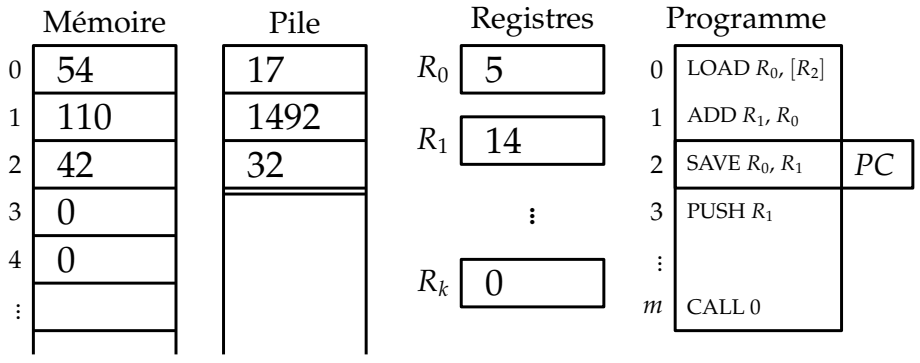
I : forme minimale de IMP

Pour simplifier les simulations entre modèles, il est souvent utile d'avoir une sorte de forme normale minimale du modèle cible.

Théorème Toute fonction calculée par un programme IMP est calculée par un programme I : un programme IMP utilisant au plus **une boucle** et **deux variables**.

Remarque La version à deux variables est un peu technique mais on peut se convaincre facilement qu'on sait le faire avec un nombre constant de variables.

RAM : l'archétype du processeur moderne



Les registres stockent des entiers de taille arbitraire, le jeu d'instructions est choisit suffisament expressif.

Calcule des fonctions partielles $f : \mathbb{N} \to \mathbb{N}$.

REG : forme minimale de RAM

Théorème Toute fonction calculée par un programme RAM est calculée par un programme REG : un programme RAM utilisant seulement **3 registres** et **sans pile ni mémoire**.

Remarque La mémoire, les registres et la pile stockent à chaque instant un tuple d'entiers. Avec un codage astucieux, on peut les coder dans un unique entier. On se garde deux autres registres pour manipuler ce codage.

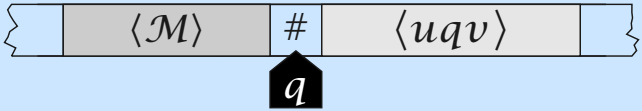
Turing-équivalence

Théorème Les modèles de calcul **MT**, **IMP** et **RAM** calculent les mêmes fonctions (à un **codage acceptable** près).

Démonstration

1. **MT++** simule **REG** *au programme du TD !*
2. **REG** est équivalent à **RAM**
3. **RAM** simule **I** *par compilation du langage !*
4. **I** est équivalent à **IMP**
5. **IMP** simule **MT** *par simulation des MT !*
6. **MT** est équivalent à **MT++**

où **MT++** désigne les MT enrichies avec rubans mutiples, multi-têtes, semi-infinis, *etc.* ■



2. Machines programmables

Changer de machine

L'**automate** au cœur de nos **ordinateurs** est fixé *in silico*.

Contrairement aux calculatrices de notre enfance à 4, 10, 100 fonctions, on ne change plus de machine pour avoir de nouvelles possibilités de calcul.

Les **programmes** sont des **données** comme les autres, **chargés** et **exécutés** et même **compilés** sur place.

Cette possibilité existe dans la plupart des modèles de calcul !

Machine de Turing universelle

Théorème Il existe une machine de Turing **universelle** U qui **simule** toutes les machines de Turing.

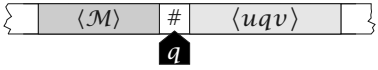
Pour toute machine de Turing $\mathcal{M}$ sur l'alphabet Σ et toute entrée $u \in \Sigma^*$, la machine U s'arrête sur l'entrée $\langle \mathcal{M}, u \rangle$ si et seulement si $\mathcal{M}$ s'arrête sur l'entrée u. Elle accepte si et seulement si $\mathcal{M}$ accepte et dans ce cas

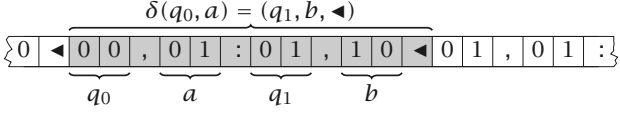
$$f_U\left(\langle \mathcal{M}, u \rangle\right) = \left\langle f_{\mathcal{M}}(u) \right\rangle .$$

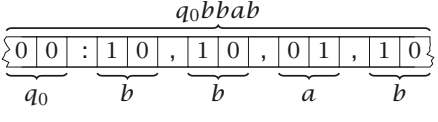
Remarque Nécessite de préciser un **codage acceptable** des MT.

Notation $\langle x \rangle$ désigne le **codage** de x.

Principes de construction







Dans le prochain épisode

Proposition Il existe des (*multitudes de*) langages **non récursifs**.

Il suffit de compter :

- les MT sont **dénombrables** ;
- les langages sur $\{a, b\}$ ne le sont **pas**.

Idée utiliser l'**argument diagonal** de Cantor