B) INDECIDABILITE (incalculable)

Questions fondamentales et directrices :

1) Que peut-on calculer ?
2) Qu'est-ce qu'un calcul ?

La première sous-entend qu'il y a des choses qu'on ne peut pas tout calculer, et on va les exhiber.
Quant à la seconde, on peut préciser un peu plus : on se demandera ce qu'est une méthode de calcul. On va en effet faire la différence entre "méthode de calcul" et "application (= le déroulement) de la méthode" sur un cas particulier (=une instance). C'est ce fait qu'on appellera ici "calcul".

En ce qui concerne les méthodes de calcul, une qualification importante réside en l'effectivité.
On n'en donnera pas une définition formelle, on n'a pas de "vraie" définition.
Intuitivement, on peut la caractériser comme une méthode qui effectue vraiment le calcul. Autrement dit, qui le décompose en des opérations vraiment élémentaires (i.e. "bêtes" et mécaniques). Les conséquences sont non négligeables :

- On a des opérations qu'on peut faire réfléchir
- On a des opérations qui pourraient être effectuées par des machines à notre place

Le problème étant que la notion de "Méthode de Calcul Effective" (= MCE) n'est pas formalisée. C'est d'ailleurs ce à quoi s'est attelé Turing (1936) et il a finalement réussi à formaliser une MCE à l'aide d'une machine dite "de Turing". Il a également démontré qu'il existe certaines fonctions bien définies qui ne sont pas calculables par une machine de Turing.
Mais comment être sûr que la formalisation de MCE par des machines de Turing est la bonne ? On ne peut, en l'état actuel des connaissances, pas y répondre.
Cette question a d'ailleurs fait l'objet d'une thèse (la thèse de Church-Turing) dont la conclusion se résume ainsi : les fonctions pour lesquelles il existe une MCE sont exactement les fonctions calculables par une machine de Turing.

I) Préliminaires : que veut-on calculer ?

On veut calculer des fonctions telles que :

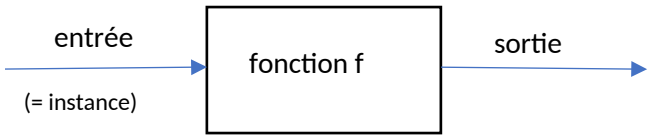
$$f : D \longrightarrow A$$
$$x \longmapsto f(x)$$

Où D et A sont (des ensembles) codables sur un alphabet fini.

Remarque : On peut parler d'application en rajoutant $\perp$ à l'ensemble d'arrivée (= $A \cup \{\perp\}$).
Donc on peut toujours, en l'espèce, passer de fonctions à applications.

Exemple : On peut parler de la fonction carrée qui va de $\mathbb{N}$ dans $\mathbb{N}$ et qui, à un entier, associe son carré

D'un point de vue plus "informatique", une fonction va de $\Sigma 1^*$ à $\Sigma 2^*$, où $\Sigma 1$ et $\Sigma 2$ sont des alphabets finis.
Dans la suite de ce cours, on considèrera les fonctions selon le schéma suivant :



Le calcul correspond alors à l'exécution de la fonction ; tandis que la méthode de calcul correspond à l'algorithme, au programme qui l'implémente.

Idéalement, on voudrait donc une méthode de calcul effective pour toutes les fonctions
$f : \Sigma 1^* \rightarrow \Sigma 2^*$.
On a alors deux exigences sur la méthode :

- Elle doit être effective au sens "défini" plus haut
- Elle doit toujours terminer en un nombre fini d'étapes

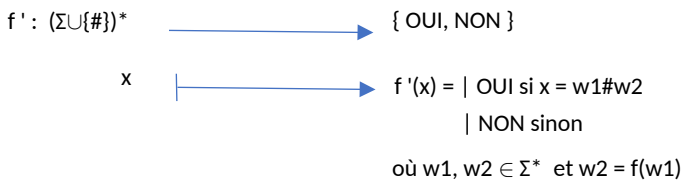
On se confronte alors à une nouvelle restriction : toute la théorie est construite avec des applications de Σ^* vers l'ensemble {OUI, NON}.

Pourtant : Montrons qu'on peut toujours trouver une équivalence entre "application calculable" et "application de décision"
Soit une application f (calculable) telle que :

$$f : \Sigma^* \longrightarrow \Sigma^*$$
$$x \longmapsto f(x)$$

Remarque: En posant $\Sigma = \Sigma 1 \cup \Sigma 2$, on n'impose aucune restriction sur ce qu'on a déjà vu.

Soit f' une application (de décision) telle que :



(i) Montrons que si on sait calculer f par une MCE, alors on peut calculer f ' par la MCE suivante :

1) On vérifie que x contient exactement un # :
   - Si c'est le cas : on calcule f(w1) où w1 est ce qui précède le # dans x
   - Sinon : on répond NON
2) On vérifie que f(w1) = w2 où w2 est ce qui suit le # dans x :
   - Si on a égalité : on répond OUI
   - Sinon : on répond NON

(ii) Montrons que si on sait calculer f ' par une MCE, alors on sait calculer f par la MCE suivante : *(on se donne x et on veut calculer f(x))*

1) On énumère les mots w de Σ* (qui est dénombrable)
2) Pour chaque mot w, on calcule f '(x#w) :
   - Si f '(x#w) = OUI : on répond w
   - Sinon : on continue à énumérer les mots w de Σ*

On peut remarquer que ce deuxième "algo" va se terminer puisqu'il existe toujours un w qui va être l'image de x par f.

Donc la restriction soulevée plus haut n'en est en réalité pas une.
Finalement, on peut définir une application f : Σ* → {OUI, NON} par le sous-ensemble de mots de Σ* tels que f(w) = OUI (avec w ∈ Σ*).
On appelle d'ailleurs "langage" un ensemble de mots.

Dans la suite, on chercher à déterminer quels sont les langages L décidables (i.e. tels qu'il existe une MCE qui prend une entrée un mot w ∈ Σ* et qui répond OUI si et seulement si w ∈ L).

II) Machine de Turing

a) Définition d'une machine de Turing

On peut la voir comme un automate avec mémoire qui prend un lecture un ruban.
De façon intuitive :

Définition formelle :
Une machine de Turing est un septuplet noté $(Q, \Sigma, \#, \Gamma, \delta, q_0, F)$ où :

- Q : un ensemble fini d'états
- Σ : l'alphabet d'entrée
- # : un symbole spécial appelé "blanc"
- Γ : l'alphabet du ruban tel que $\Sigma \cup \{\#\} \subseteq \Gamma$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$ *L et R pour (respectivement) Left et Right ?*
- q_0 : état initial
- F : ensemble des états acceptants

Une transition $q_i, x \mapsto q_j, y, \alpha$ correspond à :

La machine est dans l'état q_i, la case du ruban qui est pointée par la tête de lecture contient x et la machine évolue alors :

- En passant dans l'état q_j
- En écrivant y à la place de x dans la case de ruban pointée
- Et en bougeant la tête de lecture d'une case dans la direction α

Remarque : δ est appelée "fonction de transition". On peut dresser une table de transitions :

Q	Γ	Q	Γ	{L, R}
q_i	x	q_j	y	L
q_3	a	q_2	b	R
…	…	…	…	…

Remarque : Dans une machine déterministe, pour tout (q,x) il existe au plus un triplet (q', x', α)

b) <u>Fonctionnement d'une machine de Turing</u>

Initialement, le mot $w = w_1w_2...w_k$ est placé au début du ruban, il est suivi d'une infinité de blanc, la tête de lecture est sur la première case et la machine est dans l'unique état initial q_0.
Idée du ruban à l'état initial :

w_1	w_2	w_3	…	w_k	#	#	#	#	…

↑