Résumé de LINFO1123

compilation du 23 février 2023

Thomas Debelle

Juin 2023

Table des matières

Préface

Bonjour à toi !

Cette synthèse recueille toutes les informations importantes données au cours, pendant les séances de tp et est amélioré grâce au note du Syllabus. Elle ne remplace pas le cours donc écoutez bien les conseils et potentielles astuces que les professeurs peuvent vous donner. Notre synthèse est plus une aide qui on l'espère vous sera à toutes et tous utiles.

Elle a été réalisée par toutes les personnes que tu vois mentionné. Si jamais cette synthèse a une faute, manque de précision, typo ou n'est pas à jour par rapport à la matière actuelle ou bien que tu veux simplement contribuer en y apportant ta connaissance ? Rien de plus simple ! Améliore la en te rendant ici où tu trouveras toutes les infos pour mettre ce document à jour. (*en plus tu auras ton nom en gros ici et sur la page du github*)

Nous espérons que cette synthèse te sera utile d'une quelconque manière ! Bonne lecture et bonne étude.

Chapitre 1

Concepts

Dans ce chapitre, on s'intéresse aux ensembles, cardinalité et équipotences de ces derniers

1.1 Ensemble

Un ensemble est un *collection* d'objets, *sans répétition*, ces derniers sont appelés *éléments* de l'ensemble. Donc un ensemble peut être des chiffres, des lettres, il peut être vide symbolisé par *void* On peut réaliser des opérations dessus, on peut déterminer des *sous-ensembles d'ensemble* donc des ensembles issus d'ensemble. On a également une notion s'appelant le *complément* d'un ensemble dénoté $\tilde{A}$

Langage

Un *langage* n'est autre qu'un mot ou bien un ensemble de caractères d'une taille fixée. Une chaine vide est écrite via le caractère "ϵ".
On forme un langage via un *alphabet* qui n'est autre qu'un ensemble de symboles, on le dénote "Σ".
Tout langage est donc une suite de symbole issue de l'*alphabet*.
Σ^* correspond à l'ensemble des langages formés via l'alphabet.

Relations

Lorsque nous avons deux ensembles appelés A et B, on peut établir une relation appelé R qui nous donne un sous-ensemble AXB On peut représenter la relation par une table.

Fonctions

Lorsque nous avons deux ensembles appelés A et B, on peut avoir ce qu'on appelle une *fonction* F. C'est une relation tel que :

$$\exists a \in A : \exists b \in B : \quad < a, b > \quad \in f \tag{1.1}$$

Il n'existe pas plus d'un b pour un a. Si pour un a il n'existe pas de b, on dit que $f(a)$ est indéfini et donc $f(a) = \perp$ ou *bottom*.

Propriétés des fonctions

- un domaine de fonction ou dom(f) $= a \in A | f(a) \neq \perp$
- une image de fonciton ou image(f) $= b \in B | \exists a \in A : b = f(a)$
- f est dit fonction totale si dom(f) $= A$
- f est dit fonction partielle si dom(f) $\in A$

— f est surjectif ssi image(f) = B autrement dit, tout élement est associé à minimum 1 élément dans B.
— f est injectif ssi $\forall a, a' \in A : a \neq a' \Rightarrow f(a) \neq f(a')$ autrement dit on ne fait correspondre qu'au plus un élément de A dans B.
— f est bijectif s'il combine *surjectif* et *injectif*

Intéressons nous aux extensions qui est le fait de rajouter une fonction qui ne définit un élément de B pas encore défini.

$$\forall x \in A : g(x) \neq \perp \Rightarrow f(x) = g(x) \tag{1.2}$$

f à la même valeur que g partout où g est défini.

Définition d'une fonction

Comme dit précédemment, une fonction est défini par sa table. On va souvent utiliser une description de la table qui permet que celle-ci soit clair et bien défini. De plus, on a pas besoin de savoir comment calculer ceci.
On peut également définir une table via une fonction ou un algorithme.

1.2 Ensemble énumérable

On dit que 2 ensembles ont le même cardinal (A et B) ssi il existe une bijection entre ces 2 ensembles. Donc chaque élément de A correspond à un élément de B.

On dit d'un ensemble qu'il est dénombrable ssi il est fini ou il existe une bijection entre l'ensemble $\mathbb{N}$ et cet ensemble.

Exemples

— L'ensemble $\mathbb{Z}$
— L'ensemble des nombres pairs
— Des paires d'entiers
— L'ensemble des programmes Java

Propriétés

Tout sous-ensemble d'ensemble énumérable est *énumérable*. L'union et l'intersection d'ensembles énumérables est *énumérable*.

En s'intéressant à l'ensemble des programmes informatiques, on se rend compte que c'est une *ensemble énumérable infini*. De plus, les programmes informatiques ne considèrent que des choses *énumérables*.

1.3 Cantor

Le théorème de *Cantor* nous dit que l'ensemble des nombres entre 0 et 1 compris est *non énumérable*.

$$E = x \in \mathbb{R} | 0 < x \leq 1 \tag{1.3}$$

Preuve

Pour prouver cela, on va réaliser une table et on va réaliser une *diagonalisation de Cantor*.

	chiffre 1	chiffre 2	...	chiffre $k+1$	...
x_0	x_{00}	x_{01}	...	x_{0k}	...
x_1	x_{10}	x_{11}	...	x_{1k}	...
...	...	...	...	...	...
x_k	x_{k0}	x_{k1}	...	x_{kk}	...
...	...	...	...	...	...

Ensuite, on va définir notre nombre de la diagonale qui vaut $d = 0.x_{00}x_{11}...x_{kk}$. De cet valeur, on va créer une valeur d' qui a comme propriété $x_{kk} \neq x'_{kk} \forall k$.
Mais, on doit stocker notre valeur d' dans la table. On la stock à p ce qui donne $d' = 0.x'_{p0}x'_{p1}...x'_{pp}$ mais à cause de la construction de $d = 0.x_{00}x_{11}...x_{pp}$. Par construction, $x'_{pp} \neq x_{pp}$ mais cela ne peut être respecté. Donc, **il n'y a pas** de *bijection* des $\mathbb{N}$ vers cet ensemble. Donc cet ensemble est *non énumérable*.

Autre ensemble non énumérable

— L'ensemble des $\mathbb{R}$.
— L'ensemble des sous-ensemble de $\mathbb{N}$.
— L'ensemble des chaines infinies de caractères d'un alphabet fini.
— L'ensemble des *fonctions* de $\mathbb{N}$ dans $\mathbb{N}$.

Chose intéressante à noter, comme on a une infinité non énumérable de fonctions $\mathbb{N}$ dans $\mathbb{N}$ et un nombre de programme informatique *infini énumérable*. On ne peut résoudre tous les problèmes informatiques donc.

Chapitre 2

Programmes calculables

2.1 Les algorithmes

Un algorithme est un ensemble *d'instructions* qui a pour but de produire un résultat. Donc un algorithme n'est pas une fonction. Il *calcule* une fonction. Un algorithme n'est pas forcément un *programme*, il peut être un *organigramme*. C'est un *ensemble fini d'instructions*. C'est une sorte de *calculateur*. Ici, on va considérer nos algorithmes comme *n'ayant pas de limite* de :
— Taille de données
— Taille d'instructions
— Taille de la mémoire, mais on a une utilisation finie.

2.1.1 Calculabilité

Avant de continuer, il faut définir la *calculabilité* des algorithmes car sans *formalisme*, les algorithmes sont non rigoureux, non exploitables.

Ici, on base cette notion sur celle des *programmes informatiques*. (plus intuitif). Ainsi, on possède **2 univers** celui des *programmes informatiques* et celui des *problèmes*. Pour être plus précis, on se base sur le langage Java et on se limite au fonction $\mathbb{N} \to \mathbb{N}$. Ainsi pour les fonctions, on aura **1 entrée** et **1 sortie**. (on peut également généraliser ceci en disant que $\mathbb{N}^n \to \mathbb{N}$)

2.2 Fonction calculable

Une fonction est dite *calculable* s'il existe un *programme Java* recevant **1 donnée** étant un nombre $\in \mathbb{N}$ et la fonction va nous retourner la *valeur* de $f(x)$ *si* elle est défini.
Si le programme *ne se termine pas* donc pas défini ou erreur d'exécution on dit que $f(x) = \perp$. On définit bien la notion de calculabilité sur *l'existence d'un programme*. on a 2 types de fonctions

1. Fonction *partielle* calculable : on a *parfois* un résultat
2. Fonction *totale* calculable : on peut *toujours* calculé quelque chose.

2.2.1 Ensemble récursif

Maintenant, on va essayer de déterminer la calculabilité sur *un ensemble de fonctions*. Le principe de décision de *calculabilité* est le principe dit *récursif*.
A est récursif si il existe un programme *Java* qui recevant n'importe quelle donnée sous forme d'un $\mathbb{N}$ fourni comme résultat :
— 1 si $x \in A$
— 0 si $x \notin A$

Donc on est face à un *algorithme* qui calcule si x est dans A ou non. C'est un algorithme complet et se termine toujours. (attention de ne pas confondre *récursif* et *récursivité*)

On dit qu'un ensemble d'algorithme est récursivement énumérable s'il est *récursif* sauf qu'il retourne $\neq 1 \quad x \notin A$ ou ne se termine pas.