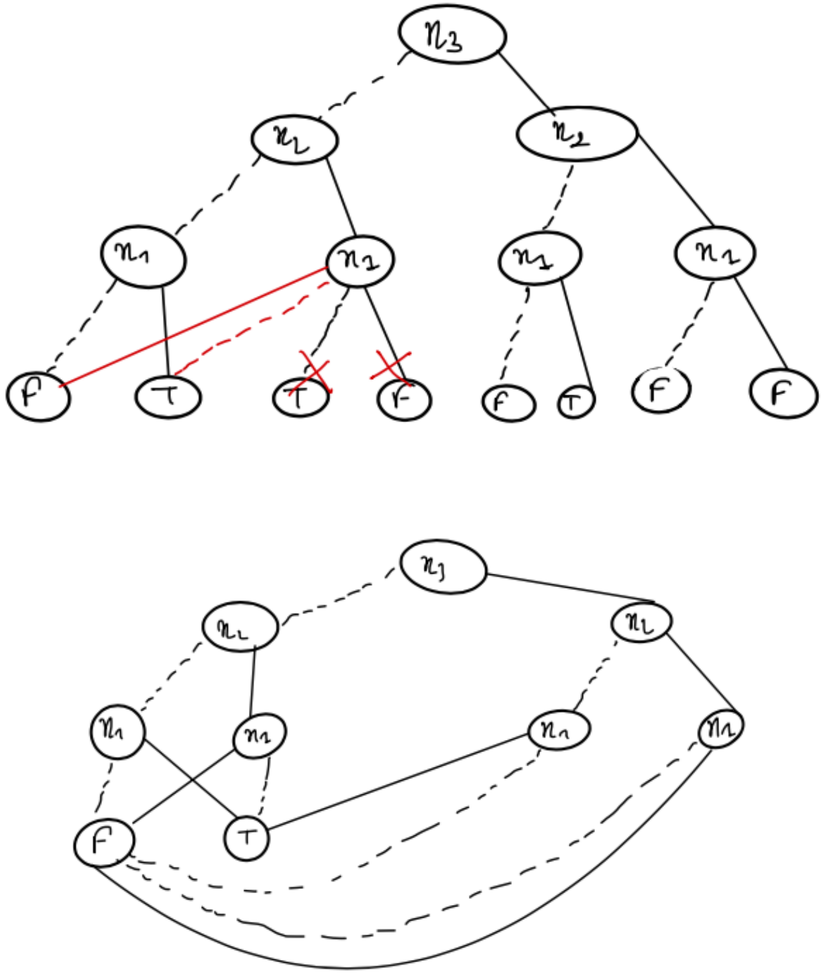
Projet d'ALGAV

Maxime DERRI & Yaël ZARROUK

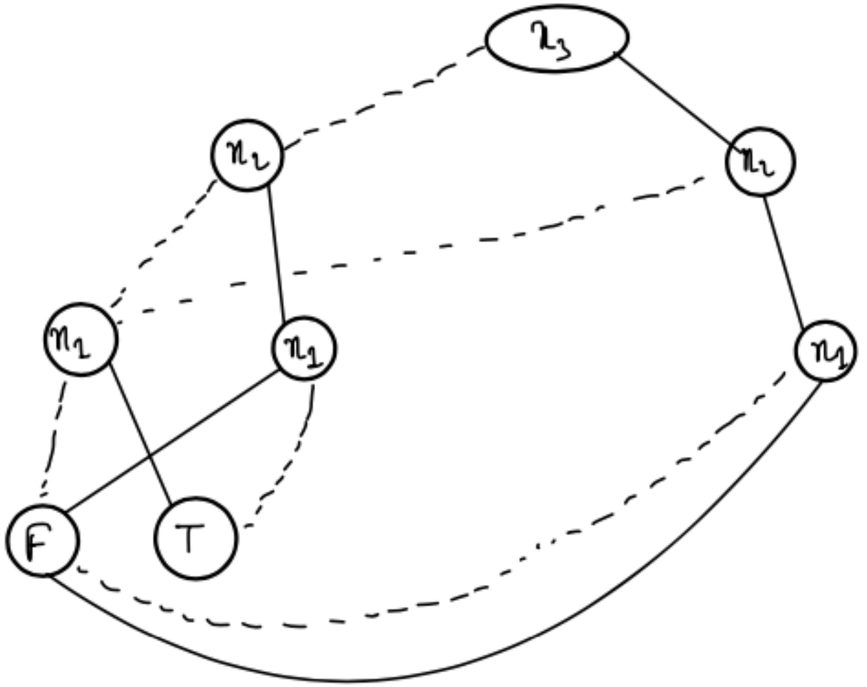
2022 - 2023

La fonction de compression en DAG(mot de Luka, hashset)

- Permet de transformer un arbre de décision en DAG
- On stocke dans une table de hash plusieurs couples (mot de Luka, les adresses du noeud correspondant aux noeuds de Luka de manière unique)
- Si on rencontre de nouveau un mot de Luka qui existe déjà dans la table de hachage, on fait pointer le noeud vers celui qui existe déjà dans la table et on supprime le doublon
- Sinon on stocke le couple (mot de Luka, noeud)
- Utilisation de **Terminal rule et Merging rule**



Terminal rule: on compresse les feuilles



Mergin rule: on compresse les noeuds internes

Sa complexité

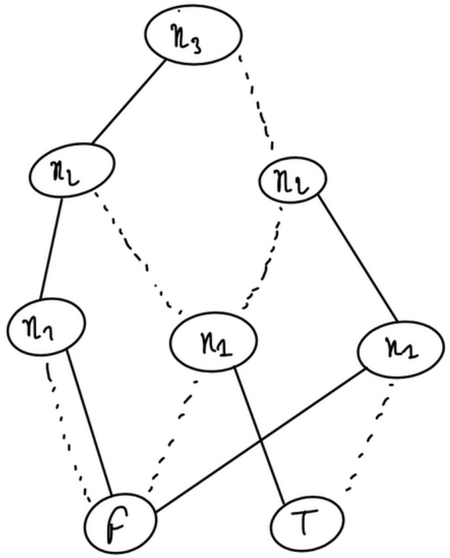
Pour un nombre de noeuds n, on parcourt uniquement 1 fois chaque noeud en O(n). On utilise une table de hachage pour stocker les couples déjà rencontrés O(1) en moyenne et en O(n) dans le pire des cas.

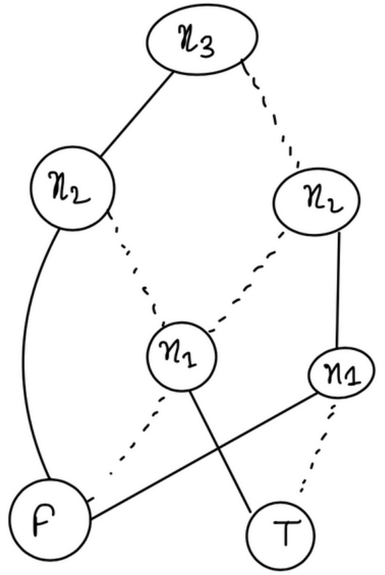
- Complexité moyenne: O(n)
- Complexité dans le pire cas: O(n^2)

La fonction de compression de ROBDD

Son fonctionnement :

- Si le fils gauche du fils gauche et le fils droit du fils gauche pointent vers le même noeud, alors on l'insère dans un tableau au cas ou un autre noeud le référence, on change donc le pointeur
- Le côté droit est symétrique
- À la fin du traitement on delete les noeuds qui ne sont plus utilisés
- Utilisation de **Deletion rule**





Sa complexité

- On parcourt chaque noeud 1 seule fois, O(n)
- Si jamais on doit stocker le noeud intermediaire (le fils) dans un tableau, O(1)
- On doit supprimer les noeuds intermédiares stockés dans la table alors c'est en O(1) mais comme on fait pour tous les noeuds alors c'est en O(n)
- On a donc une complexité de "O(n)+O(n)"=O(n)

Extension: Pool de Threads

- On a ajouté une option qui utilise les threads afin de réduire le temps de calcul
- implémentation d'une Pool de threads qui va permettre d'avoir plusieurs threads allant piocher dans une structure (une queue) des tâches (jobs) a exécuter (calculs des ROBDD).
- implémentation d'une synchronisation afin d'éviter les datas race (pour permettre les accès concurrents aux données).
- En utilisant les threads on a un temps de 4 secondes pour 5 variables à 500000 arbres et sans les threads on a un temps de 9 secondes
- Pour 5 variables, la distribution exacte prendrait plusieurs heures