Identification d'états se répétant infiniment souvent dans les structures de Kripke

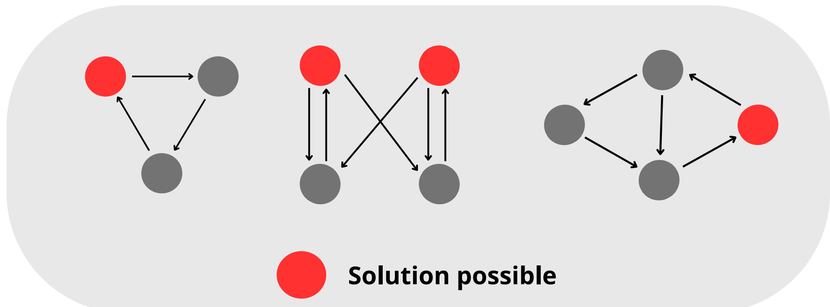
Boudjémâa Salah & Max Fillère
Encadrant : Johan Arcile

Contexte

Dans le contexte de la **vérification de modèle**, certaines techniques tirent profits de la cyclicité des espaces d'états. En particulier, la *Layer based exploration* nécessite l'identification d'un état ou d'un groupe[1] d'états **présent infiniment souvent** dans toutes les traces d'exécution.
Une propriété intéressante d'un tel groupe est que son retrait produit une structure acyclique. Ce problème correspond au problème du *Directed Feedback Vertex Set*

[1]Pour toute trace d'exécution, au moins un état du groupe est présent infiniement souvent

Le problème du **Directed Feedback Vertex Set** (DFVS) consiste à, trouver, dans un graphe orienté, un ensemble de sommets dont la suppression rend le graphe acyclique. Ce problème est **NP-complet**,

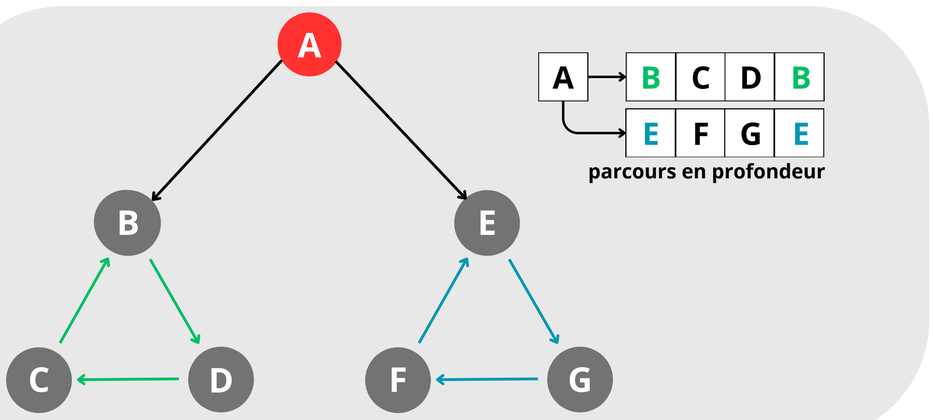


Objectif

L'objectif est d'implémenter un algorithme aussi performant que possible qui soit capable de donner, s'il existe, un état **présent infiniment souvent.** Si un tel état n'existe pas, l'algorithme doit donner un groupe d'états présents infiniment souvent, aussi petit que possible.

Calcul des cycles

La première étape à réaliser est d'implémenter un algorithme afin de retourner tous les **cycles** de la structure. Celui-ci est basé sur un **parcours en profondeur**. Il enregistre le chemin en cours et détecte un cycle dès qu'un état est visité une nouvelle fois.



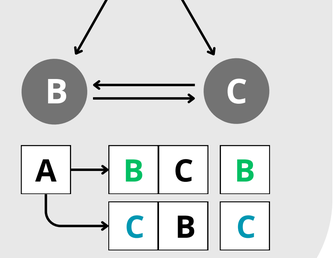

```
Chemin = [etat_initial]
Pile = [(etat_initial,0)]

Tant que pile non vide :
    Dépiler
    (E,i) = élément dépilé
    Si E à au moins i voisins :
        Empiler (E,i+1)
        Si V dans le chemin :  # cycle détecté
            Si cycle inconnu :
                Cycles += sous liste du chemin [V:fin]
        Sinon :
            Ajouter V au chemin
            Empiler (V,0)
    Sinon :
        Dépiler le chemin
Retourner Cycles
```

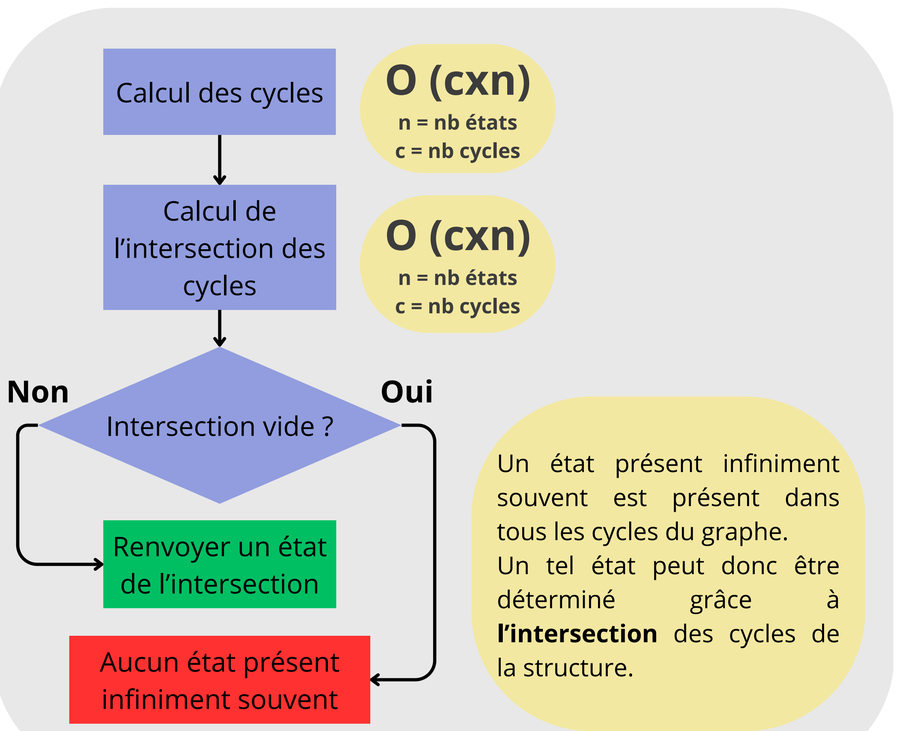
(E, i) :
E: état étudié,
i: rang de son voisin V dans sa liste de voisins

O(cxn)
n = nb états
c = nb cycles

On sélectionne les cycles **uniques** parmi les cycles trouvés. Un même cycle peut être accessible à partir de différents voisins

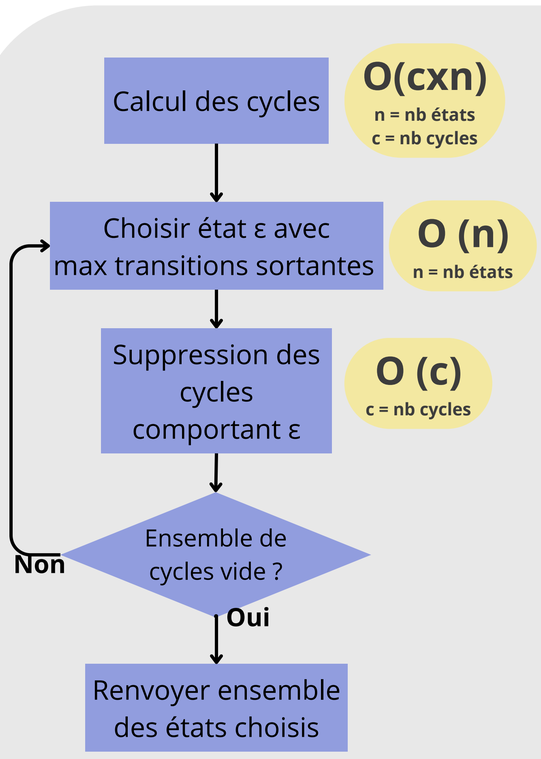


Détection d'un état présent infiniment souvent

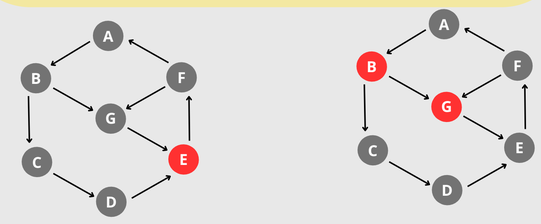


Un état présent infiniment souvent est présent dans tous les cycles du graphe.
Un tel état peut donc être déterminé grâce à **l'intersection** des cycles de la structure.

Détection d'un groupe d'états présents infiniment souvent



La méthode la plus simple est l'utilisation d'un algorithme **"glouton"**. C'est un algorithme qui par définition n'est pas optimal. Il se caractérise par une succession de choix optimaux **locaux**, sans tenir compte de la globalité du problème.



cycles : [ABCDEF, ABGEF, EFG]

t	1	2
ε	B	G
cycles	[EFG]	[]
états choisis	[B]	[G]

L'implémentation des algorithmes est faite en python