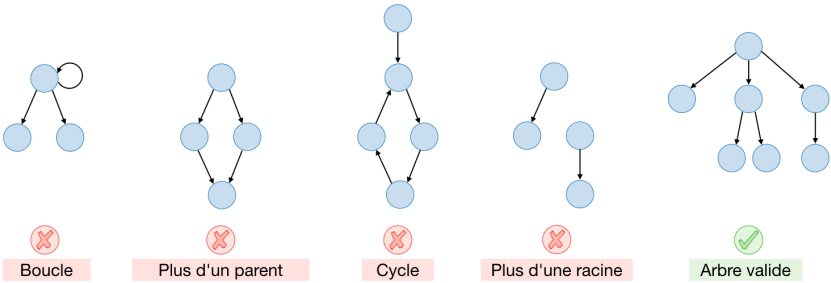
2 Modèles basés sur les états

2.1 Optimisation de parcours

Dans cette section, nous supposons qu'en effectuant une action a à partir d'un état s, on arrive de manière déterministe à l'état Succ(s,a). Le but de cette étude est de déterminer une séquence d'actions (a_1,a_2,a_3,a_4,...) démarrant d'un état initial et aboutissant à un état final. Pour y parvenir, notre objectif est de minimiser le coût associés à ces actions à l'aide de modèles basés sur les états (*state-based model* en anglais).

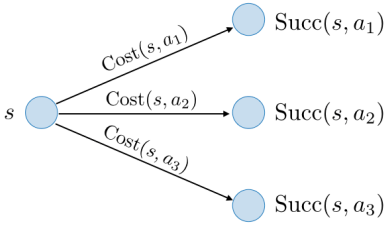
2.1.1 Parcours d'arbre

Cette catégorie d'algorithmes explore tous les états et actions possibles. Même si leur consommation en mémoire est raisonnable et peut supporter des espaces d'états de taille très grande, ce type d'algorithmes est néanmoins susceptible d'engendrer des complexités en temps exponentielles dans le pire des cas.



❒ **Problème de recherche** – Un problème de recherche est défini par :

- un état de départ s_{start}
- des actions Actions(s) pouvant être effectuées depuis l'état s
- le coût de l'action Cost(s,a) depuis l'état s pour effectuer l'action a
- le successeur Succ(s,a) de l'état s après avoir effectué l'action a
- la connaissance d'avoir atteint ou non un état final IsEnd(s)



L'objectif est de trouver un chemin minimisant le coût total des actions utilisées.

❒ **Retour sur trace** – L'algorithme de retour sur trace (en anglais *backtracking search*) est un algorithme récursif explorant naïvement toutes les possibilités jusqu'à trouver le chemin de coût minimal. Ici, le coût des actions peut aussi bien être positif que négatif.

❒ **Parcours en largeur (BFS)** – L'algorithme de parcours en largeur (en anglais *breadth-first search* ou *BFS*) est un algorithme de parcours de graphe traversant chaque niveau de manière successive. On peut le coder de manière itérative à l'aide d'une queue stockant à chaque étape