La recherche dans un espace d'états

Faculté de sciences et de génie
Département d'informatique et de génie logiciel
IFT-2003 Intelligence artificielle I
Laurence Capus

Plan

- Introduction
- Comment représenter un problème par un espace d’états?
- Comment faire une recherche dans un espace d’états?
- Conclusion

Introduction

- Nous aborderons les techniques de recherche intelligente utilisées dans le parcours de graphes.
- Mais pour parcourir un graphe avec une technique de recherche, il faut d'abord représenter le problème à résoudre par ce graphe.
- Une fois le problème représenté, nous allons consacrer un premier point à la recherche à l'aide de techniques de recherche exhaustives ou aveugles, qui ne font pas partie des techniques d'intelligence artificielle.
- Lorsque ce premier point sera bien compris, il sera plus aisé d'aborder les techniques de recherche intelligente, appelées aussi techniques de recherche heuristique.

Introduction

- Supposons que l'on se pose le problème de réaliser un système qui joue aux échecs. Pour ce faire, nous aurons à :
  - spécifier la **configuration de départ** de l'échiquier, les **règles** qui définissent des mouvements légaux, les **configurations de l'échiquier qui représentent une victoire**, pour l'un et l'autre des adversaires ;
  - rendre **explicite** le but implicite de jouer un coup légal mais aussi de gagner la partie, si possible.
- Il est facile, pour le jeu d'échecs, de donner une description formelle et complète du problème :
  - La **configuration initiale** peut se décrire à l'aide d'un tableau de 8 x 8 positions, où chacune d'entre elles contient un symbole correspondant à la pièce appropriée selon la configuration d'ouverture officielle.
  - Nous pouvons définir comme **but**, toute configuration de l'échiquier pour laquelle l'adversaire ne peut réaliser un mouvement légal et pour lequel son roi est en échec.
  - Les mouvements légaux nous permettent de passer de la configuration initiale à la configuration but ou finale. Ces mouvements légaux peuvent facilement être décrits par un ensemble de **règles** constituées de deux parties : une partie **gauche** qui sert de patron pour être assorti avec la configuration courante de l'échiquier et une partie **droite** qui traduit le changement à faire à la configuration pour refléter le mouvement.

Introduction

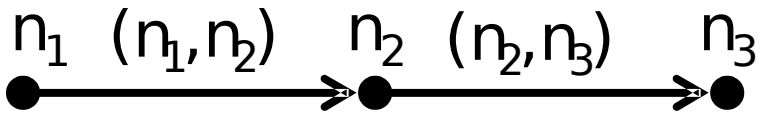
- En pratique, il y aurait, approximativement, 10^{120} règles différentes à écrire correspondant aux 10^{120} configurations possibles des pièces d'un échiquier. Un aussi grand nombre de règles à écrire poserait un sérieux problème. Personne ne serait capable de le faire. Cela prendrait trop de temps et ne se ferait certainement pas sans fautes. Aucun programme ne pourrait manipuler toutes ces règles (temps de calcul et espace mémoire).
- Pour minimiser ce genre de problème, il nous faut trouver un moyen d'écrire les règles décrivant les mouvements légaux d'une façon la plus générale possible.
- Nous avons décrit le problème du jeu d'échecs comme un problème de déplacements dans un **espace d'états**, où chaque état correspond à une position légale de l'échiquier. On peut donc, à présent, **jouer aux échecs** à partir d'un **état initial**, en utilisant un ensemble de **règles** pour passer d'un état à un autre et essayer d'arriver à un des **états finaux**.

Introduction

- Cette représentation du jeu d'échecs nous semble naturelle parce que l'ensemble des états, qui correspond à l'ensemble des configurations de l'échiquier, est artificiel et bien organisé. Ce même type de représentation est aussi utile pour des problèmes moins bien structurés bien qu'il puisse être nécessaire d'utiliser des structures plus complexes qu'une matrice pour décrire un état donné.
- La représentation d'un problème par un espace d'états est à la base de la plupart des techniques d'intelligence artificielle que nous allons considérer. Sa structure correspond à une structure de résolution de problèmes sur deux points:
  - Elle autorise la définition formelle d'un problème comme étant la nécessité de **convertir une situation donnée en une situation souhaitée** en utilisant un ensemble d'opérations permises.
  - Elle permet de définir le processus de résolution d'un problème particulier comme **une combinaison de techniques connues** (chacune représentée par **une règle définissant une transition** dans l'espace) et une **recherche** (la technique générale d'exploration de l'espace pour **essayer de trouver un trajet de l'état courant à l'état final**).

Définition

- Plus formellement, la recherche dans un espace d'états est basée sur la théorie des graphes de L. Euler (1736).
- Un graphe consiste en :
  - un ensemble fini ou infini de nœuds n_1, n_2, ...
  - un ensemble d'arcs (n_i,n_j).
- Exemple :



- Dans un graphe **étiqueté**, un descripteur est associé à chaque nœud, ce qui permet de le distinguer parmi les autres nœuds du graphe.

Définition

- La théorie des graphes permet de représenter l'espace d'états d'un problème :
  - Un **espace d'états** est représenté par un tuple de 4 éléments soit [N,A,S,GD] où :
    - N: { Nœuds / états } : ensemble de nœuds/états
    - A: { Arcs / opérations } : ensemble d'arcs/opérations
    - S: $S \subseteq N$, $S \neq \{\ \}$, $S = \{EI\}$: nœud de départ/état initial
    - GD: $GD \subseteq N$, $GD \neq \{\ \}$, $GD = \{EF\}$: nœud(s) d'arrivée/état(s) final(finaux)
  - Le chemin de la solution est : EI - - - - - - - - EF (une liste d'états).

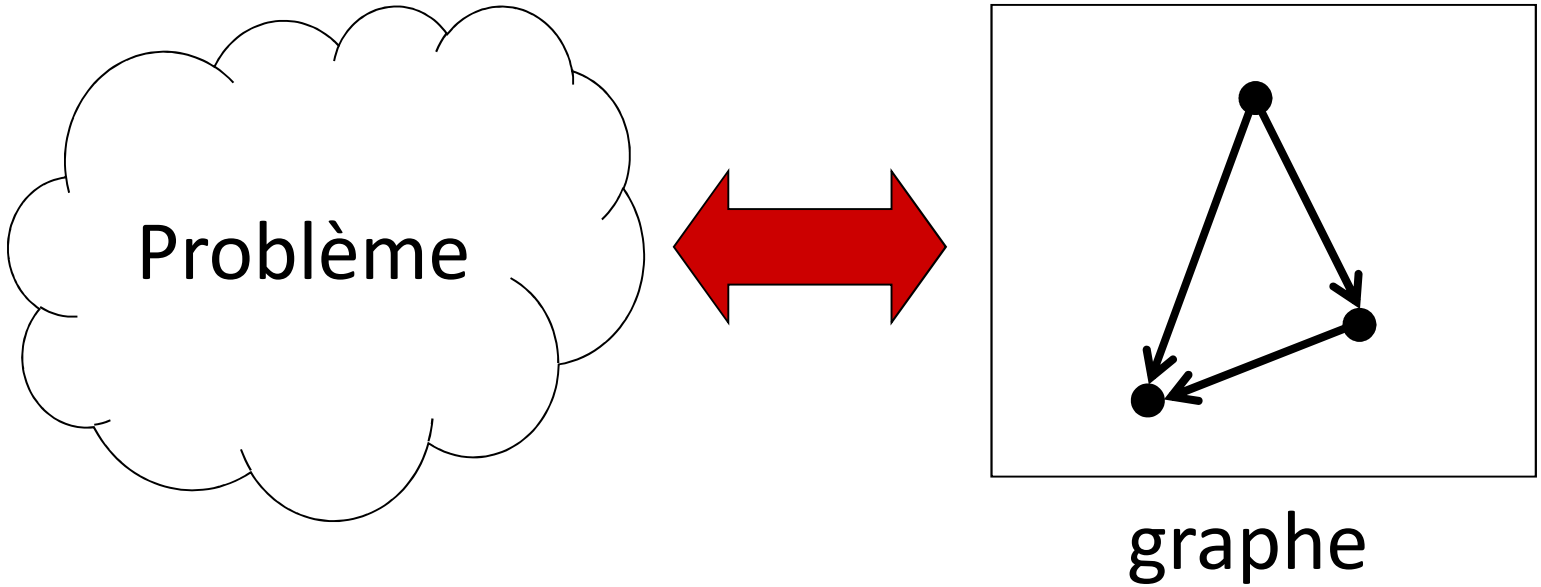
➔ Résoudre un problème représenté par un graphe revient à rechercher un trajet (suite d'arcs) du nœud de départ à un nœud d'arrivée.

Exemples

- Quelques exemples simples de représentation de problèmes par un graphe sont présentés en suivant.
- Les jeux sont de bons exemples et ont servi à de nombreuses recherches en intelligence artificielle pour tester de nouvelles techniques de recherche plus efficaces que les techniques exhaustives. Le jeu du tic-tac-toe sera d'ailleurs utilisé pour illustrer la représentation par un espace d'états.

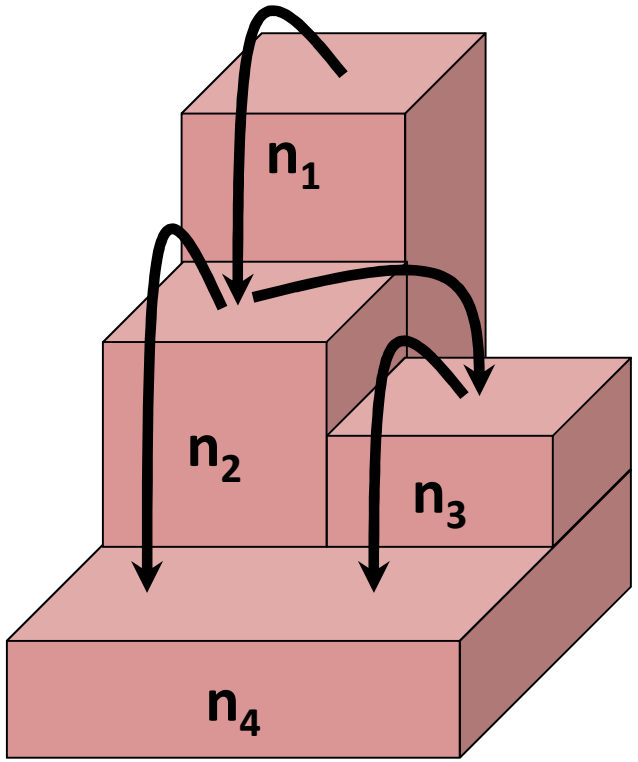
Exemples (1/6)

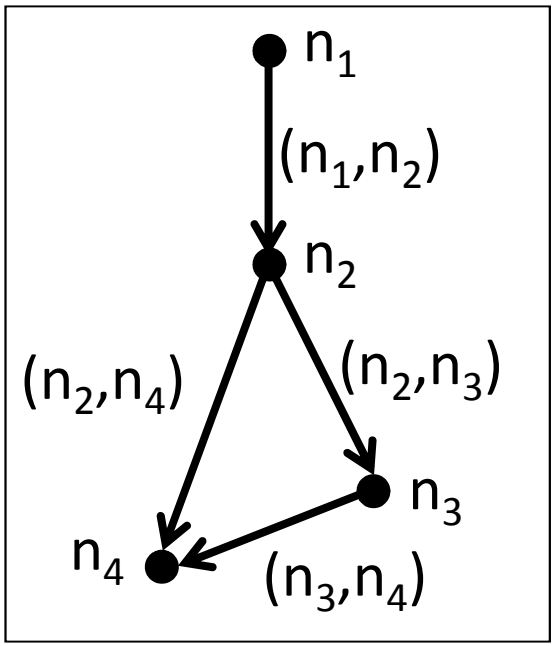
- Exemple général de modélisation d'un problème à résoudre par un graphe
  - Un nœud = un état
  - Un arc = une opération pour changer d'état



Exemples (2/6)

Exemple 1 : Comment descendre d'une série de blocs?

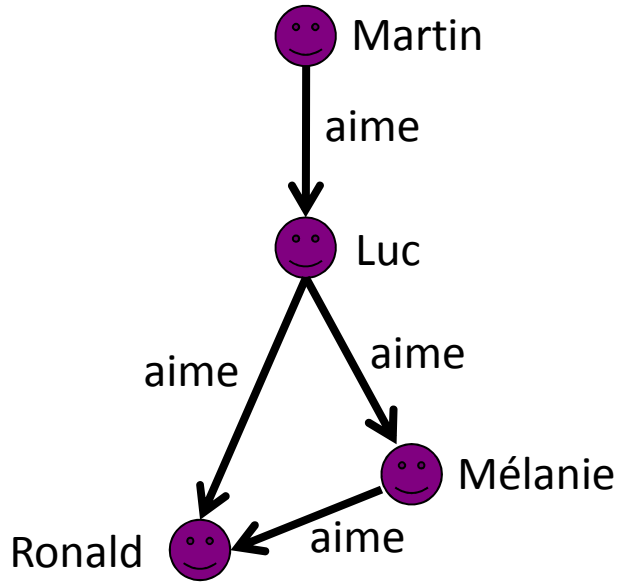




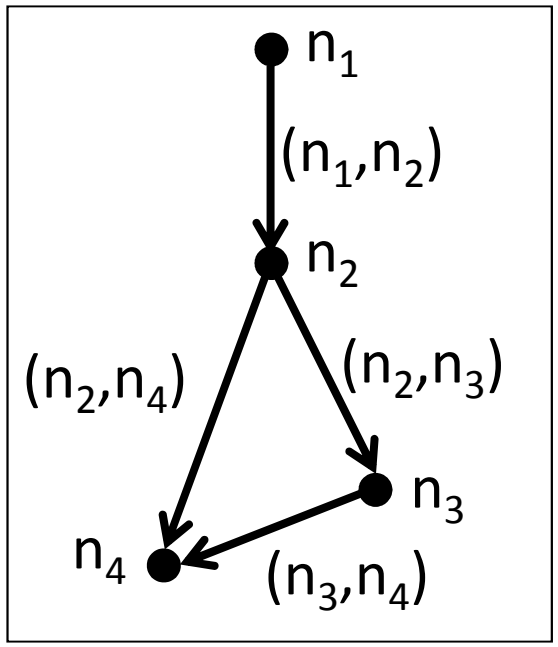
graphe B

Exemples (3/6)

Exemple 2 : Une relation d'amitié.



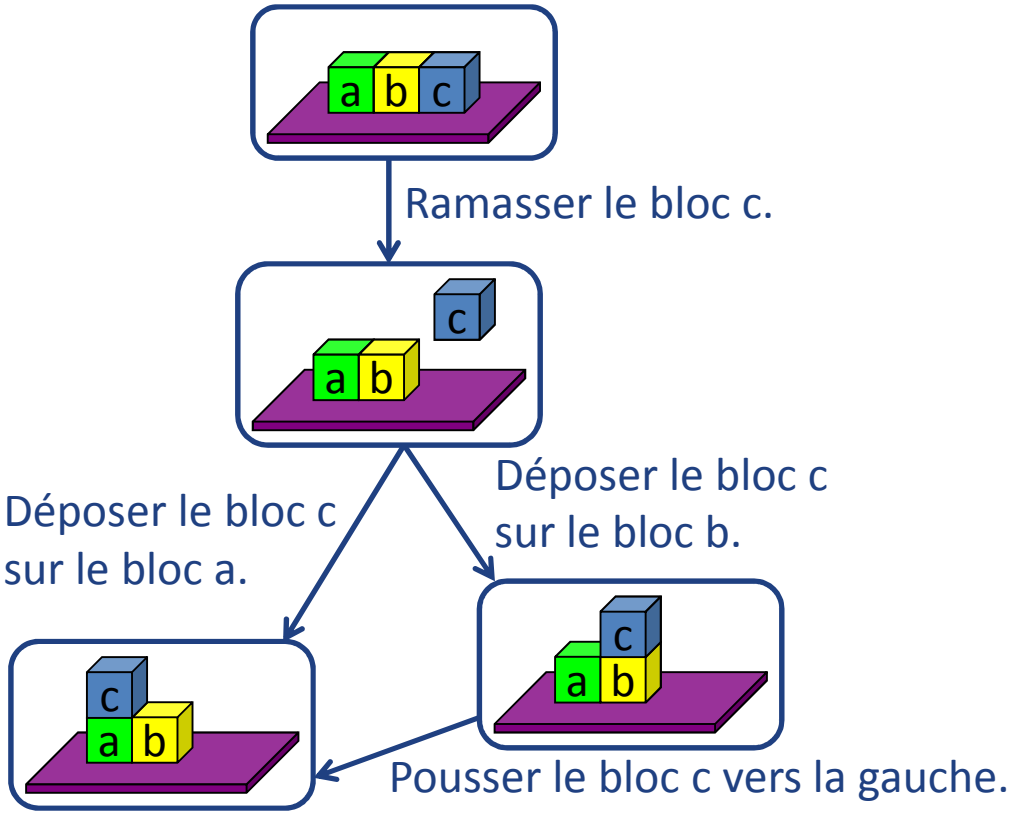
Martin aime Luc. Luc aime Ronald et Mélanie. Mélanie aime Ronald.

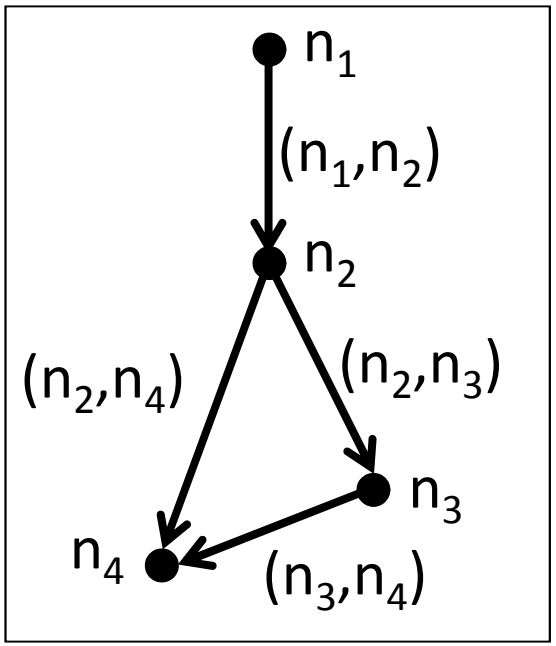


graphe B

Exemples (4/6)

Exemple 3 : Déplacer des blocs.

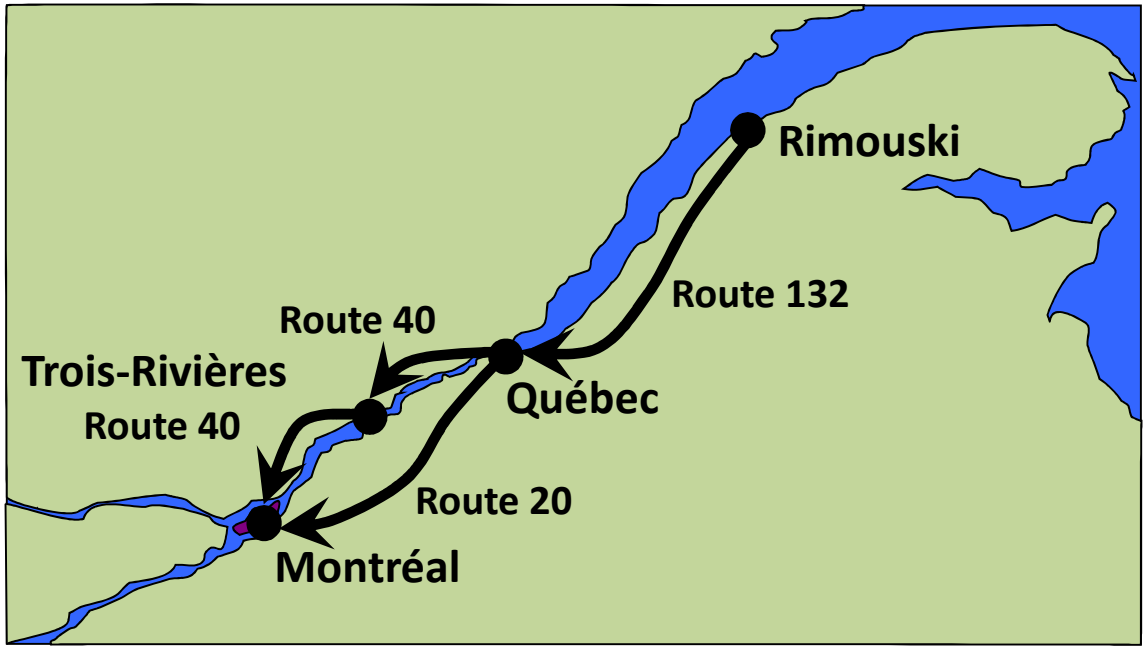


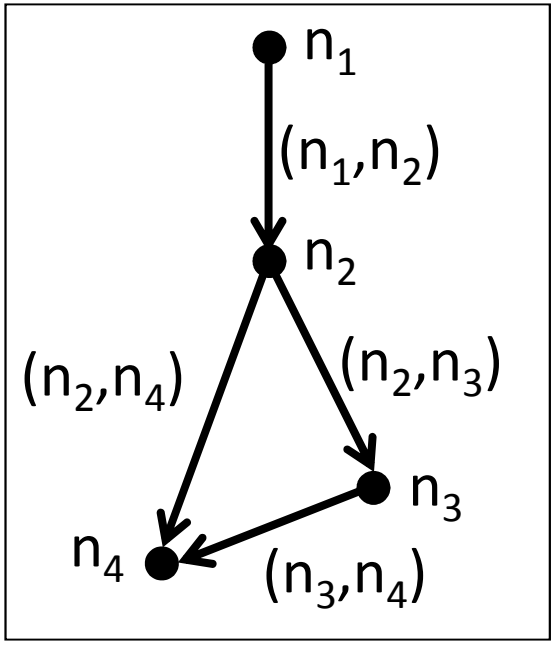


graphe B

Exemples (5/6)

Exemple 4 : Voyager de Rimouski à Montréal (Québec).





graphe B

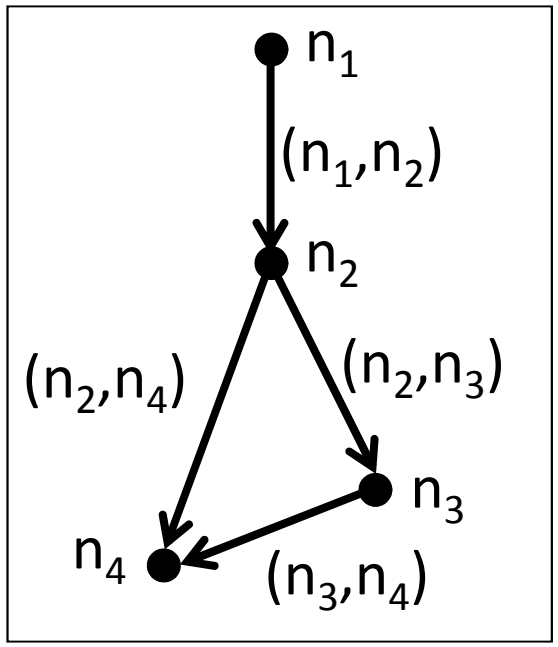
Exemples (6/6)

Exemple 5 : Un système logique.

$N1 \rightarrow N2$

$N2 \rightarrow (N3 \wedge N4)$

$N3 \rightarrow N4$

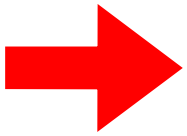


graphe B

Recherche dans un espace d'états (1/5)

Pour résoudre un problème en utilisant la recherche dans un espace d'états, il faut pouvoir répondre aux questions suivantes :

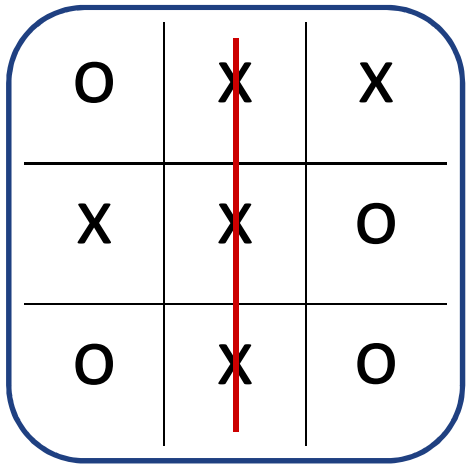
- Comment représenter un état? Quels **patrons** de symboles utiliser?
- Comment passer d'un état à un autre? Quelles **opérations** permettent de modifier un état représenté par un patron donné?
- Comment choisir la prochaine opération? Quelle **stratégie** permet de privilégier une opération plutôt qu'une autre lorsqu'il y a plusieurs alternatives?



Cette résolution correspond à l'hypothèse du système physique de symboles (posée par Newell & Simon, 1976)

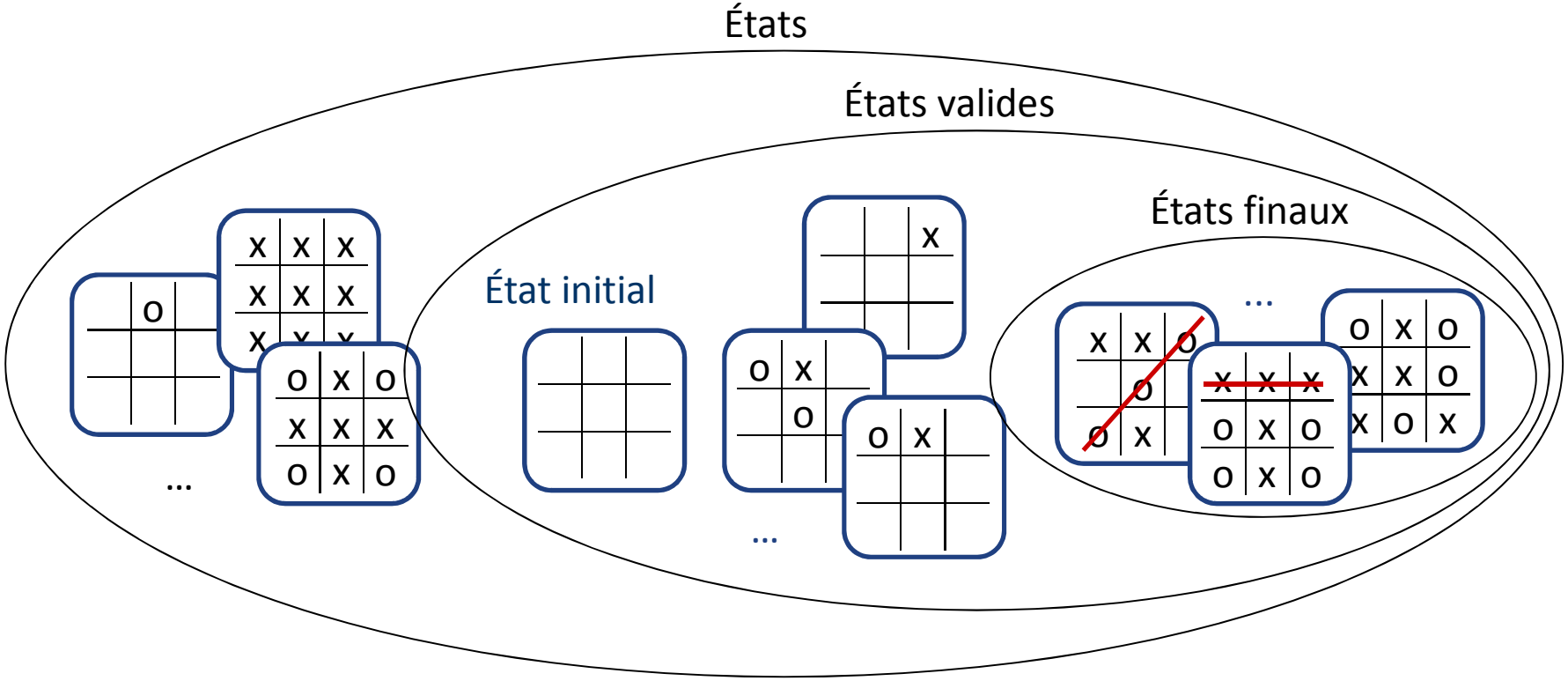
Recherche dans un espace d'états (2/5)

- Exemple : Jouer au tic-tac-toe
  - **Patrons**: Représentation du jeu, des coups,…
  - **Opérations**: Coups possibles
  - **Stratégie**: Sélection d'un coup



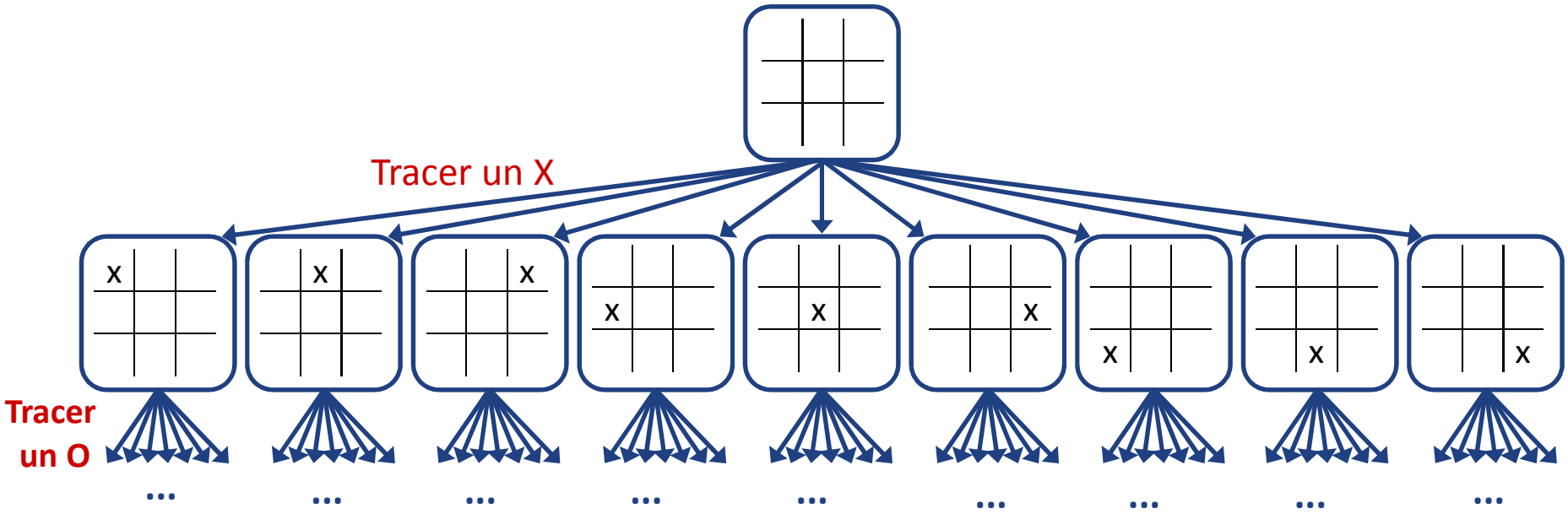
Recherche dans un espace d'états (3/5)

– Espace d'états



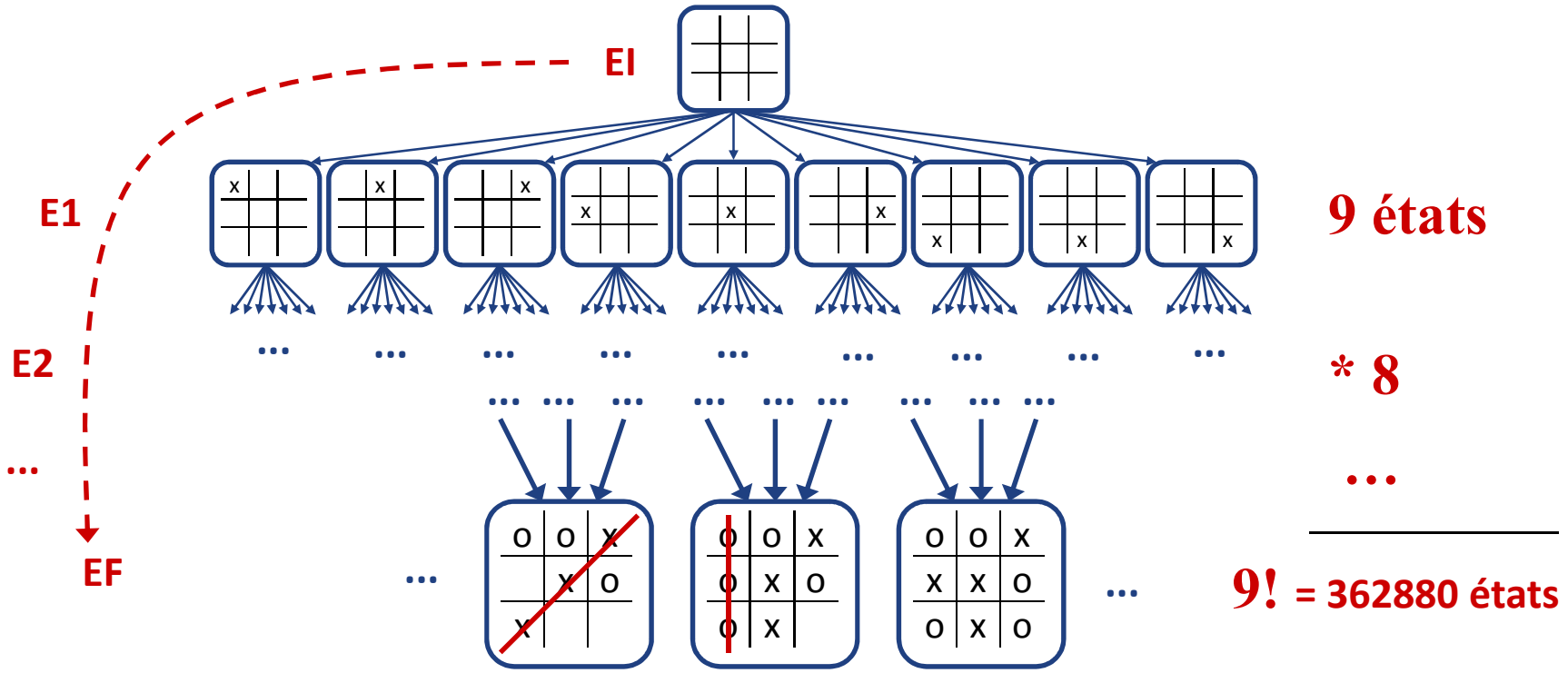
Recherche dans un espace d'états (4/5)

– Opérations possibles



Recherche dans un espace d'états (5/5)

– Taille de l'espace d'états

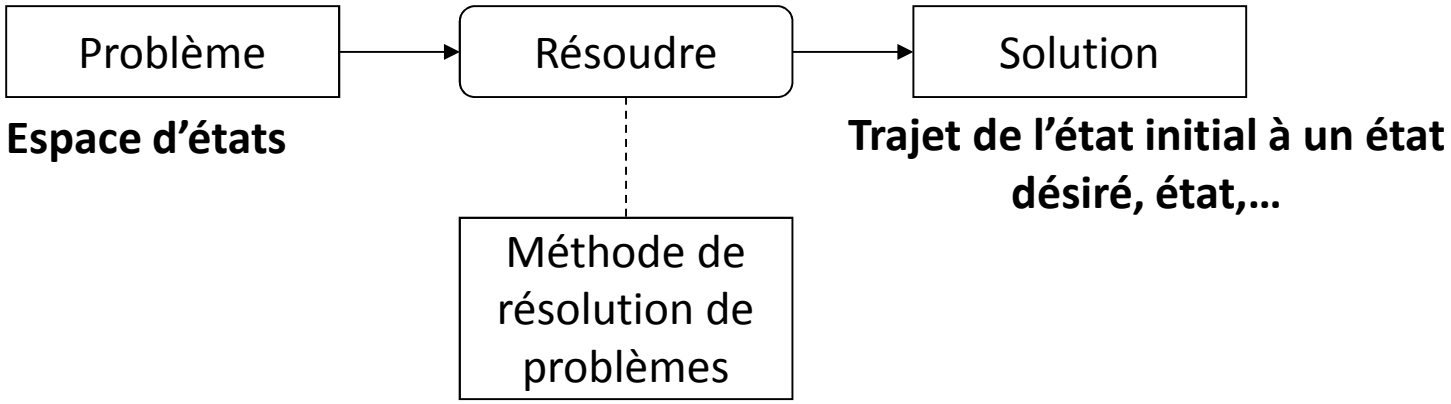


Techniques de recherche exhaustive
(ou techniques aveugles)

- Stratégies de recherche
- Recherche dirigée par les données
- Recherche dirigée par les buts
- Méthode générer-tester

Stratégies de recherche (1/3)

Une fois le problème modélisé par un espace d'états, il faut déterminer comment le résoudre :

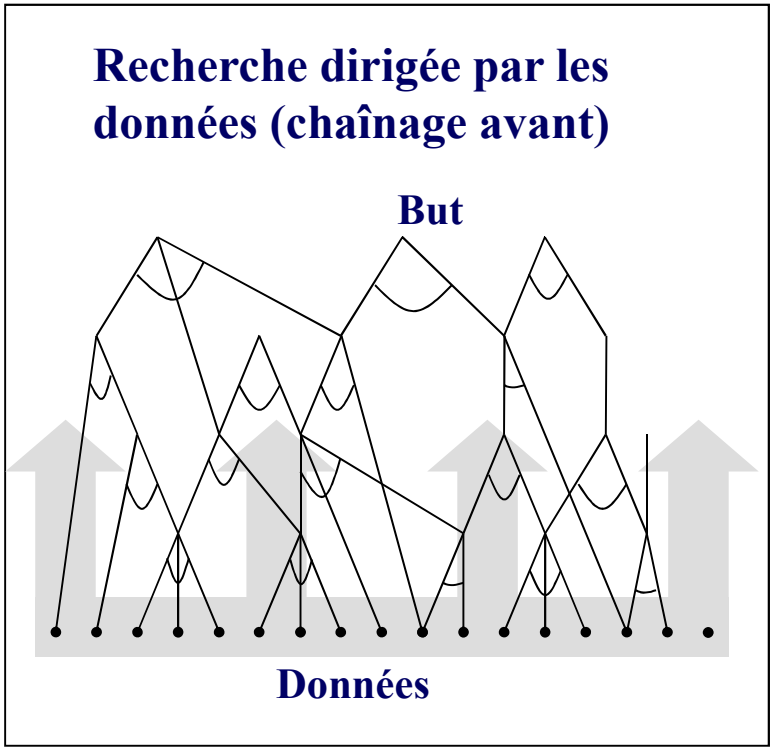


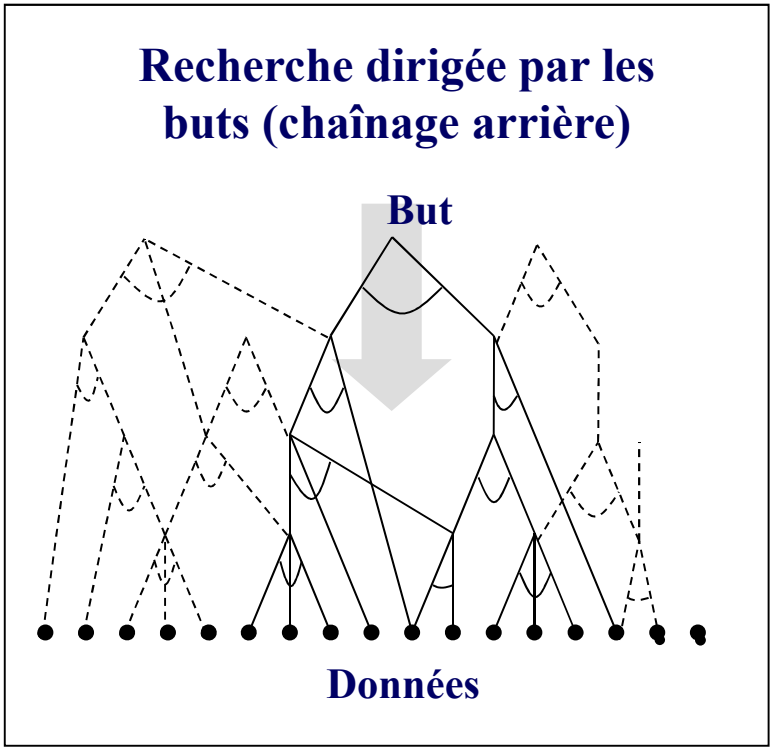
Comment résoudre le problème?

On a besoin d'une stratégie de recherche dans un espace d'états.

Stratégies de recherche (2/3)

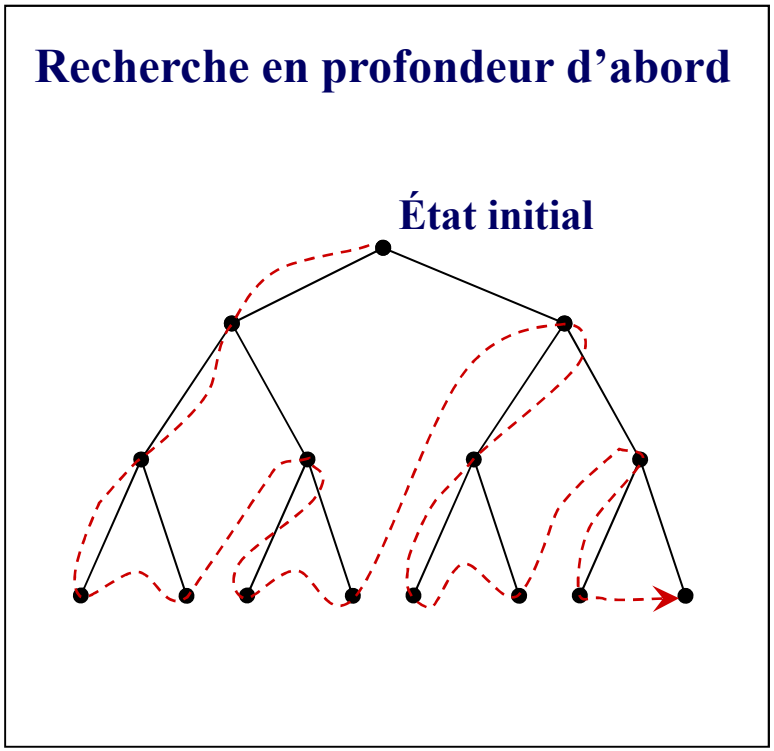
- Dans quelle direction ?
  - Recherche dirigée par les données ou par les buts

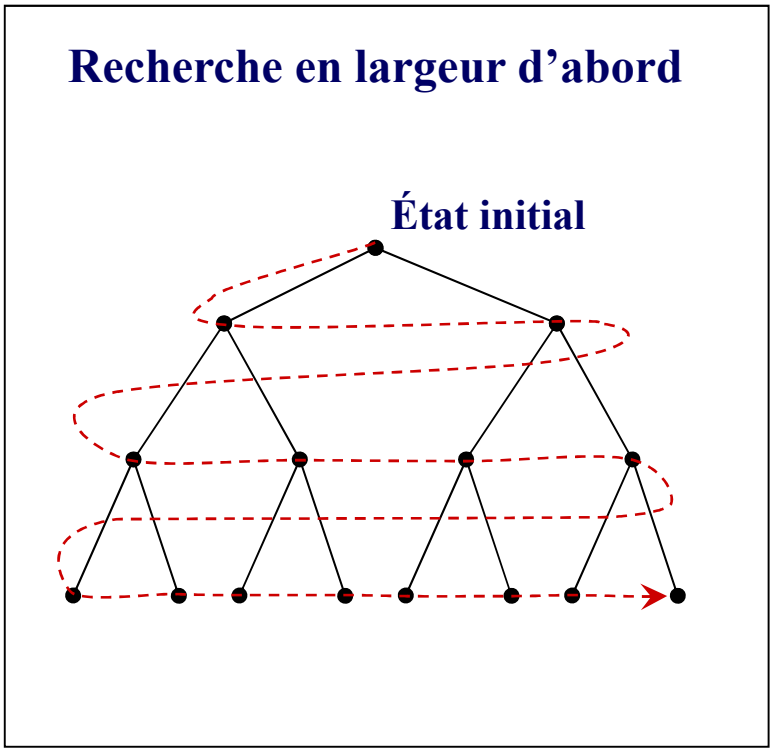




Stratégies de recherche (3/3)

- Dans quel ordre?
  - Recherche en profondeur ou en largeur d'abord (rappel)





Recherche dirigée par les données (1/3)

- Procédure de résolution de problèmes :

1. Considérer l'ensemble des faits/l'état courant
2. Déterminer l'ensemble des règles/opérations applicables, (c.-à-d. dont les conditions sont respectées dans l'état courant)
   - S'il existe des règles applicables
     a. Choisir une règle/opération applicable
     b. Appliquer la règle/l'opération, c'est-à-dire produire le nouveau fait/le nouvel état
     c. Recommencer en 1
   - Sinon, arrêter (plus de nouveaux faits à inférer)

Recherche dirigée par les données (2/3)

- Exemple : un système logique

règles

$a \wedge b \rightarrow e$

$c \wedge d \rightarrow f$

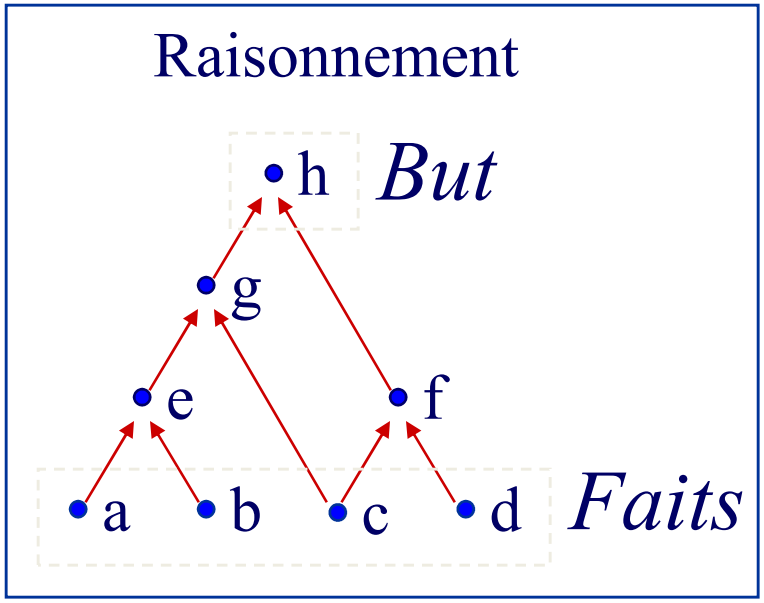
$e \wedge c \rightarrow g$

$f \wedge g \rightarrow h$

Faits

a, b, c, d

But : **h**



BUTS
(solution)

DONNÉES
(faits, hypothèses)

Recherche dirigée par les données (3/3)

- Exemple : un système logique (trace)

règles

R1: a ∧ b → e
R2: c ∧ d → f
R3: e ∧ c → g
R4: f ∧ g → h

Faits

a, b, c, d

But : **h**

#	Base de faits	Règles candidates	Règle choisie	Nouveau fait
1	a,b,c,d	R1,R2	R1	e
2	a,b,c,d,e	R1,R2,R3	R2	f
3	a,b,c,d,e,f	R1,R2,R3	R3	g
4	a,b,c,d,e,f,g	R1,R2,R3,R4	R4	h
5	a,b,c,d,e,f,g,h	R1,R2,R3,R4	aucune	aucun

Lorsque plusieurs règles sont candidates, la **stratégie de résolution de conflits** appliquée ici est de choisir la première règle candidate dont la conclusion n'est pas déjà dans la base de faits.

Recherche dirigée par les buts (1/3)

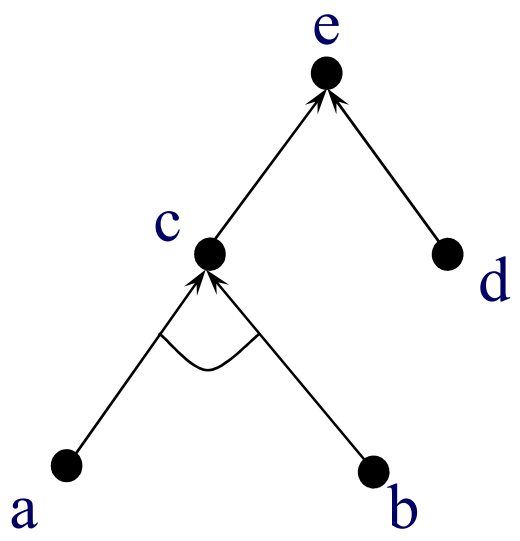
- Un graphe et/ou :
  - Extension d'un graphe conventionnel permettant de représenter les conjonctions, disjonctions et implications dans une expression logique
  - Exemple

Règles:

$a \wedge b \rightarrow c$

$c \vee d \rightarrow e$

Graphe et/ou:



Recherche dirigée par les buts (2/3)

- Procédure de résolution de problèmes :
  - Considérer le but à prouver
    1. Si le but est un fait, alors le but est prouvé
    2. Si le but est une conjonction de sous-buts, alors tenter de prouver tous les sous-buts
       - Si tous les sous-buts sont prouvés, alors le but est prouvé, sinon, le but ne peut pas être prouvé
    3. Si le but est une disjonction de sous-buts, alors tenter de prouver un des sous-buts
       - Si un des sous-buts est prouvé, alors le but est prouvé, sinon, le but ne peut pas être prouvé
    4. Si le but est la conclusion d'une règle, alors tenter de prouver les conditions de cette règle
       - Si les conditions de la règle peuvent être prouvées, alors le but est prouvé, sinon, choisir une autre règle
         - » Si aucune autre règle n'a pour conclusion le but à prouver, alors le but ne peut pas être prouvé

Recherche dirigée par les buts (3/3)

- Exemple : un système logique

règles

$a \wedge b \rightarrow e$

$c \wedge d \rightarrow f$

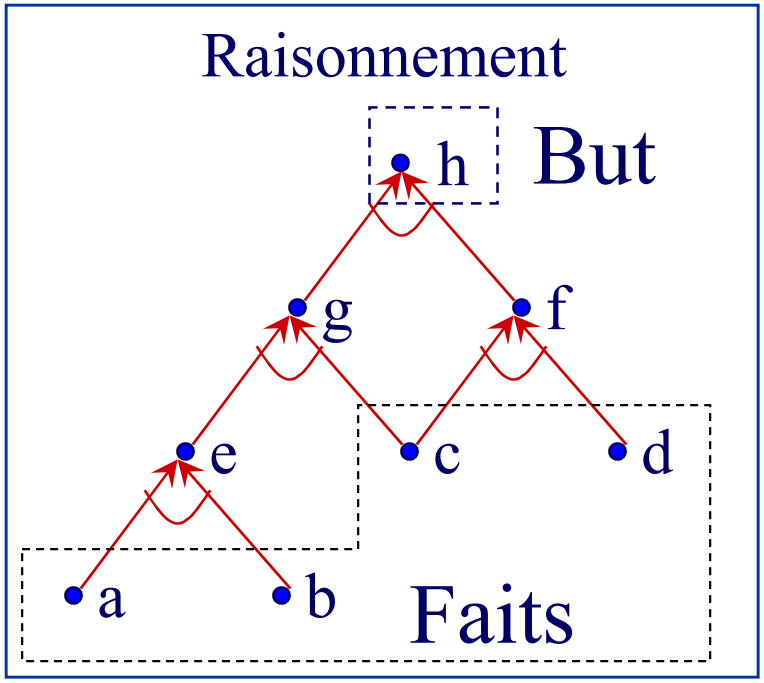
$e \wedge c \rightarrow g$

$f \wedge g \rightarrow h$

Faits

a, b, c, d

But : **h**



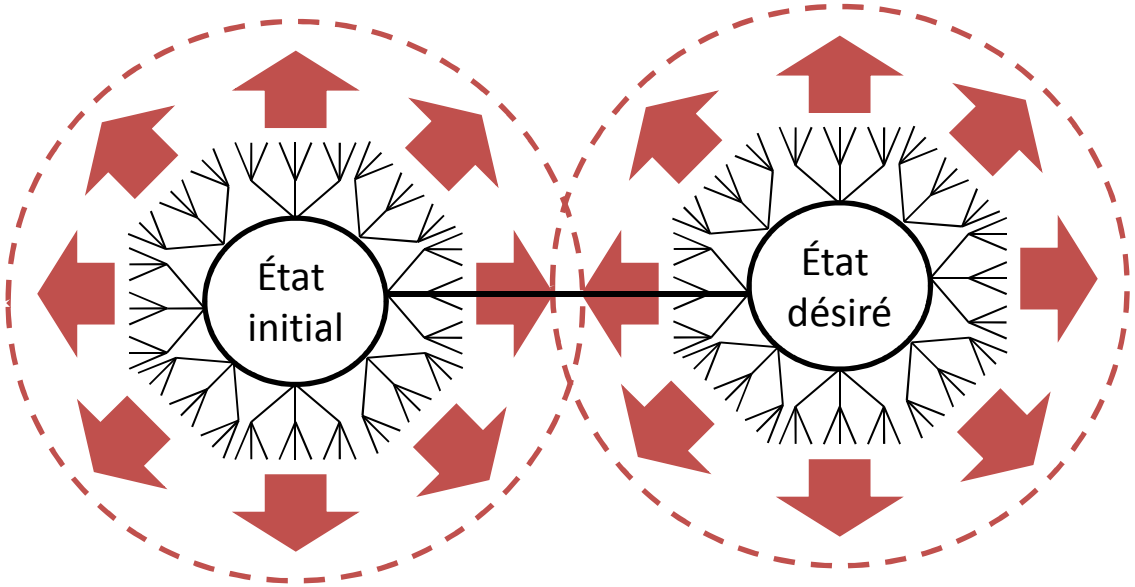
BUTS
(solution)

DONNÉES
(faits, hypothèses)

Combinaison buts + données

Avancer jusqu'à une profondeur de M à partir des données

Reculer jusqu'à une profondeur de N à partir des buts



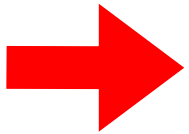
Méthode générer-tester (1/2)

- Comment résoudre un problème?

1. Générer un état qui n'a pas encore été testé.
2. Tester l'état généré:

   S'il s'agit d'un état final, arrêter.

   Sinon, ajouter l'état à l'ensemble des états déjà générés et retourner à l'étape 1.



Besoins :

- On sait à quoi ressemble un état (patrons).
- On sait comment changer d'état (opérations).
- On sait reconnaître un état final.

Méthode générer-tester (2/2)

- Maintenir deux listes
  - Une liste OUVERTS d'états générés
  - Une liste FERMÉS d'états testés
  - Initialement, OUVERTS ne contient que l'état initial

1. Si OUVERTS est vide, alors la recherche est un échec
2. Sinon
   - Si le premier état E membre de OUVERTS est un état final,
   - alors la recherche est un succès
   - Sinon,
     - Générer les fils de E
     - Les ajouter dans OUVERTS s'ils n'y sont pas déjà ou s'ils ne sont pas dans FERMÉS, ajouter E à FERMÉS
     - Recommencer en 1.

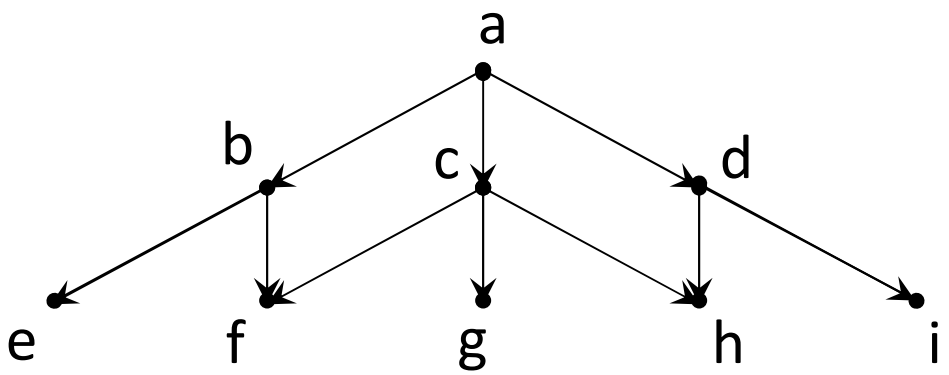
Recherche en profondeur d'abord (1/2)

Algorithme :

```
OUVERTS := [État initial];
FERMES := [];
Tant que OUVERTS ≠ [], faire :
        Enlever le premier état X de OUVERTS;
        Si X est un but, alors retourner SUCCÈS
        Sinon
                Générer les fils de X;
                Mettre X dans FERMES;
                Éliminer les fils de X qui sont déjà dans OUVERTS ou FERMES;
                Mettre les fils restant au début de OUVERTS;
Retourner ÉCHEC (il ne reste plus d'états à tester)
```

Recherche en profondeur d'abord (2/2)

Exemple



Suite des états visités : abefcghdi

OUVERTS	FERMÉS
a	
bcd	a
efcd	ab
fcd	abe
cd	abef
ghd	abefc
hd	abefcg
d	abefcgh
i	abefcghd
	abefcghdi

Recherche en largeur d'abord (1/3)

- On utilise la liste OUVERTS comme une file
- Initialement, OUVERTS ne contient que l'état initial.

1. Si OUVERTS est vide, alors la recherche est un échec
2. Sinon
   - Si le premier état E membre de OUVERTS est un état final, alors la recherche est un succès
   - Sinon,
     - Générer les fils de E
     - Les ajouter à la fin de OUVERTS s'ils n'y sont pas déjà ou s'ils ne sont pas dans FERMÉS, ajouter E à FERMÉS
     - Recommencer en 1.

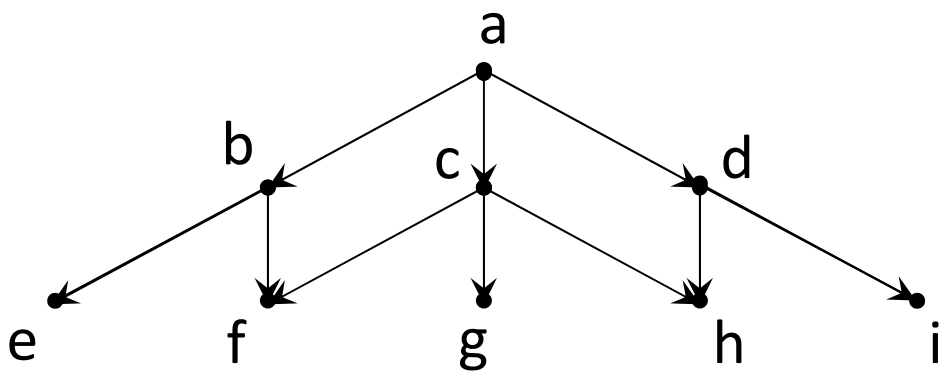
Recherche en largeur d'abord (2/3)

Algorithme

```
OUVERTS := [État initial];
FERMES := [];
Tant que OUVERTS ≠ [], faire :
        Enlever le premier état X de OUVERTS;
        Si X est un but, alors retourner SUCCÈS
        Sinon
                Générer les fils de X;
                Mettre X dans FERMES;
                Éliminer les fils de X qui sont déjà dans OUVERTS ou FERMES;
                Mettre les fils restant à la fin de OUVERTS;
Retourner ÉCHEC (il ne reste plus d'états à tester)
```

Recherche en largeur d'abord (3/3)

Exemple

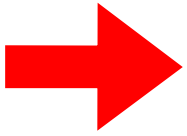


Suite des états visités : abcdefghi

OUVERTS	FERMÉS
a	
bcd	a
cdef	ab
defgh	abc
efghi	abcd
fghi	abcde
ghi	abcdef
hi	abcdefg
i	abcdefgh
	abcdefghi

Recherche en profondeur d'abord avec approfondissement itératif

- C'est un compromis entre recherche en profondeur et en largeur d'abord, soit en faisant une recherche en profondeur jusqu'à un certain niveau maximum. Les états suivants, générés à partir de l'état au niveau le plus profond, sont ajoutés à la fin de la liste des états OUVERTS.
- Avantages : Utile lorsqu'il est connu que la solution se trouve à une certaine profondeur ou lorsque des contraintes de temps limitent le nombre d'états qui peuvent être considérés.
- Limites : Comme toutes les techniques de recherche exhaustive, le temps total pour la recherche est, dans le pire cas, exponentiel en fonction de la taille des données en entrée.



Besoin d'une technique de recherche « informée »

Conclusion

– Représentation du problème à résoudre à l'aide de la recherche dans un espace d'états

- choisir le(s) symbole(s) pour représenter un état
- représenter l'état initial
- représenter l'état final ou les états finaux
- déterminer les opérations pour passer d'un état à un autre
- configurer les opérations sous forme de règles
  précondition(s) : partie gauche de la règle ou conditions d'application de la règle
  postcondition(s) : partie droite de la règle ou résultat d'application de la règle

- Techniques de recherche exhaustives: elles visitent tout l'espace d'états jusqu'à trouver un état final