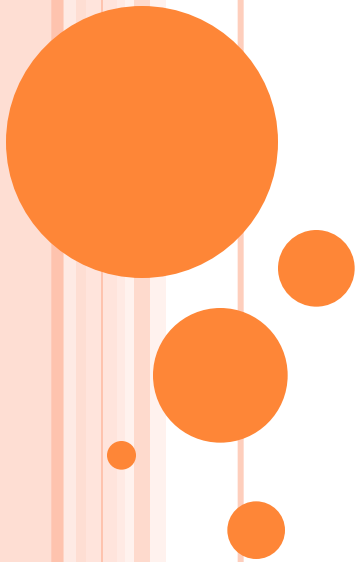


APPLICATIONS D'IA EN PROLOG



ELIZA

- Classique des débuts de l'IA
- N'utilise absolument pas d'IA
- Est censé simuler une conversation avec un psychologue

ELIZA : UN EXEMPLE DE CONVERSATION

?- eliza.

Bonjour. Qu est ce qui vous amene ?

|: "je suis triste".

depuis combien de temps etes vous triste ?

|: "depuis le depart de mon frere".

pouvez vous me parler de votre frere ?

|: "j aime discuter avec lui".

quelqu un d autre dans votre famille aime t il cela ?

|: "non".

continuez

|: "je me sens seule maintenant".

ressentez vous souvent cela ?

|: "oui, depuis que mon frere est parti !".

pouvez vous me parler de votre frere ?

|: "vous vous moquez de moi ?".

continuez

|: "bye".

Au revoir. J espere vous avoir aide

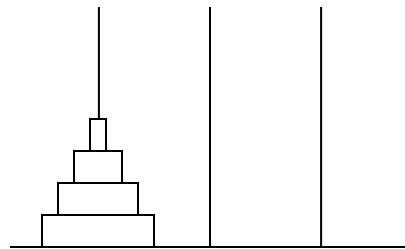
ELIZA : PRINCIPE DU PROGRAMME

- Reconnaître une configuration de mots
- Répondre en utilisant une configuration de réponse correspondante
- On définit donc des paires stimulus/réponse, par exemple :
 - je suis X
 - depuis combien de temps êtes vous X ?
- On cherche des mots-clés (père, mère, etc.) et on y associe une réponse

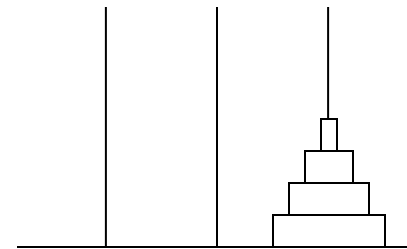
ELIZA : ALGORITHME

- Lire une phrase
- Choisir une paire (stimulus, réponse)
- Apparier la phrase et le stimulus
- Écrire la réponse associée
- Recommencer

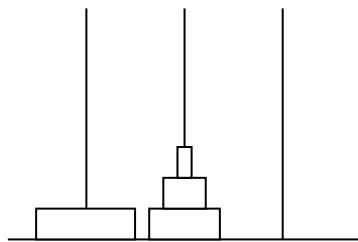
TOURS DE HANOI : DÉCOMPOSER UN PROBLÈME EN SOUS-PROBLÈMES



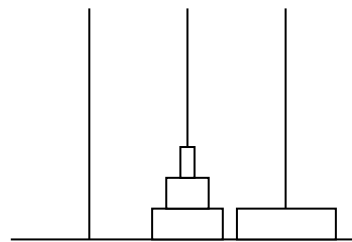
Situation initiale



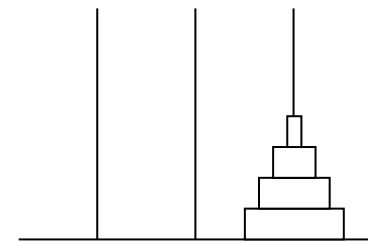
Objectif



Sous-problème 1



Sous-problème 2



Sous-problème 3

DÉCOMPOSITION D'UN PROBLÈME EN SOUS-PROBLÈMES

- Règles de décomposition :

R1 : $a \rightarrow b, c$

R2 : $d \rightarrow a, e, f$

R3 : $d \rightarrow a, k$

R4 : $f \rightarrow i$

R5 : $f \rightarrow c, j$

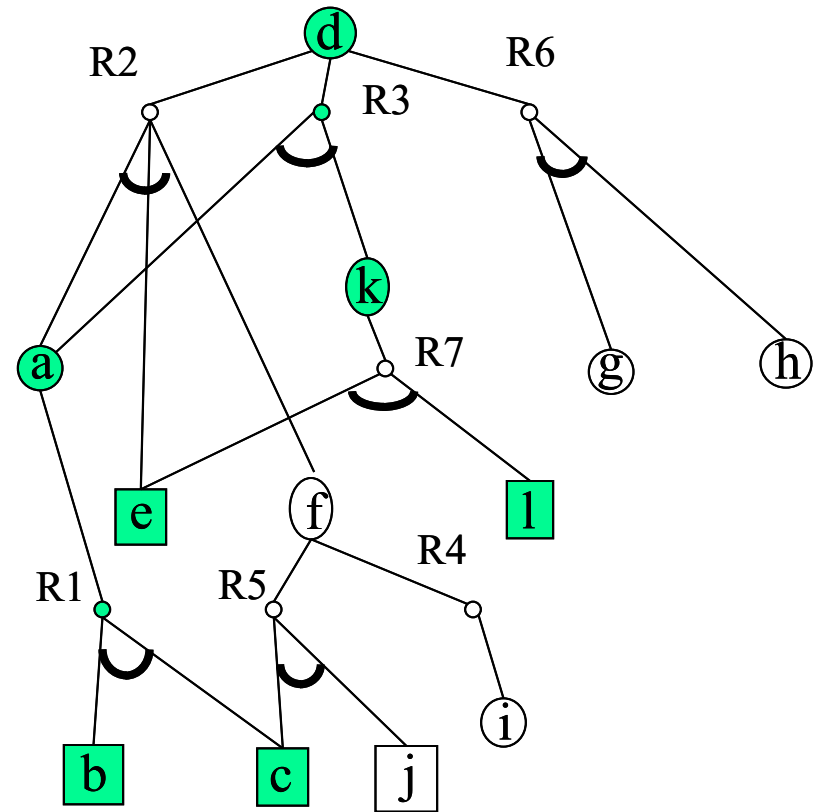
R6 : $d \rightarrow g, h$

R7 : $k \rightarrow e, l$

- Problèmes terminaux :

b, c, e, j, l

- Problème à résoudre : d



PROBLÈMES DE SATISFACTION DE CONTRAINTES

○ Définition :

- $X = \{X_1, X_2, \dots, X_n\}$ est l'ensemble des variables du problème
- On associe à chaque variable X_i son domaine $D(X_i)$, l'ensemble des valeurs que peut prendre X_i
- $C = \{C_1, C_2, \dots, C_k\}$ est l'ensemble des contraintes
Chaque contrainte C_j est une relation entre certaines variables de X , restreignant les valeurs que peuvent prendre simultanément ces variables

○ Exemple : cryptarithmétique, problème du zèbre

EXEMPLE : LE PROBLÈME DES REINES

- Placer N reines sur un échiquier $N \times N$ sans qu'aucune reine ne puisse en prendre une autre.
- Exemple pour $N=4$:

	R		
			R
R			
		R	

REPRÉSENTATION DU PROBLÈME

- Première solution : chaque case échiquier(i,j) prend la valeur *vide* ou *reine*
 - grande combinatoire dans la résolution du problème
- Deuxième solution :
 - $\text{ligne}(i) = j$ si la $i^{\text{ème}}$ ligne a une reine en colonne j
 - $\text{ligne}(i) = 0$ si cette ligne est vide
 - intègre déjà une partie des contraintes, puisqu'on ne peut pas avoir deux reines sur la même ligne

RÉSOUUDRE UN PROBLÈME DE SATISFACTION DE CONTRAINTES (1)

- Résoudre = affecter une valeur à toutes les variables sans violer de contraintes
- Première méthode : « générer et tester »
 - On construit une solution puis on vérifie que les contraintes sont satisfaites
 - Dans l'exemple : placer une reine sur chaque ligne puis vérifier qu'elles ne sont pas en prise

GÉNÉRER ET TESTER SUR LES 4 REINES

- $A = \{(x_1, 1), (x_2, 1), (x_3, 1), (x_4, 1)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 1), (x_3, 1), (x_4, 2)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 1), (x_3, 1), (x_4, 3)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 1), (x_3, 1), (x_4, 4)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 1), (x_3, 2), (x_4, 1)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 1), (x_3, 2), (x_4, 2)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 1), (x_3, 2), (x_4, 3)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 1), (x_3, 2), (x_4, 4)\} \Rightarrow$ inconsistante
-

RÉSOLUTION D'UN PROBLÈME DE SATISFACTION DE CONTRAINTES (2)

- Deuxième méthode : ne pas développer une solution partielle qui viole déjà les contraintes
 - Dans l'exemple : à chaque fois qu'on place une reine, on vérifie qu'elle n'est en prise avec aucune autre, sinon on revient sur les derniers choix

RETOUR ARRIÈRE SUR LES 4 REINES

- $A = \{(x_1, 1), (x_2, ?), (x_3, ?), (x_4, ?)\} \Rightarrow$ consistante
- $A = \{(x_1, 1), (x_2, 1), (x_3, ?), (x_4, ?)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 2), (x_3, ?), (x_4, ?)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 3), (x_3, ?), (x_4, ?)\} \Rightarrow$ consistante
- $A = \{(x_1, 1), (x_2, 3), (x_3, 1), (x_4, ?)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 3), (x_3, 2), (x_4, ?)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 3), (x_3, 3), (x_4, ?)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 3), (x_3, 4), (x_4, ?)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 4), (x_3, ?), (x_4, ?)\} \Rightarrow$ consistante
- $A = \{(x_1, 1), (x_2, 4), (x_3, 1), (x_4, ?)\} \Rightarrow$ inconsistante
- $A = \{(x_1, 1), (x_2, 3), (x_3, 2), (x_4, ?)\} \Rightarrow$ consistante
-

RÉSoudre UN PROBLÈME DE SATISFACTION DE CONTRAINTES (3)

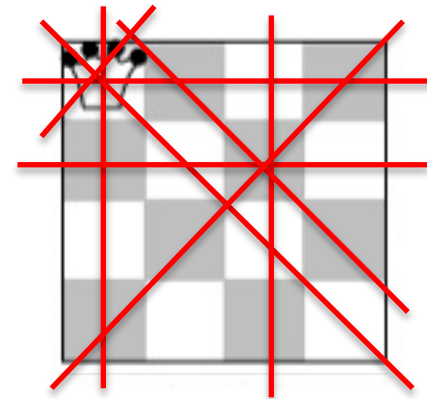
- Troisième méthode : le filtrage
 - Réduire le domaine des variables à chaque affectation
 - Dans l'exemple : à chaque fois qu'on place une reine, on élimine du domaine des autres reines toutes les positions qui sont maintenant attaquées

FILTRAGE SUR LES 4 REINES

- $A = \{(x_1, 1), (x_2, ?), (x_3, ?), (x_4, ?)\}$
- $D(x_1) = \{1\}, D(x_2) = \{3, 4\}, D(x_3) = \{2, 4\}, D(x_4) = \{2, 3\}$

- $A = \{(x_1, 1), (x_2, 3), (x_3, ?), (x_4, ?)\}$
- $D(x_1) = \{1\}, D(x_2) = \{3\}, D(x_3) = \{\}, D(x_4) = \{2\}$

- ... retour arrière et on re-filtre

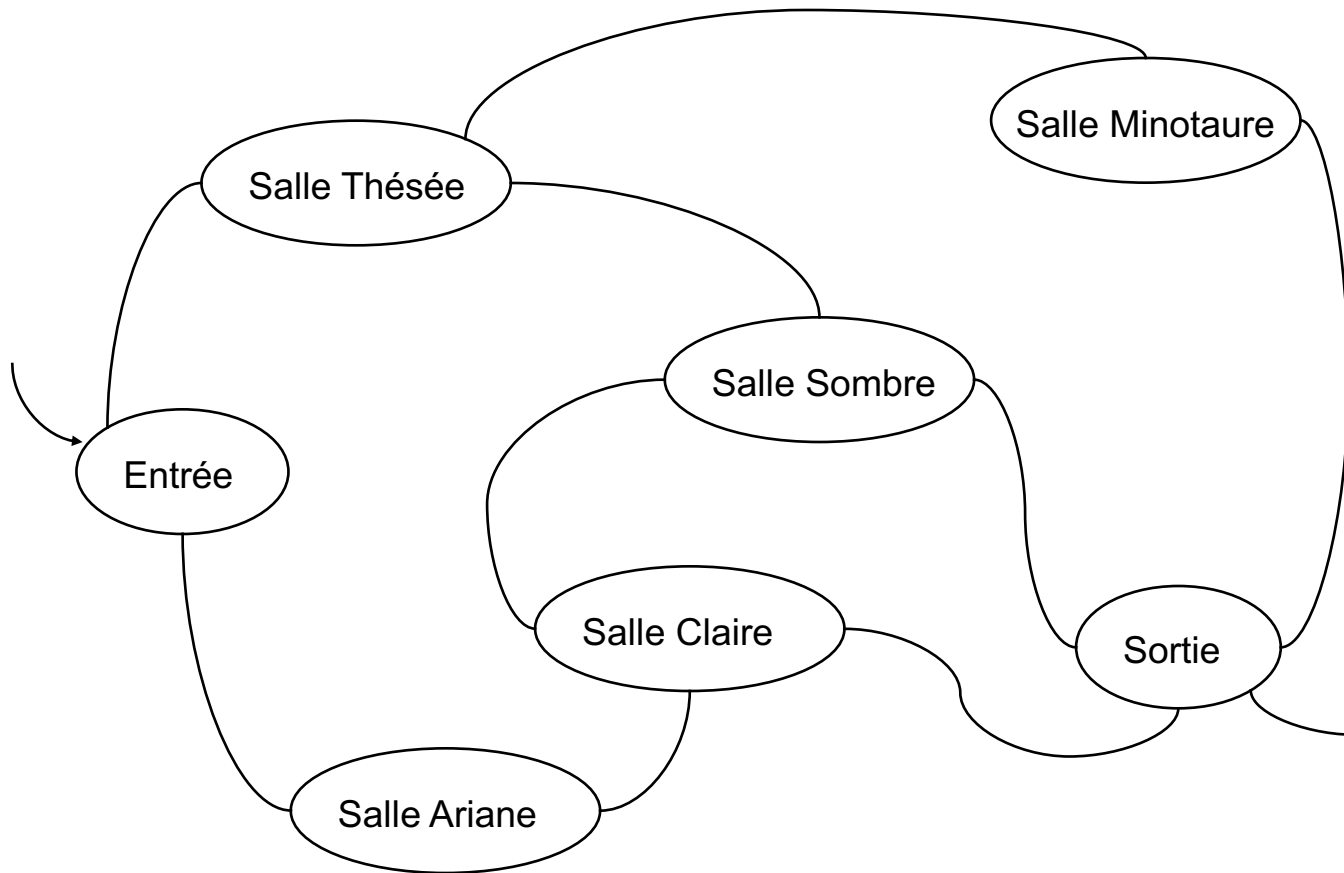


- $A = \{(x_1, 2), (x_2, ?), (x_3, ?), (x_4, ?)\}$
- $D(x_1) = \{2\}, D(x_2) = \{4\}, D(x_3) = \{1, 3\}, D(x_4) = \{1, 3, 4\}$
- ...

RÉSoudre UN PROBLÈME DE SATISFACTION DE CONTRAINTES (4)

- Quatrième méthode : utiliser une heuristique
 - par exemple commencer par affecter la variable dont le domaine est le plus petit (pas réalisable dans notre exemple)

UN PROBLÈME DE LABYRINTHE



GÉNÉRALISATION : RECHERCHE DANS UN GRAPHE D'ÉTAT

- On définit pour chaque problème :
 - Un état initial
 - Un état final (ou plusieurs)
 - Des états interdits (éventuellement)
 - Des opérateurs de transition
- On définit un algorithme général de recherche dans le graphe ainsi construit

ALGORITHME DE RECHERCHE

Liste des états
déjà rencontrés

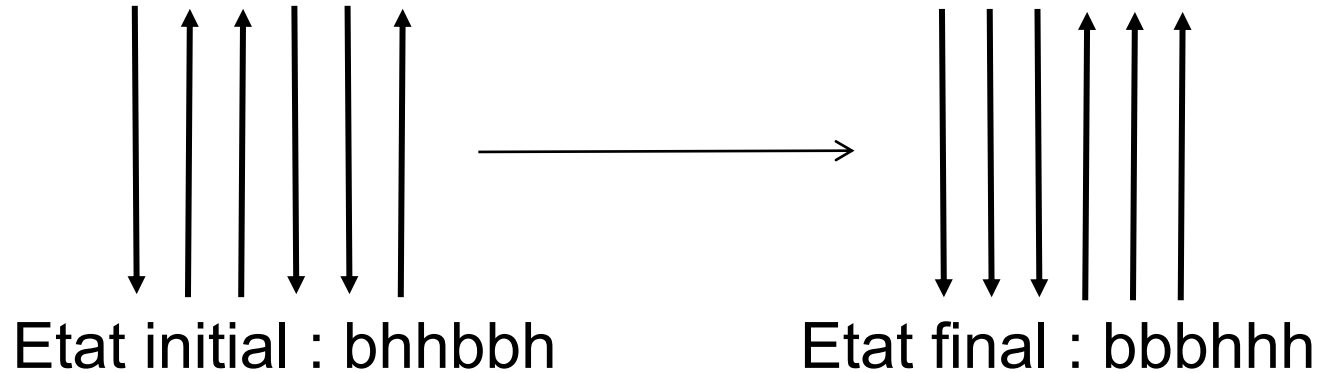
État initial → État courant

Opérateur

État suivant → État final

Liste des
opérateurs à
appliquer

UN PROBLÈME ANALOGUE



Quatre opérateurs de transition pour retourner deux flèches adjacentes :

R1 : hh → bb

R2 : hb → bh

R3 : bh → hb

R4 : bb → hh