Intelligence Artificielle
Logique (1/3)

Bruno Bouzy

http://web.mi.parisdescartes.fr/~bouzy
bruno.bouzy@parisdescartes.fr

Licence 3 Informatique
UFR Mathématiques et Informatique
Université Paris Descartes

Motivations

- Agents fondés sur les connaissances
  - Représentation des connaissances
  - Processus de raisonnement

⇒ Tirer parti de connaissances grâce à une capacité à combiner et recombiner des informations pour les adapter à une multitude de fins.

→ Mathématicien démontre un théorème
→ Astronome calcule la durée de vie de la Terre

⇒ Environnements *partiellement observables* : combiner connaissances générales et percepts reçus pour inférer des aspects cachés de l'état courant.

→ Médecin ausculte un patient
→ Compréhension du langage naturel :

- "John a vu le diamant à travers le carreau et l'a convoité"
- "John a lancé un caillou à travers le carreau et l'a cassé"
- Connaissances de **sens commun**

Motivations

- Agents fondés sur les connaissances
  - Représentation des connaissances
  - Processus de raisonnement

$\Rightarrow$ Tirer parti de connaissances grâce à une capacité à combiner et recombiner des informations pour les adapter à une multitude de fins.

- $\rightarrow$ Mathématicien démontre un théorème
- $\rightarrow$ Astronome calcule la durée de vie de la Terre

$\Rightarrow$ Environnements *partiellement observables* : combiner connaissances générales et percepts reçus pour inférer des aspects cachés de l'état courant.

- $\rightarrow$ Médecin ausculte un patient
- $\rightarrow$ Compréhension du langage naturel :
  - "John a vu le diamant à travers le carreau et l'a convoité"
  - "John a lancé un caillou à travers le carreau et l'a cassé"
  - Connaissances de **sens commun**

Motivations

- Agents fondés sur les connaissances
  - Représentation des connaissances
  - Processus de raisonnement

$\Rightarrow$ Tirer parti de connaissances grâce à une capacité à combiner et recombiner des informations pour les adapter à une multitude de fins.

- $\rightarrow$ Mathématicien démontre un théorème
- $\rightarrow$ Astronome calcule la durée de vie de la Terre

$\Rightarrow$ Environnements *partiellement observables* : combiner connaissances générales et percepts reçus pour inférer des aspects cachés de l'état courant.

- $\rightarrow$ Médecin ausculte un patient
- $\rightarrow$ Compréhension du langage naturel :
  - "John a vu le diamant à travers le carreau et l'a convoité"
  - "John a lancé un caillou à travers le carreau et l'a cassé"
  - Connaissances de **sens commun**

Agents logiques

- Agents fondés sur les connaissances
- Le monde du Wumpus
- Principe généraux de la logique
- Logique propositionnelle
- Schémas de raisonnement en logique propositionnelle
- Agents basés sur la logique propositionnelle
- Conclusion

Agents logiques

Base de connaissances (BC)

- Base de connaissances : ensemble d'énoncés exprimés dans un langage formel
- Les agents logiques peuvent être vus au :
  - niveau des connaissances : ce qu'ils savent, quelle que soit l'implémentation
  - niveau des implémentations : structures de données dans la BC, et les algorithmes qui les manipulent
- Approche **déclarative** pour construire la base de connaissances
  - `Tell` : ce qu'ils doivent savoir
  - `Ask` : demander ce qu'ils doivent faire. La réponse doit **résulter** de BC

Agent basé sur les connaissances

Un agent basé sur les connaissances doit être capable de :

- Représenter les états, les actions
- Incorporer de nouvelles perceptions
- Mettre à jour sa représentation interne du monde
- Déduire les propriétés cachées du monde
- Déduire les actions appropriées

Exemple simple d'un agent basé sur les connaissances

Programme agent basé sur les connaissances

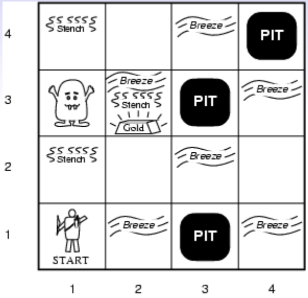
fonction KB-Agent(*percept*) **retourne** *action*
  variables statiques : KB, base de connaissances
    t, compteur initialisé à 0, indique le temps

  Tell(KB, Make-percept-sentence(*percept*, t))
  action $\leftarrow$ Ask(KB, Make-action-query(t))
  Tell(KB, Make-action-sentence(*action*, t))
  $t \leftarrow t + 1$
  retourner *action*

Agents logiques

Le monde du Wumpus

- Environnement
  - Agent commence en case [1,1]
  - Cases adjacentes au Wumpus sentent mauvais
  - Brise dans les cases adjacentes aux puits
  - Lueur dans la cases contenant de l'or
  - Tirer tue le Wumpus s'il est en face
  - On ne peut tirer qu'une fois
  - S'il est tué, le Wumpus crie
  - Choc si l'agent se heurte à un mur
  - Saisir l'or si même case que l'agent
- Capteurs : odeur, brise, lueur, choc, cri
- Percepts : liste de 5 symboles
  Ex : [odeur, brise, rien, rien, rien]
- Actions : tourne gauche, tourne droite, avance, attrappe, tire



- Mesures de performance :
  - or : +1000;
  - mort : -1000;
  - action : -1;
  - utiliser la flèche : -10

Caractérisation du monde du Wumpus

- **Totalement observable**
- **Déterministe**
- **Episodique**
- **Statique**
- **Discret**
- **Mono-agent**

Caractérisation du monde du Wumpus

- **Totalement observable** Non. Perception locale uniquement
- **Déterministe**
- **Episodique**
- **Statique**
- **Discret**
- **Mono-agent**

Caractérisation du monde du Wumpus

- **Totalement observable** Non. Perception locale uniquement
- **Déterministe** Oui
- **Episodique**
- **Statique**
- **Discret**
- **Mono-agent**

Caractérisation du monde du Wumpus

- **Totalement observable** Non. Perception locale uniquement
- **Déterministe** Oui
- **Episodique** Non. Séquentiel au niveau des actions
- **Statique**
- **Discret**
- **Mono-agent**

Caractérisation du monde du Wumpus

- **Totalement observable** Non. Perception locale uniquement
- **Déterministe** Oui
- **Episodique** Non. Séquentiel au niveau des actions
- **Statique** Oui. Le Wumpus et les puits ne bougent pas
- **Discret**
- **Mono-agent**

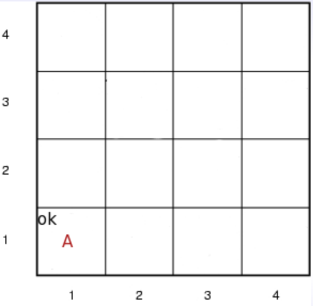
Caractérisation du monde du Wumpus

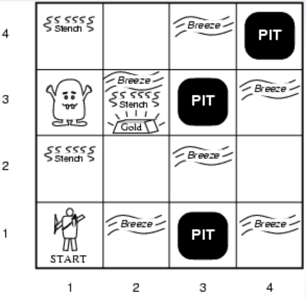
- **Totalement observable** Non. Perception locale uniquement
- **Déterministe** Oui
- **Episodique** Non. Séquentiel au niveau des actions
- **Statique** Oui. Le Wumpus et les puits ne bougent pas
- **Discret** Oui
- **Mono-agent**

Caractérisation du monde du Wumpus

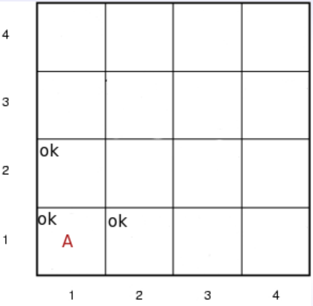
- **Totalement observable** Non. Perception locale uniquement
- **Déterministe** Oui
- **Episodique** Non. Séquentiel au niveau des actions
- **Statique** Oui. Le Wumpus et les puits ne bougent pas
- **Discret** Oui
- **Mono-agent** Oui. Le Wumpus est une caractéristique de la nature

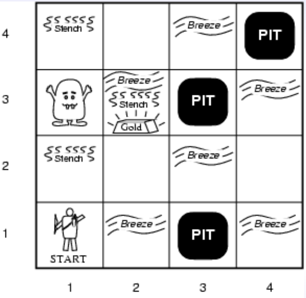
Explorer un monde du Wumpus



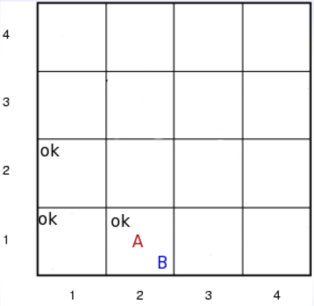


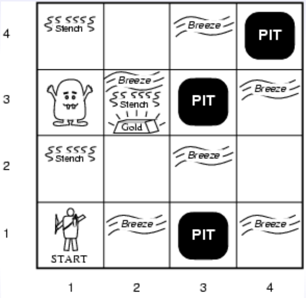
Explorer un monde du Wumpus



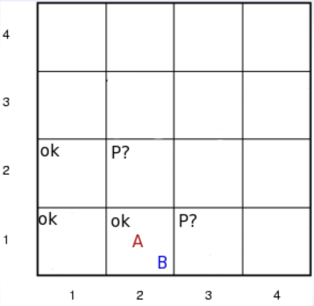


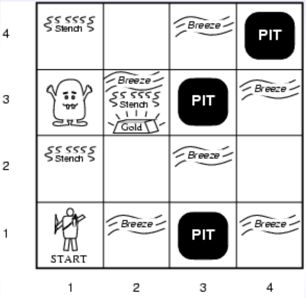
Explorer un monde du Wumpus



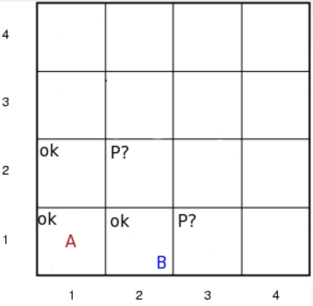


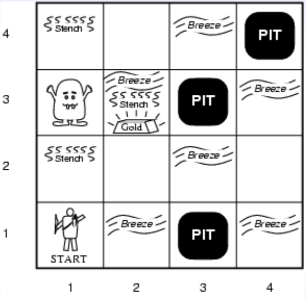
Explorer un monde du Wumpus



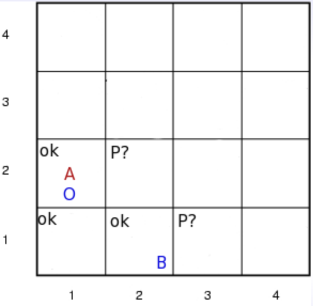


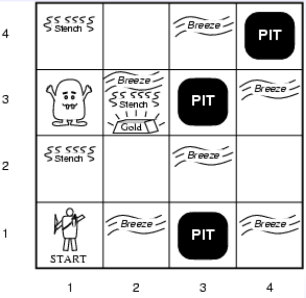
Explorer un monde du Wumpus



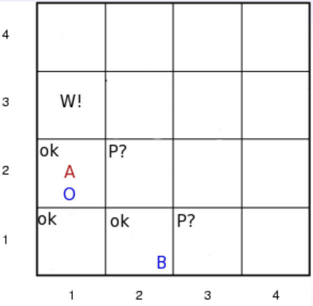


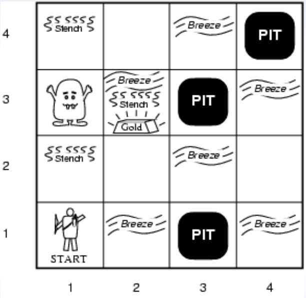
Explorer un monde du Wumpus



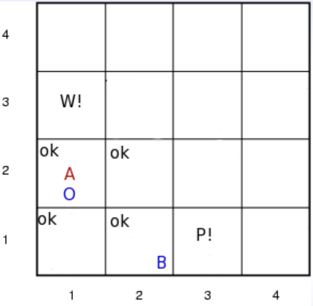


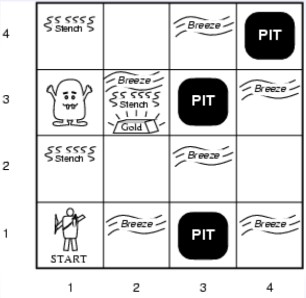
Explorer un monde du Wumpus



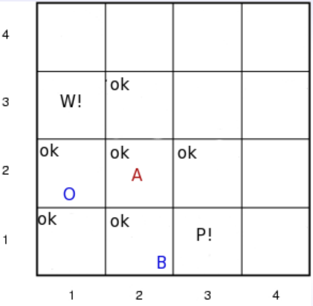


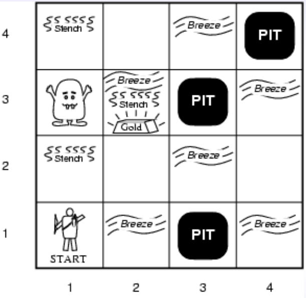
Explorer un monde du Wumpus



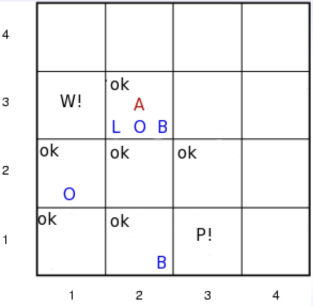


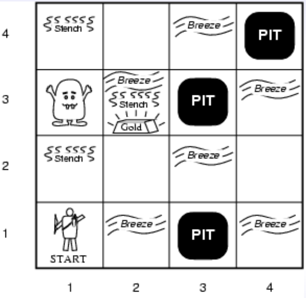
Explorer un monde du Wumpus





Explorer un monde du Wumpus





Agents logiques

Principe généraux de la logique

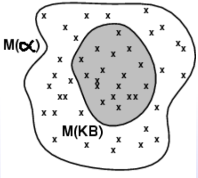
- **Logique** : langage formel permettant de représenter des informations à partir desquelles on peu tirer des conclusions
- La **syntaxe** désigne les phrases (ou énoncés) bien formées dans le langage
- La **sémantique** désigne la signification, le sens de ces phrases
- Par exemple, dans le langage arithmétique :
  - $x + y = 4$ est une phrase syntaxiquement correcte
  - $x4y+ =$ n'en est pas une
  - $2 + 3 = 4$ est une phrase syntaxiquement correcte mais sémantiquement incorrecte
  - $x + y = 4$ est vraie ssi x et y sont des nombres, et que leur somme fait 4
  - $x + y = 4$ est vraie dans un monde où $x = 1$ et $y = 3$
  - $x + y = 4$ est fausse dans un monde où $x = 2$ et $y = 1$

Relation de conséquences

- **Relation de conséquences** : un énoncé découle logiquement d'un autre énoncé : $\alpha \models \beta$
- $\alpha \models \beta$ est vraie si et seulement si β est vraie dans tous mondes où α est vraie
    - Si α est vraie, β doit être vraie
    - Par exemple, $(x + y = 4) \models (x + y \leq 4)$
- Bases de connaissances = ensemble d'énoncés. Une BC a un énoncé pour conséquence : $BC \models \alpha$
- La relation de conséquences est une relation entre des énoncés (la syntaxe) basée sur la sémantique

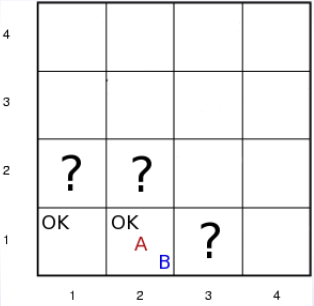
Les modèles

- Les logiciens pensent en terme de modèles, qui sont des mondes structurés dans lesquels la vérité ou la fausseté de chaque énoncé peut être évaluée
- m est un modèle de l'énoncé α si α est vraie dans m
- $M(\alpha)$ est l'ensemble de tous les modèles de α
- $BC \models \alpha$ si et seulement si $M(BC) \subseteq M(\alpha)$

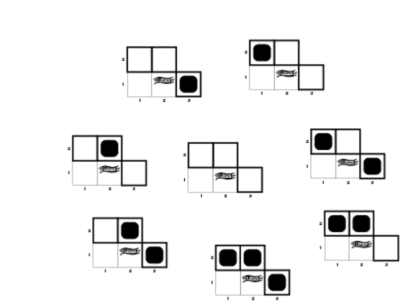


Relation de conséquences dans le monde du Wumpus

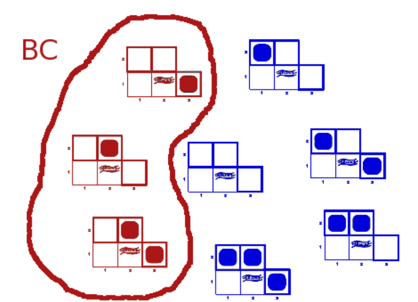
- Situation après avoir effectué
  - Rien en [1,1]
  - Droite
  - Brise en [2,1]
- Considérer les modèles possible pour la base de connaissances en ne considérant que les puits
- $2^3 = 8$ modèles possibles



Modèles du monde du Wumpus

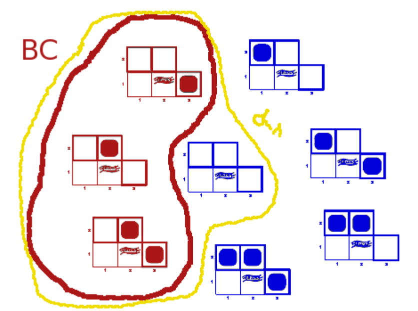


Modèles du monde du Wumpus



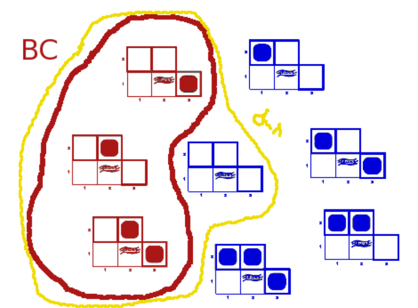
- BC = règles du monde Wumpus + observations

Modèles du monde du Wumpus



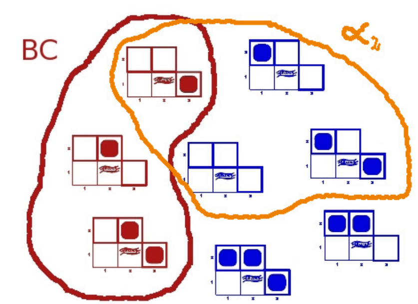
- BC = règles du monde Wumpus + observations
- α_1 = "[1,2] est sans puits"

Modèles du monde du Wumpus



- BC = règles du monde Wumpus + observations
- α_1 = "[1,2] est sans puits"
- $BC \models \alpha_1$, prouvé par vérification des modèles (*model checking*)

Modèles du monde du Wumpus



- BC = règles du monde Wumpus + observations
- α_2 = "[2,2] est sans puits"
- $BC \not\models \alpha_2$

Inférence logique

- $KB \vdash_i \alpha$: l'énoncé α est dérivé de KB par la procédure i
- Validité (soundness) : i est valide si, lorsque $KB \vdash_i \alpha$ est vrai, alors $KB \models \alpha$ est également vrai
- Complétude (completness) : i est complète si, lorsque $KB \models \alpha$ est vrai, alors $KB \vdash_i \alpha$ est également vrai
- Une procédure valide et complète permet de répondre à toute question dont la réponse peut être déduite de la base de connaissances

Agents logiques

- Agents fondés sur les connaissances
- Le monde du Wumpus
- Principe généraux de la logique
- **Logique propositionnelle**
- Schémas de raisonnement en logique propositionnelle
- Agents basés sur la logique propositionnelle
- Conclusion

Logique propositionelle - syntaxe

- Logique propositionnelle = logique très simple
- Un **énoncé** est un énoncé atomique ou un énoncé complexe
- Un **symbole propositionnel** est une proposition qui peut être vraie ou fausse P, Q, R...
- Enoncés atomiques : un seul symbole propositionnel, *vrai* ou *faux*. Appelé aussi un littéral
- Enoncés complexes :
  - Si E est un énoncé, $\neg E$ est un énoncé (négation)
  - Si E_1 et E_2 sont des énoncés, $E_1 \wedge E_2$ est un énoncé (conjonction)
  - Si E_1 et E_2 sont des énoncés, $E_1 \vee E_2$ est un énoncé (disjonction)
  - Si E_1 et E_2 sont des énoncés, $E_1 \Rightarrow E_2$ est un énoncé (implication)
  - Si E_1 et E_2 sont des énoncés, $E_1 \Leftrightarrow E_2$ est un énoncé (équivalence)

Logique propositionelle - sémantique

- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
  - 3 symboles propositionnels $P_{1,1}$, $P_{2,2}$ et $P_{3,1}$
  - $m_1 = \{P_{1,1} = Faux,\ P_{2,2} = Faux,\ P_{3,1} = Vrai\}$
- n symboles propositionnels $= 2^n$ modèles possibles
- Règles pour évaluer un énoncé en fonction d'un modèle m :

$\neg E$	est vrai ssi	E est faux
$E_1 \wedge E_2$	est vrai ssi	E_1 est vrai **et** E_2 est vrai
$E_1 \vee E_2$	est vrai ssi	E_1 est vrai **ou** E_2 est vrai
$E_1 \Rightarrow E_2$	est vrai ssi	E_1 est faux **ou** E_2 est vrai
$E_1 \Rightarrow E_2$	est faux ssi	E_1 est vrai **et** E_2 est faux
$E_1 \Leftrightarrow E_2$	est vrai ssi	$E_1 \Rightarrow E_2$ est vrai **et** $E_2 \Rightarrow E_1$ est vrai

- Un processus récursif simple permet d'évaluer un énoncé. Par exemple :

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = Vrai \wedge (Faux \vee Vrai) = Vrai \wedge Vrai = Vrai$$

Table de vérité des connecteurs logiques

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
vrai	*vrai*	*faux*	*vrai*	*vrai*	*vrai*	*vrai*
vrai	*faux*	*faux*	*faux*	*vrai*	*faux*	*faux*
faux	*vrai*	*vrai*	*faux*	*vrai*	*vrai*	*faux*
faux	*faux*	*vrai*	*faux*	*faux*	*vrai*	*vrai*

Base de connaissances du monde du Wumpus (simplifié)

- $P_{i,j}$ vrai s'il y a un puits en $[i,j]$
- $B_{i,j}$ vrai s'il y a une brise en $[i,j]$
- Base de connaissances :
  - $R_1 : \neg P_{1,1}$
  - Brise ssi puits dans une case adjacente :
    $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
    $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
  - $R_4 : \neg B_{1,1}$
  - $R_5 : B_{2,1}$
- BC : $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$

Base de connaissances du monde du Wumpus (simplifié)

- 7 symboles propositionnels : $2^7 = 128$ modèles possibles

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	BC
faux	*faux*	*faux*	*faux*	*faux*	*faux*	*faux*	*vrai*	*vrai*	*vrai*	*vrai*	*faux*	*faux*
faux	*faux*	*faux*	*faux*	*faux*	*faux*	*vrai*	*vrai*	*vrai*	*faux*	*vrai*	*faux*	*faux*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
faux	*vrai*	*faux*	*faux*	*faux*	*faux*	*faux*	*vrai*	*vrai*	*faux*	*vrai*	*vrai*	*faux*
faux	*vrai*	*faux*	*faux*	*faux*	*faux*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*
faux	*vrai*	*faux*	*faux*	*faux*	*vrai*	*faux*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*
faux	*vrai*	*faux*	*faux*	*faux*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*
faux	*vrai*	*faux*	*faux*	*vrai*	*faux*	*faux*	*vrai*	*faux*	*faux*	*vrai*	*vrai*	*faux*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
vrai	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*	*faux*	*vrai*	*vrai*	*faux*	*vrai*	*faux*

Inférence par énumération

Enumération en profondeur d'abord de tous les modèles

fonction TT-Entails(*KB*, α) **retourne** *vrai* ou *faux*

  variables statiques : *KB*, base de connaissances

  α, requête, énoncé propositionnel

  symboles $\leftarrow$ liste de symboles propositionnels dans *KB* et α

  retourner TT-Check-All(*KB*, α, *symboles*, [])

fonction TT-Check-All(*KB*, α, *symboles*, *modele*) **retourne** *vrai* ou *faux*

  si Empty?(*symboles*) **alors**

    si PL-True?(*KB*, *modele*) **alors retourner** PL-True?(α, *modele*)

    sinon retourner *vrai*

  sinon faire

    P $\leftarrow$ First(*symboles*); *reste* $\leftarrow$ Rest(*symboles*)

    retourner TT-Check-All(*KB*, α, *reste*, Extend(*P*, *vrai*, *modele*))

      et TT-Check-All(*KB*, α, *reste*, Extend(*P*, *faux*, *modele*))

Inférence par énumération

- Algorithme **valide** et **complet**
- Pour n symboles :
  - complexité temporelle en $O(2^n)$
  - complexité spatiale en $O(n)$

Equivalence logique

- Deux énoncés sont logiquement équivalents si et seulement s'ils sont vrais dans les même modèles :

$$\alpha \equiv \beta \Leftrightarrow \alpha \models \beta \text{ et } \beta \models \alpha$$

$$
\begin{array}{rcl}
(\alpha \wedge \beta) & \equiv & (\beta \wedge \alpha) \text{ commutativité de } \wedge \\
(\alpha \vee \beta) & \equiv & (\beta \vee \alpha) \text{ commutativité de } \vee \\
((\alpha \wedge \beta) \wedge \gamma) & \equiv & (\alpha \wedge (\beta \wedge \gamma)) \text{ associativité de } \wedge \\
((\alpha \vee \beta) \vee \gamma) & \equiv & (\alpha \vee (\beta \vee \gamma)) \text{ associativité de } \vee \\
\neg(\neg\alpha) & \equiv & \alpha \text{ élimination de la double négation} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\beta \Rightarrow \neg\alpha) \text{ contraposition} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\alpha \vee \beta) \text{ élimination de l'implication} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \text{ élimination de l'équivalence} \\
\neg(\alpha \wedge \beta) & \equiv & (\neg\alpha \vee \neg\beta) \text{ De Morgan} \\
\neg(\alpha \vee \beta) & \equiv & (\neg\alpha \wedge \neg\beta) \text{ De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) & \equiv & ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \text{ distributivité de } \wedge \text{ par rapport à } \vee \\
(\alpha \vee (\beta \wedge \gamma)) & \equiv & ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \text{ distributivité de } \vee \text{ par rapport à } \wedge
\end{array}
$$

Validité et satisfiabilité

- Un énoncé est valide s'il est vrai dans tous les modèles. On dit aussi tautologie
  - Exemples : *vrai*; $A \vee \neg A$; $A \Rightarrow A$; $(A \wedge (A \Rightarrow B)) \Rightarrow B$
- Théorème de la déduction :

$$KB \models \alpha \text{ si et seulement si } (KB \Rightarrow \alpha) \text{ est valide}$$

- Un énoncé est satisfiable s'il est vrai dans certains modèles
  - Exemples : $A \vee B$; C
- Un énoncé est insatisfiable s'il n'est vrai dans aucun modèle
  - Exemple : $A \wedge \neg A$
- Satisfiabilité :

$$KB \models \alpha \text{ si et seulement si } (KB \wedge \neg\alpha) \text{ est insatisfiable}$$

Validité et satisfiabilité

- Un énoncé est valide s'il est vrai dans tous les modèles. On dit aussi tautologie
  - Exemples : *vrai*; $A \vee \neg A$; $A \Rightarrow A$; $(A \wedge (A \Rightarrow B)) \Rightarrow B$
- Théorème de la déduction :

  $KB \models \alpha$ si et seulement si $(KB \Rightarrow \alpha)$ est valide
- Un énoncé est satisfiable s'il est vrai dans certains modèles
  - Exemples : $A \vee B$; C
- Un énoncé est insatisfiable s'il n'est vrai dans aucun modèle
  - Exemple : $A \wedge \neg A$
- Satisfiabilité :

  $KB \models \alpha$ si et seulement si $(KB \wedge \neg\alpha)$ est insatisfiable

Agents logiques

Méthodes de preuve

Les méthodes de preuves sont de deux principaux types :

- **Application des règles d'inférence**
  - Génération légitime (valide) de nouveaux énoncés à partir de ceux que l'on a déjà
  - **Preuve** : séquence d'applications des règles d'inférence
  - Nécessite la transformation des énoncés en forme normale
- **Vérification des modèles** (Model checking)
  - Enumération de la table de vérité (toujours exponentiel en n)
  - Amélioré par backtracking (Davis-Putnam-Logemann-Loveland (DPLL))
  - Recherche heuristique dans l'espace d'état (valide mais incomplet)

Schémas de raisonnement en logique propositionnelle

- Modus Ponens :

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

- Elimination de la conjonction :

$$\frac{\alpha \land \beta}{\alpha}$$

- Elimination de l'équivalence :

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)}$$

$$\frac{(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

Exemple

- $R_1 : \neg P_{1,1}$; $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$; $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$; $R_4 : \neg B_{1,1}$; $R_5 : B_{2,1}$
- On veut prouver $\neg P_{1,2}$ (pas de puits en $[1, 2]$)
- **Elimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- **Elimination de la conjonction** à R_6 :
  $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- **Equivalence logique des contraposées** :
  $R_8 : \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$
- **Modus Ponens** avec R_8 et R_4 :
  $R_9 : \neg(P_{1,2} \vee P_{2,1})$
- **Règle de De Morgan** :
  $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

Exemple

- $R_1 : \neg P_{1,1}$; $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$; $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$; $R_4 : \neg B_{1,1}$; $R_5 : B_{2,1}$
- On veut prouver $\neg P_{1,2}$ (pas de puits en $[1, 2]$)
- **Elimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- **Elimination de la conjonction** à R_6 :
  $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- **Equivalence logique des contraposées** :
  $R_8 : \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$
- **Modus Ponens** avec R_8 et R_4 :
  $R_9 : \neg(P_{1,2} \vee P_{2,1})$
- **Règle de De Morgan** :
  $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

Exemple

- $R_1 : \neg P_{1,1}$; $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$; $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$; $R_4 : \neg B_{1,1}$; $R_5 : B_{2,1}$
- On veut prouver $\neg P_{1,2}$ (pas de puits en $[1,2]$)
- **Elimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- **Elimination de la conjonction** à R_6 :
  $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- **Equivalence logique des contraposées** :
  $R_8 : \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$
- **Modus Ponens** avec R_8 et R_4 :
  $R_9 : \neg(P_{1,2} \vee P_{2,1})$
- **Règle de De Morgan** :
  $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

Exemple

- $R_1 : \neg P_{1,1}$; $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$; $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$; $R_4 : \neg B_{1,1}$; $R_5 : B_{2,1}$
- On veut prouver $\neg P_{1,2}$ (pas de puits en $[1, 2]$)
- **Elimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- **Elimination de la conjonction** à R_6 :
  $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- **Equivalence logique des contraposées** :
  $R_8 : \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$
- **Modus Ponens** avec R_8 et R_4 :
  $R_9 : \neg(P_{1,2} \vee P_{2,1})$
- **Règle de De Morgan** :
  $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

Exemple

- $R_1 : \neg P_{1,1}$; $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$; $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$; $R_4 : \neg B_{1,1}$; $R_5 : B_{2,1}$
- On veut prouver $\neg P_{1,2}$ (pas de puits en $[1,2]$)
- **Elimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- **Elimination de la conjonction** à R_6 :
  $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- **Equivalence logique des contraposées** :
  $R_8 : \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$
- **Modus Ponens** avec R_8 et R_4 :
  $R_9 : \neg(P_{1,2} \vee P_{2,1})$
- **Règle de De Morgan** :
  $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

Exemple

- $R_1 : \neg P_{1,1}$; $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$; $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$; $R_4 : \neg B_{1,1}$; $R_5 : B_{2,1}$
- On veut prouver $\neg P_{1,2}$ (pas de puits en $[1,2]$)
- **Elimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- **Elimination de la conjonction** à R_6 :
  $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- **Equivalence logique des contraposées** :
  $R_8 : \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$
- **Modus Ponens** avec R_8 et R_4 :
  $R_9 : \neg(P_{1,2} \vee P_{2,1})$
- **Règle de De Morgan** :
  $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

Résolution : exemple

- Pas de brise en $[1, 2]$; on ajoute les règles suivantes dans la base de connaissance :
  R_{11} : $\neg B_{1,2}$
  R_{12} : $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$
- Même processus que pour obtenir R_{10} :
  R_{13} : $\neg P_{2,2}$
  R_{14} : $\neg P_{1,3}$
- Elimination de la double implication en R_3 + Modus ponens avec R_5 :
  R_{15} : $P_{1,1} \vee P_{2,2} \vee P_{3,1}$
- Règle de résolution : $\neg P_{2,2}$ dans R_{13} se résoud avec $P_{2,2}$ dans R_{15} :
  R_{16} : $P_{1,1} \vee P_{3,1}$
- $\neg P_{1,1}$ dans R_1 se résoud avec $P_{1,1}$ dans R_{16} :
  R_{17} : $P_{3,1}$

Résolution

- Forme normale conjonctive (CNF) : conjonction de disjonctions de littéraux.
  - Une disjonction de littéraux est une clause
  - Exemple : $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
- Résolution unitaire. Chaque l est un littéral, l_i et $\neg l_i$ sont des littéraux complémentaires :

$$\frac{l_1 \vee \ldots \vee l_i \vee \ldots \vee l_k, \quad \neg l_i}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

- Résolution. Chaque l est un littéral, $m_j = \neg l_i$:

$$\frac{l_1 \vee \ldots \vee l_i \vee \ldots \vee l_k, \quad m_1 \vee \ldots \vee m_j \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

- La résolution est valide et complète pour la logique propositionnelle

Transformation d'un énoncé en CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Elimination de $\Leftrightarrow$: remplacement de $\alpha \Leftrightarrow \beta$ par $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

   $$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

2. Elimination de $\Rightarrow$: remplacement de $\alpha \Rightarrow \beta$ par $\neg\alpha \vee \beta$

   $$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$$

3. Déplacer $\neg$ “à l'intérieur” en utilisant les règles de Morgan et la double négation :

   $$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$$

4. Appliquer la loi de distributivité sur $\wedge$ et $\vee$

   $$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

Algorithme de résolution

- Démonstration par l'absurde : pour montrer $KB \models \alpha$, on montre que $KB \wedge \neg\alpha$ n'est pas satisfiable

Algorithme de résolution

fonction PL-Resolution(KB, α) **retourne** *vrai* ou *faux*
    clauses $\leftarrow$ ensemble de clauses dans la représentation CNF de $KB \wedge \neg\alpha$
    nouveau $\leftarrow \{\}$
    loop do
        pour chaque C_i, C_j **dans** *clauses* **faire**
            resolvants $\leftarrow$ PL-Resoud(C_i, C_j)
            si *resolvants* contient la clause vide **alors retourner** *vrai*
            nouveau $\leftarrow$ *nouveau* $\cup$ *resolvants*
        si *nouveau* $\subseteq$ *clauses* **alors retourner** *faux*
        clauses $\leftarrow$ *clause* $\cup$ *nouveau*

Clauses de Horn

- Clauses de Horn : disjonction de littéraux dont **un au maximum est positif**
  - $(\neg L_{1,1} \vee \neg Brise \vee B_{1,1})$ est une clause de Horn
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$ n'est pas une clause de Horn
- Toute clause de Horn peut s'écrire sous la forme d'une implication avec
  - Prémisse = conjonction de littéraux positifs
  - Conclusion = littéral positif unique
  - $(\neg L_{1,1} \vee \neg Brise \vee B_{1,1}) = ((L_{1,1} \wedge Brise) \Rightarrow B_{1,1})$
- Clauses définies : clauses de Horn ayant **exactement** un littéral positif
- Littéral positif = tête; littéraux négatifs = corps de la clause
- Fait = clause sans littéraux négatifs

Formes de Horn

- Forme de Horn : BC = **conjonction** de **clauses de Horn**
- Modus Ponens pour les clauses de Horn :

$$\frac{\alpha_1, \ldots, \alpha_n \qquad (\alpha_1 \wedge \ldots \wedge \alpha_n) \Rightarrow \beta}{\beta}$$

- Ce Modus Ponens peut être utilisé pour le chaînage avant ou chaînage arrière
- Ces algorithmes sont très naturels et sont réalisés en temps linéaire

Chaînage avant

- Idée : appliquer toutes les règles dont les prémisses sont satisfaits dans la base de connaissances
- Ajouter les conclusions de ces règles dans la base de connaissances, jusqu'à ce que la requête soit satisfaite
- Le chaînage avant est valide et complet pour les bases de connaissances de Horn

Chaînage avant

Chaînage avant

fonction PL-FC-Entails(KB, q) **retourne** *vrai* ou *faux*

variables locales :

compteur table indexée par clause, initialement le nombre de prémisses

infer table, indexée par symbole, chaque entrée initialement à *faux*

agenda liste de symboles, initialement symboles vrais dans KB

tant que *agenda* n'est pas vide **faire**

$p \leftarrow$ Pop(*agenda*)

si $p = q$ **alors retourner** *vrai*

si non $infer[p]$ **alors faire**

$infer[p] \leftarrow vrai$

pour chaque clause de Horn c dans laquelle la prémisse p apparait **faire**

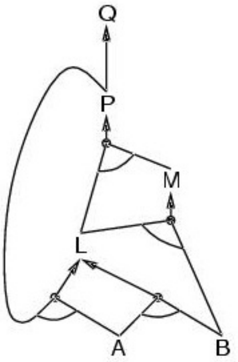
$compteur[c] \leftarrow compteur[c] - 1$

si $compteur[c] = 0$ **alors faire** Push(Head[c], *agenda*)

retourner *faux*

Chaînage avant : exemple

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Preuve de complétude

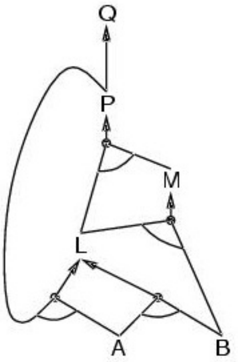
- La procédure de chaînage avant permet d'obtenir tout énoncé atomique pouvant être déduit de KB
    1. L'algorithme atteint un **point fixe** au terme duquel aucune nouvelle inférence n'est possible
    2. L'état final peut être vu comme un **modèle** m dans lequel tout symbole inféré est mis à *vrai*, tous les autres à *faux*
    3. Toutes les clauses définies dans la KB d'origine sont vraies dans m
    4. Donc m est un modèle de KB
    5. Si $KB \models q$ est vrai, q est vrai dans tous les modèles de KB, donc dans m

Chaînage arrière

- Idée : Partir de la requête et rebrousser chemin
  - Vérifier si q n'est pas vérifiée dans la BC
  - Chercher dans la BC les implications ayant q pour conclusion, et essayer de prouver leurs prémisses
- Eviter les boucles : vérifier si le nouveau sous-but n'est pas déjà dans la liste des buts à établir
- Eviter de répéter le même travail : vérifier si le nouveau sous-but a déjà été prouvé vrai ou faux

Chaînage arrière

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Chaînage avant vs chaînage arrière

- Chaînage avant : raisonnement piloté par les données
  - Conclusions à partir de percepts entrants
  - Pas toujours de requête spécifique en tête
  - Beaucoup de conséquences déduites, toutes ne sont pas utiles ou nécessaires
- Chaînage arrière : raisonnement piloté par le but
  - Répondre à des questions spécifiques
  - Se limite aux seuls faits pertinents
  - La complexité du chaînage arrière peut être **bien inférieure** à une fonction linéaire à la taille de la base de connaissances

Algorithmes efficaces d'inférence propositionnelle

Deux familles d'algorithmes efficaces pour l'inférence propositionnelle :

- Exploration par **backtracking**
  - Algorithme DPLL (Davis, Putnam, Logemann, Loveland)
- Algorithmes de recherche locale incomplète
  - Algorithme `WalkSAT`

Algorithme DPLL

- Cet algorithme détermine si un énoncé logique en CNF est satisfiable
- Améliorations par rapport à l'énumération de la table de vérité
  - **Elagage**
    - Une clause est vraie si l'un des littéraux est vrai
    - Un énoncé est faux si l'une des clauses est fausse
  - **Heuristique des symboles purs**
    - Un symbole pur est un symbole qui apparait toujours avec le même "signe" dans toutes les clauses
    - $(A \vee \neg B) \wedge (\neg B \vee \neg C) \wedge (C \vee A)$. A et B sont purs, C est impur
    - Instancier les littéraux des symboles purs à *vrai*
  - **Heuristique de la clause unitaire**
    - Clause unitaire : clause qui ne contient qu'un littéral
    - Ce littéral doit être *vrai*

Algorithme DPLL

Algorithme DPLL

fonction DPLL-Satisfiable?(s) **retourne** *vrai* ou *faux*

  entrées : s, un énoncé propositionnel

  $C \leftarrow$ ensemble des clauses dans la représentation CNF de s

  $S \leftarrow$ liste des symboles propositionnels dans s

  retourner DPLL(C, S, [])

fonction DPLL(C, S, *modele*) **retourne** *vrai* ou *faux*

  si toute clause de C est vraie dans *modele* **alors retourner** *vrai*

  si une clause de C est fausse dans *modele* **alors retourner** *faux*

  $P, valeur \leftarrow$ Find-Pure-Symbol(S, C, *modele*)

  si non nul(P) **alors retourner** DPLL(C, $S \setminus P$, Extend(P, *valeur*, *modele*))

  $P, valeur \leftarrow$ Find-Unit-Clause(C, *modele*)

  si non nul(P) **alors retourner** DPLL(C, $S \setminus P$, Extend(P, *valeur*, *modele*))

  $P \leftarrow$ First(S); *reste* $\leftarrow$ Rest(S)

  retourner DPLL(C, *reste*, Extend(P, *vrai*, *modele*))

    ou DPLL(C, *reste*, Extend(P, *faux*, *modele*))

Algorithme `WalkSAT`

- Algorithme de recherche locale incomplète
- Chaque itération : sélection d'une clause non satisfaite et un symbole à "basculer"
- Choix du symbole à basculer :
  - Fonction d'évaluation : heuristique Min-Conflicts qui minimise le nombre de clauses non satisfaites
  - Etape de parcours aléatoire qui sélectionne le symbole au hasard

Algorithme `WalkSAT`

Algorithme `WalkSAT`

fonction WALKSAT(*clauses*, *p*, *max_flips*) **retourne** un modèle satisfiable ou *erreur*

entrées : *clauses*, un ensemble de clauses

p probabilité de choisir le parcours aléatoire

max_flips le nombre de “bascules” autorisées avant de renoncer

modele ← affectation aléatoire de *vrai*/*faux* des symboles dans *clauses*

pour $i = 1$ **à** *max_flips* **faire**

si *modele* satisfait *clauses* **alors retourner** *modele*

clause ← une clause sélectionnée au hasard parmi les *clauses* fausses de *modele*

avec la probabilité *p* basculer dans *modele* la valeur d'un symbole sélectionné au hasard dans *clause*

sinon basculer le symbole dans *clause* qui maximise le nombre de clauses satisfaites

retourner *erreur*

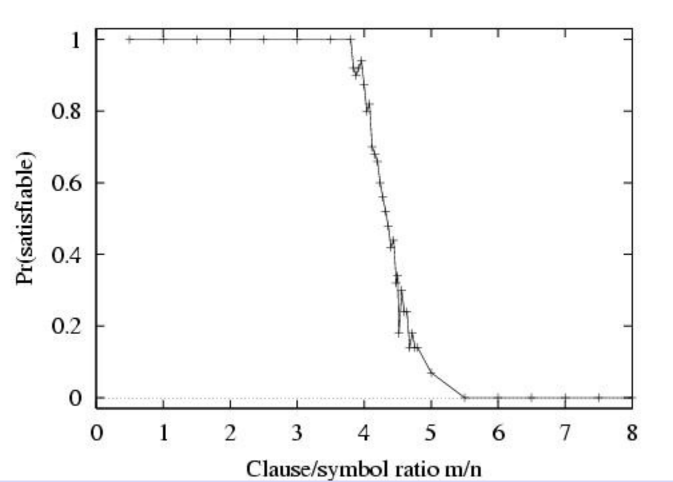
Problèmes de satisfiabilité difficiles

- Soit l'énoncé 3-CNF généré aléatoirement suivant :
  $(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$
- 16 des 32 affectations possibles sont des modèles de cet énoncé

  $\rightarrow$ en moyenne 2 tentatives aléatoires pour trouver un modèle
- Problème difficile : augmenter le nombre de clauses en laissant fixe le nombre de symboles

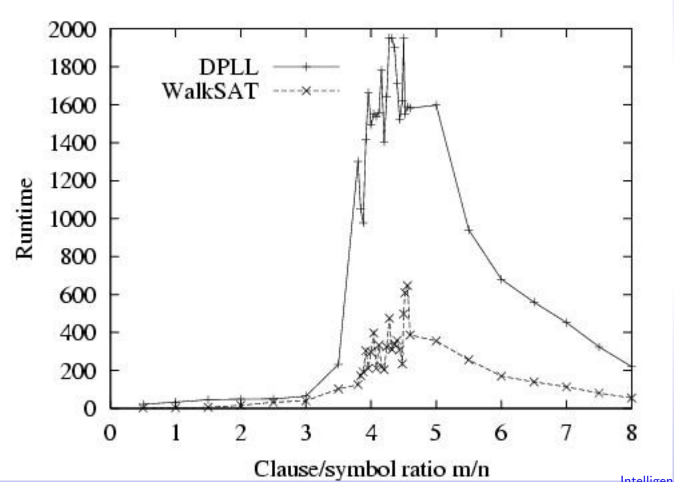
  $\rightarrow$ Problème plus contraint
- m nombre de clauses, n nombre de symboles
- Problèmes difficiles : ratio aux alentours de $\frac{m}{n} = 4.3$: **point critique**

Problèmes de satisfiabilité difficiles



Problèmes de satisfiabilité difficiles

- Temps d'execution médian sur 100 énoncés 3-CNF aléatoires **satisfiables** avec n=50



Agents logiques

Agents basés sur la logique propositionnelle dans le monde du Wumpus

- $\neg P_{1,1}$
- $\neg W_{1,1}$
- $B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y}$
- $S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y}$
- $W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$
- $\neg W_{1,1} \vee \neg W_{1,2}$
- $\neg W_{1,1} \vee \neg W_{1,3}$
- $\ldots$

$\Rightarrow$ 64 symboles propositionnels distincts; 155 énoncés

Agents basés sur la LP dans le monde du Wumpus

fonction PL-Wumpus-Agent(*percept*) **retourne** une action

entrées : *percept* : une liste [*odeur*, *brise*, *lueur*]

var. statiques : KB, contenant au départ la "physique" du monde du Wumpus ; x, y, O la position de l'agent (1, 1, *droite*) au départ ; V un tableau indiquant les cases visitées, initiallement à *faux* ; A action la plus récente de l'agent, initialement à *nul*; P séquence d'actions, initialement vide

si *odeur* **alors** Tell(KB, $S_{x,y}$) **sinon** Tell(KB, $\neg S_{x,y}$)

si *brise* **alors** Tell(KB, $B_{x,y}$) **sinon** Tell(KB, $\neg B_{x,y}$)

si *lueur* **alors** $A \leftarrow$ *ramasser*

sinon si P n'est pas vide **alors** $A \leftarrow$ Pop(P)

sinon si pour une case voisine $[i, j]$, Ask(KB, $(\neg P_{i,j} \wedge \neg W_{i,j})$) est *vrai* **ou**

pour une case voisine $[i, j]$, Ask(KB, $P_{i,j} \vee W_{i,j}$) est *faux*

alors $P \leftarrow$ A*(Route-Problem($[x, y]$, O, $[i, j]$, V)); $A \leftarrow$ Pop(P)

sinon $A \leftarrow$ un déplacement choisi de manière aléatoire

retourner A

Limitation de l'expressivité de la logique propositionnelle

- La base de connaissances doit contenir des énoncés pour représenter "physiquement" toute case
- A chaque temps t et pour chaque localisation $[x, y]$, on a

$$L^t_{x,y} \wedge \ droite^t \ avance^t \ \Rightarrow L^t_{x+1,y}$$

- Prolifération très rapides des clauses

Agents logiques

Conclusion

- Les agents logiques appliquent l'**inférence** sur une **base de connaissances** pour déduire de nouvelles informations et prendre une décision
- Concepts basiques de la logique
    - **Syntaxe** : structure formelle des **énoncés**
    - **Sémantique** : **vérité** de chaque énoncé dans un **modèle**
    - **Conséquence** : vérité nécessaire d'un énoncé par rapport à un autre
    - **Inférence** : dérivation de nouveaux énoncés à partir d'anciens
    - **Validité** : l'inférence ne dérive que des énoncés qui sont des conséquences
    - **Complétude** : l'inférence dérive tous les énoncés qui sont des conséquences
- La résolution est complète pour la logique propositionnelle
- Les chaînages avant et arrière sont linéaire en temps, et complets pour les clauses de Horn
- La logique propositionnelle manque de pouvoir d'expression