Intelligence artificielle

Inférence en logique propositionnelle

Elise Bonzon

elise.bonzon@u-paris.fr

LIPADE - Université de Paris
http://www.math-info.univ-paris5.fr/∽bonzon/

Inférence en logique propositionnelle

1. Schémas de raisonnement en logique propositionnelle
2. Agents basés sur la logique propositionnelle
3. Conclusion

Schémas de raisonnement en logique propositionnelle

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

Calcul de conséquences logiques

Calcul de conséquences

- La déduction est une forme de raisonnement
  - Elle calcule des conséquences logiques d'une BC
  - En utilisant une procédure de démonstration : une preuve

Calcul de conséquences

- La déduction est une forme de raisonnement
  - Elle calcule des conséquences logiques d'une BC
  - En utilisant une procédure de démonstration : une preuve
- Un théorème est une proposition démontrable en appliquant des règles d'inférence

Calcul de conséquences

- La déduction est une forme de raisonnement
  - Elle calcule des conséquences logiques d'une BC
  - En utilisant une procédure de démonstration : une preuve
- Un théorème est une proposition démontrable en appliquant des règles d'inférence
- $BC \vdash_i \alpha$ signifie que α peut être démontré à partir de BC grâce à la procédure i

Calcul de conséquences

- La déduction est une forme de raisonnement
  - Elle calcule des conséquences logiques d'une BC
  - En utilisant une procédure de démonstration : une preuve
- Un théorème est une proposition démontrable en appliquant des règles d'inférence
- $BC \vdash_i \alpha$ signifie que α peut être démontré à partir de BC grâce à la procédure i

Rappel

- Une procédure i est valide (sound) si tout ce qu'elle permet de démontrer à partir de BC est une conséquence logique de BC : si $BC \vdash_i \alpha$, alors $BC \models \alpha$
- Une procédure i est complète si tout ce qui est conséquence logique de BC peut être démontré par i : si $BC \models \alpha$ alors $BC \vdash_i \alpha$

Inférence en logique propositionnelle

Pour pouvoir démontrer de nouvelles conséquences, on a besoin :

- Des règles de ré-écritures (équivalences logiques)
- Et de règles d'inférence
  - Un ensemble de conditions
  - Une partie conclusion (vraie si les conditions sont vérifiées)

Règles d'inférence en logique propositionnelle (1/2)

Schémas pour des étapes élémentaires de démonstration :

Règles d'inférence en logique propositionnelle (1/2)

Schémas pour des étapes élémentaires de démonstration :

Modus Ponens $\dfrac{\alpha \Rightarrow \beta, \alpha}{\beta}$

Règles d'inférence en logique propositionnelle (1/2)

Schémas pour des étapes élémentaires de démonstration :

Modus Ponens $\dfrac{\alpha \Rightarrow \beta, \alpha}{\beta}$

Élimination de la conjonction $\dfrac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$

Règles d'inférence en logique propositionnelle (1/2)

Schémas pour des étapes élémentaires de démonstration :

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

Élimination de la conjonction
$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

Introduction de la conjonction
$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

Règles d'inférence en logique propositionnelle (1/2)

Schémas pour des étapes élémentaires de démonstration :

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

Élimination de la conjonction
$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

Introduction de la conjonction
$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

Introduction de la disjonction
$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

Règles d'inférence en logique propositionnelle (1/2)

Schémas pour des étapes élémentaires de démonstration :

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

Élimination de la conjonction
$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

Introduction de la conjonction
$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

Introduction de la disjonction
$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

Élimination de la double négation
$$\frac{\neg\neg\alpha}{\alpha}$$

Règles d'inférence en logique propositionnelle (2/2)

Élimination de l'équivalence $\dfrac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$

Règles d'inférence en logique propositionnelle (2/2)

Élimination de l'équivalence

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

Introduction de l'équivalence

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

Règles d'inférence en logique propositionnelle (2/2)

Élimination de l'équivalence

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

Introduction de l'équivalence

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

Résolution unitaire

$$\frac{\alpha \vee \beta, \neg\beta}{\alpha}$$

Règles d'inférence en logique propositionnelle (2/2)

Élimination de l'équivalence

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

Introduction de l'équivalence

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

Résolution unitaire

$$\frac{\alpha \vee \beta, \neg\beta}{\alpha}$$

Résolution

$$\frac{\alpha \vee \beta, \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Méthodes de preuve : déduction

Déductions

Une formule A se déduit d'un ensemble de formules $\{B_1, B_2, \ldots, B_n\}$, noté $B_1, B_2, \ldots, B_n \vdash A$ s'il existe une suite finie $(A_1, A_2, \ldots, A_i, \ldots, A)$, où chaque A_i est

- Soit l'un des B_i
- Soit obtenu par l'application d'une règle d'inférence sur deux éléments A_j, A_k de la suite déjà obtenue $(j, k < i)$.

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 $[1, 2]$)

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 $[1, 2]$)
- **Élimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

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 $[1, 2]$)
- **Élimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- **Élimination de la conjonction** à R_6 :
  $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$

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 $[1, 2]$)
- **Élimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- **Élimination de la conjonction** à R_6 :
  $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- **Équivalence logique des contraposées** :
  $R_8 : \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$

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 $[1, 2]$)
- **Élimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- **Élimination de la conjonction** à R_6 :
  $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- **Équivalence 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 :
  $R_9 : \neg(P_{1,2} \vee P_{2,1})$

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 $[1, 2]$)
- **Élimination de l'équivalence** à R_2 :
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- **Élimination de la conjonction** à R_6 :
  $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- **Équivalence 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** :
  $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

Schémas de raisonnement en logique propositionnelle

Preuves par résolution

Preuve par résolution : démarche

1. Normaliser la représentation
   → les formes normales : clauses
2. Introduction d'une règle d'inférence unique
   → la résolution

Standardisation de la représentation

- Il existe plusieurs manières d'exprimer les mêmes propositions

$$p \Rightarrow q \equiv \neg p \vee q \equiv \neg(p \wedge \neg q)$$

$\Rightarrow$ Besoin d'avoir une forme standardisée ou canonique

Standardisation de la représentation

- Il existe plusieurs manières d'exprimer les mêmes propositions

$$p \Rightarrow q \equiv \neg p \vee q \equiv \neg(p \wedge \neg q)$$

$\Rightarrow$ Besoin d'avoir une forme standardisée ou canonique

Forme normale conjonctive (CNF)

Forme normale conjonctive (CNF) : conjonction de disjonctions de littéraux.

- Une disjonction de littéraux est une clause
- Exemple : $(a \vee \neg b) \wedge (b \vee \neg c \vee \neg d)$

Standardisation de la représentation

- Il existe plusieurs manières d'exprimer les mêmes propositions

$$p \Rightarrow q \equiv \neg p \vee q \equiv \neg(p \wedge \neg q)$$

$\Rightarrow$ Besoin d'avoir une forme standardisée ou canonique

Forme normale conjonctive (CNF)

Forme normale conjonctive (CNF) : conjonction de disjonctions de littéraux.

- Une disjonction de littéraux est une clause
- Exemple : $(a \vee \neg b) \wedge (b \vee \neg c \vee \neg d)$

La transformation d'une wff en CNF est toujours possible

Traduction d'une wff en CNF

Traduction d'une wff en CNF

Jusqu'à 5 étapes nécessaires :

1. Éliminer les équivalences
2. Éliminer les implications
3. Faire migrer les négations "à l'intérieur"
4. Éliminer les doubles négations
5. Appliquer la loi de distributivité sur $\wedge$ et $\vee$

Traduction d'une wff en CNF : un exemple

$$(\neg p \wedge (\neg q \Rightarrow r)) \Rightarrow s$$

Traduction d'une wff en CNF : un exemple

$$(\neg p \wedge (\neg q \Rightarrow r)) \Rightarrow s$$

$$\equiv \neg(\neg p \wedge (\neg q \Rightarrow r)) \vee s$$

2. Éliminer les implications

Traduction d'une wff en CNF : un exemple

$$(\neg p \wedge (\neg q \Rightarrow r)) \Rightarrow s$$

$\equiv \neg(\neg p \wedge (\neg q \Rightarrow r)) \vee s$ 2. Éliminer les implications

$\equiv \neg(\neg p \wedge (q \vee r)) \vee s$ 2. Éliminer les implications

Traduction d'une wff en CNF : un exemple

$$
\begin{aligned}
& (\neg p \wedge (\neg q \Rightarrow r)) \Rightarrow s \\
\equiv & \neg(\neg p \wedge (\neg q \Rightarrow r)) \vee s && \text{2. Éliminer les implications} \\
\equiv & \neg(\neg p \wedge (q \vee r)) \vee s && \text{2. Éliminer les implications} \\
\equiv & (\neg\neg p \vee \neg(q \vee r)) \vee s && \text{3. Migrer les négations}
\end{aligned}
$$

Traduction d'une wff en CNF : un exemple

$$
\begin{aligned}
& (\neg p \wedge (\neg q \Rightarrow r)) \Rightarrow s \\
\equiv & \neg(\neg p \wedge (\neg q \Rightarrow r)) \vee s && \text{2. Éliminer les implications} \\
\equiv & \neg(\neg p \wedge (q \vee r)) \vee s && \text{2. Éliminer les implications} \\
\equiv & (\neg\neg p \vee \neg(q \vee r)) \vee s && \text{3. Migrer les négations} \\
\equiv & (\neg\neg p \vee (\neg q \wedge \neg r)) \vee s && \text{3. Migrer les négations}
\end{aligned}
$$

Traduction d'une wff en CNF : un exemple

$$
\begin{aligned}
& (\neg p \wedge (\neg q \Rightarrow r)) \Rightarrow s \\
\equiv & \neg(\neg p \wedge (\neg q \Rightarrow r)) \vee s && \text{2. Éliminer les implications} \\
\equiv & \neg(\neg p \wedge (q \vee r)) \vee s && \text{2. Éliminer les implications} \\
\equiv & (\neg\neg p \vee \neg(q \vee r)) \vee s && \text{3. Migrer les négations} \\
\equiv & (\neg\neg p \vee (\neg q \wedge \neg r)) \vee s && \text{3. Migrer les négations} \\
\equiv & (p \vee (\neg q \wedge \neg r)) \vee s && \text{4. Éliminer les doubles négations}
\end{aligned}
$$

Traduction d'une wff en CNF : un exemple

$$
\begin{aligned}
& (\neg p \wedge (\neg q \Rightarrow r)) \Rightarrow s \\
\equiv & \neg(\neg p \wedge (\neg q \Rightarrow r)) \vee s && \text{2. Éliminer les implications} \\
\equiv & \neg(\neg p \wedge (q \vee r)) \vee s && \text{2. Éliminer les implications} \\
\equiv & (\neg\neg p \vee \neg(q \vee r)) \vee s && \text{3. Migrer les négations} \\
\equiv & (\neg\neg p \vee (\neg q \wedge \neg r)) \vee s && \text{3. Migrer les négations} \\
\equiv & (p \vee (\neg q \wedge \neg r)) \vee s && \text{4. Éliminer les doubles négations} \\
\equiv & ((p \vee \neg q) \wedge (p \vee \neg r)) \vee s && \text{5. Distribuer les } \wedge \text{ sur les } \vee
\end{aligned}
$$

Traduction d'une wff en CNF : un exemple

$$
\begin{aligned}
& (\neg p \wedge (\neg q \Rightarrow r)) \Rightarrow s \\
\equiv & \neg(\neg p \wedge (\neg q \Rightarrow r)) \vee s && \text{2. Éliminer les implications} \\
\equiv & \neg(\neg p \wedge (q \vee r)) \vee s && \text{2. Éliminer les implications} \\
\equiv & (\neg\neg p \vee \neg(q \vee r)) \vee s && \text{3. Migrer les négations} \\
\equiv & (\neg\neg p \vee (\neg q \wedge \neg r)) \vee s && \text{3. Migrer les négations} \\
\equiv & (p \vee (\neg q \wedge \neg r)) \vee s && \text{4. Éliminer les doubles négations} \\
\equiv & ((p \vee \neg q) \wedge (p \vee \neg r)) \vee s && \text{5. Distribuer les } \wedge \text{ sur les } \vee \\
\equiv & (p \vee \neg q \vee s) \wedge (p \vee \neg r \vee s) && \text{5. Distribuer les } \wedge \text{ sur les } \vee
\end{aligned}
$$

Inférence par résolution

- Idée :
  - Soient les clauses $(p \vee q)$ et $(\neg q \vee r)$
    - Si q est vrai, alors r est vrai
    - Si q est faux, alors p est vrai
  - On peut donc conclure $(p \vee r)$

Inférence par résolution

- Idée :
  - Soient les clauses $(p \vee q)$ et $(\neg q \vee r)$
    - Si q est vrai, alors r est vrai
    - Si q est faux, alors p est vrai
  - On peut donc conclure $(p \vee r)$

Résolution unitaire $$\frac{\alpha \vee \beta, \neg\beta}{\alpha}$$

Résolution $$\frac{\alpha_1 \vee \alpha_2 \vee \ldots \vee \beta \vee \ldots \vee \alpha_n, \gamma_1 \vee \gamma_2 \vee \ldots \vee \neg\beta \vee \ldots \vee \gamma_p}{\alpha_1 \vee \ldots \vee \alpha_n \vee \gamma_1 \vee \ldots \vee \gamma_p}$$

Inférence par résolution

- Idée :
  - Soient les clauses $(p \vee q)$ et $(\neg q \vee r)$
    - Si q est vrai, alors r est vrai
    - Si q est faux, alors p est vrai
  - On peut donc conclure $(p \vee r)$

Résolution unitaire $$\frac{\alpha \vee \beta, \neg\beta}{\alpha}$$

Résolution $$\frac{\alpha_1 \vee \alpha_2 \vee \ldots \vee \beta \vee \ldots \vee \alpha_n, \gamma_1 \vee \gamma_2 \vee \ldots \vee \neg\beta \vee \ldots \vee \gamma_p}{\alpha_1 \vee \ldots \vee \alpha_n \vee \gamma_1 \vee \ldots \vee \gamma_p}$$

- Le modus ponens est un cas particulier de la résolution

Algorithme de résolution

- Application du théorème de la déduction : pour montrer $BC \models \alpha$, on montre que $BC \wedge \neg\alpha$ est insatisfiable
- Méthodologie :
  - Ajouter la négation de la conclusion désirée à la base de connaissances
  - Obtention de la clause vide par résolution
- La résolution par réfutation est valide et complète pour la logique propositionnelle

Algorithme de résolution

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-Résout(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*

Schémas de raisonnement en logique propositionnelle

Systèmes à base de règles

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 :

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

- 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

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 apparaît **faire**

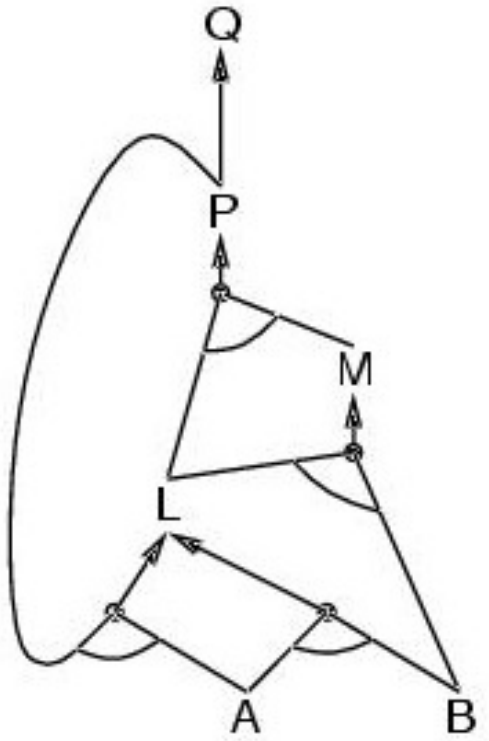
compteur$[c] \leftarrow$ *compteur*$[c] - 1$

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

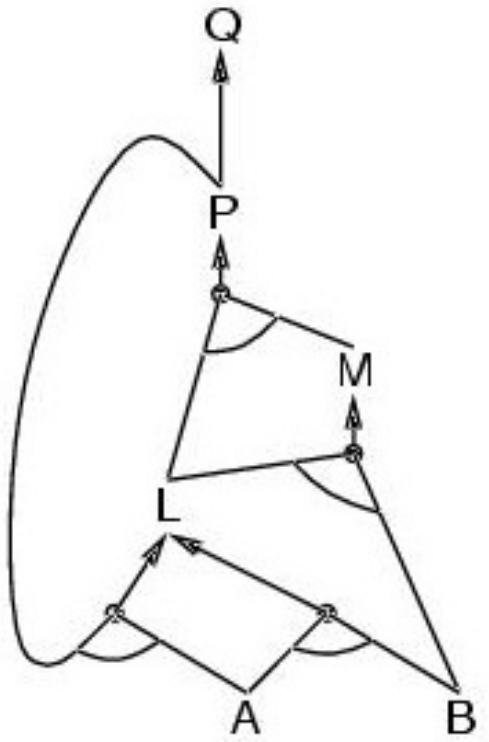
- La procédure de chaînage avant permet d'obtenir tout énoncé atomique pouvant être déduit de KB
  1. L'algorithme atteint un **point fixe** au terme duquel aucune nouvelle inférence n'est possible
  2. L'état final peut être vu comme un **modèle** m dans lequel tout symbole inféré est mis à *vrai*, 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

Schémas de raisonnement en logique propositionnelle

Algorithmes efficaces d'inférence propositionnelle

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

Algorithme DPLL

- Cet algorithme détermine si un énoncé logique en CNF est satisfiable
- Améliorations par rapport à l'énumération de la table de vérité

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**

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

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

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**

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 apparaît toujours avec le même "signe" dans toutes les clauses

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 apparaît 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

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 apparaît 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*

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 apparaît 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**

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 apparaît 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

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 apparaît 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 `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 $\leftarrow$ affectation aléatoire de *vrai*/*faux* des symboles dans *clauses*

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

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

clause $\leftarrow$ 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*

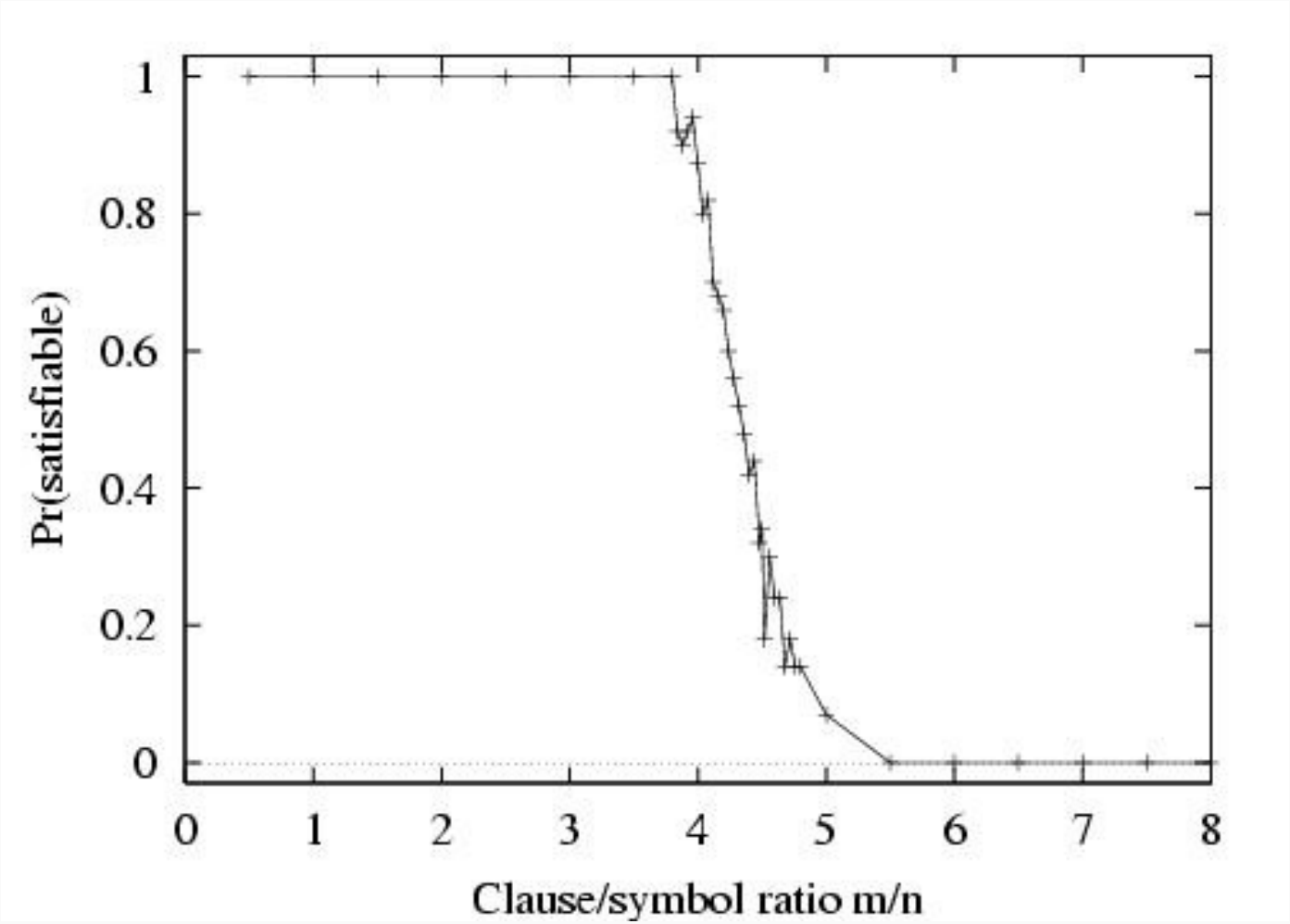
sinon basculer le symbole dans *clause* qui maximise le nombre de clauses satisfaites

retourner *erreur*

Problèmes de satisfiabilité difficiles

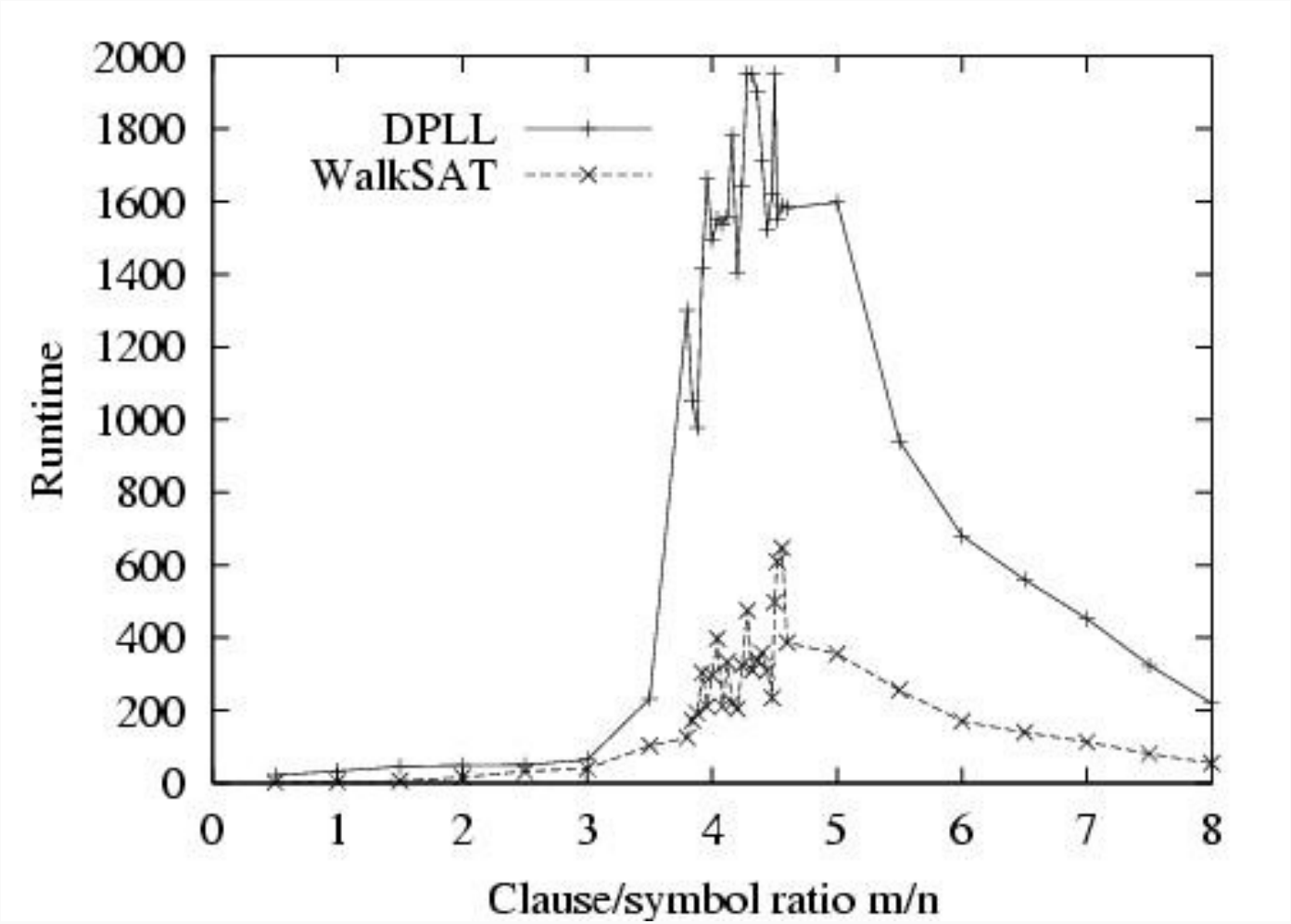
- Soit l'énoncé 3-CNF généré aléatoirement suivant :
  $(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$
- 16 des 32 affectations possibles sont des modèles de cet énoncé
  $\rightarrow$ en moyenne 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'exécution médian sur 100 énoncés 3-CNF aléatoires **satisfiables** avec n=50



Agents basés sur la logique propositionnelle

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, initialement 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

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

- Prolifération très rapide des clauses

Conclusion

Conclusion

- Les agents logiques appliquent l'**inférence** sur une **base de connaissances** pour déduire de nouvelles informations et prendre une décision
- La résolution est valide et complète pour la logique propositionnelle
- Les chaînages avant et arrière sont linéaire en temps, valides, et complets pour les clauses de Horn
- **La logique propositionnelle manque de pouvoir d'expression**