Représentation des connaissances et raisonnement

Logique propositionnelle

Elise Bonzon

elise.bonzon@u-paris.fr

LIPADE - Université Paris Cité
http://helios.mi.parisdescartes.fr/∽bonzon/

Introduction

1. Principe généraux de la logique
2. Logique propositionnelle
3. Schémas de raisonnement en logique propositionnelle
4. Conclusion

Principe généraux de la logique

Principe généraux de la logique

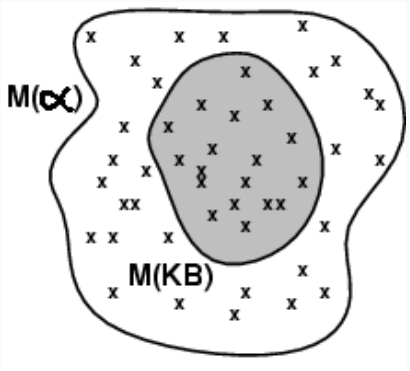
- **Logique** : langage formel permettant de représenter des informations à partir desquelles on peut tirer des conclusions
- La **syntaxe** désigne les phrases (ou énoncés) bien formées dans le langage
- La **sémantique** désigne la signification, le sens de ces phrases
- Par exemple, dans le langage 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
- 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 $M(BC) \subseteq M(\alpha)$



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

Logique propositionnelle

Logique propositionelle - syntaxe

- Les atomes :
  - Constantes logiques $\top$ (vrai) et $\bot$ (faux)
  - **Symbole propositionnel** : proposition qui peut être vraie ou fausse
    a, b, c...
- Les connecteurs logiques :
  - $\neg$ (négation)
  - $\wedge$ (et)
  - $\vee$ (ou)
  - $\Rightarrow$ (implication)
  - $\Leftrightarrow$ (équivalence)
- Un atome (précédé ou non de $\neg$) est appelé un littéral
- Les formules bien formées (wffs)

Logique propositionelle - syntaxe

Formule bien formée

- Tout atome est une wff
- Si E_1 et E_2 sont des wffs, alors
  - $\neg E_1$ est une wwf (négation)
  - $E_1 \wedge E_2$ est une wwf (conjonction)
  - $E_1 \vee E_2$ est une wwf (disjonction)
  - $E_1 \Rightarrow E_2$ est une wwf (implication)
  - $E_1 \Leftrightarrow E_2$ est une wwf (équivalence)
- **Ordre de priorité des opérateurs** : $\neg > \wedge > \vee > \Rightarrow, \Leftrightarrow$

Logique propositionelle - sémantique

- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
- n symboles propositionnels $= 2^n$ modèles possibles
- Règles pour évaluer une wff 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

Table de vérité des connecteurs logiques

E_1	E_2	$\neg E_1$	$E_1 \wedge E_2$	$E_1 \vee E_2$	$E_1 \Rightarrow E_2$	$E_1 \Leftrightarrow E_2$
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*

- La valeur de vérité d'une wff est calculée récursivement en utilisant la table de vérité ci-dessus
- Une wff peut avoir différentes valeurs de vérité dans différentes interprétations (différents modèles)

Equivalence logique

- Deux wffs sont logiquement équivalentes si et seulement si elles sont vraies 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

- Une wff est valide si elle est vraie 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 :

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

Validité et satisfiabilité

- Une wff est valide si elle est vraie 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 :

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

- Une wff est satisfiable si elle est vraie dans certains modèles
  - Exemples : $a \vee b$; c
- Une wff est insatisfiable si elle n'est vraie dans aucun modèle
  - Exemple : $a \wedge \neg a$
- Satisfiabilité :

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

Schémas de raisonnement en logique propositionnelle

Raisonnement en logique propositionnelle

- Raisonnement : Organisation d'un ensemble d'**étapes élémentaires** (inférences)

Raisonnement en logique propositionnelle

- Raisonnement : Organisation d'un ensemble d'**étapes élémentaires** (inférences)
- La machine n'a pas accès à la sémantique
  - Seulement aux propositions de la base de connaissances

  $\Rightarrow$ Comment calculer les conséquences logiques ?

Raisonnement en logique propositionnelle

- Raisonnement : Organisation d'un ensemble d'**étapes élémentaires** (inférences)
- La machine n'a pas accès à la sémantique
  - Seulement aux propositions de la base de connaissances

  $\Rightarrow$ Comment calculer les conséquences logiques ?
- Par manipulation syntaxique des formules
  - La machine n'examine pas les valeurs de vérité des formules

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 à partir des axiomes (grâce à 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 à partir des axiomes (grâce à 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 à partir des axiomes (grâce à 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 $\dfrac{\alpha \Rightarrow \beta, \alpha}{\beta}$

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

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

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

Elimination 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
$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

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

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

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

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

Elimination 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)

Elimination 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)

Elimination 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 un axiome
- 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 $(j, k < i)$.

Méthodes de preuve : stratégies

- Stratégie ascendante :
  - Partir des éléments de la base de connaissance pour en chercher les conséquences, dont celle(s) qui nous intéresse(nt)
- Stratégie guidée par les buts :
  - Partir des conséquences recherchées, et voir quelles règles d'inférence pourraient permettre de les démontrer
  - Démarche récursive

Stratégie ascendante

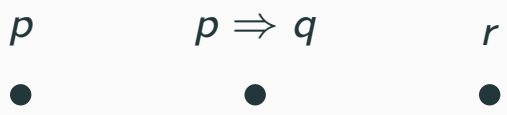
- Partir des éléments de la base de connaissance pour en chercher les conséquences, dont celle(s) qui nous intéresse(nt)

Stratégie ascendante

- Partir des éléments de la base de connaissance pour en chercher les conséquences, dont celle(s) qui nous intéresse(nt)
- Par exemple, est-ce que $q \wedge r$ est conséquence de $\{p, p \Rightarrow q, r\}$?

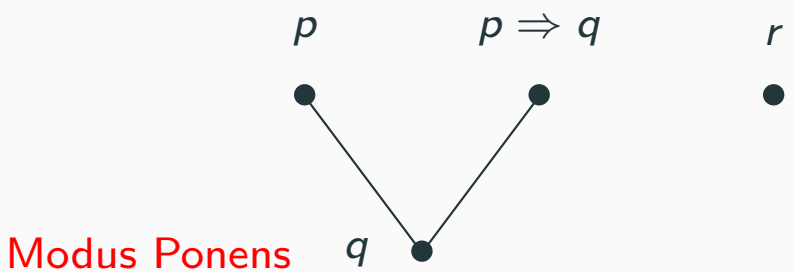
Stratégie ascendante

- Partir des éléments de la base de connaissance pour en chercher les conséquences, dont celle(s) qui nous intéresse(nt)
- Par exemple, est-ce que $q \wedge r$ est conséquence de $\{p, p \Rightarrow q, r\}$?



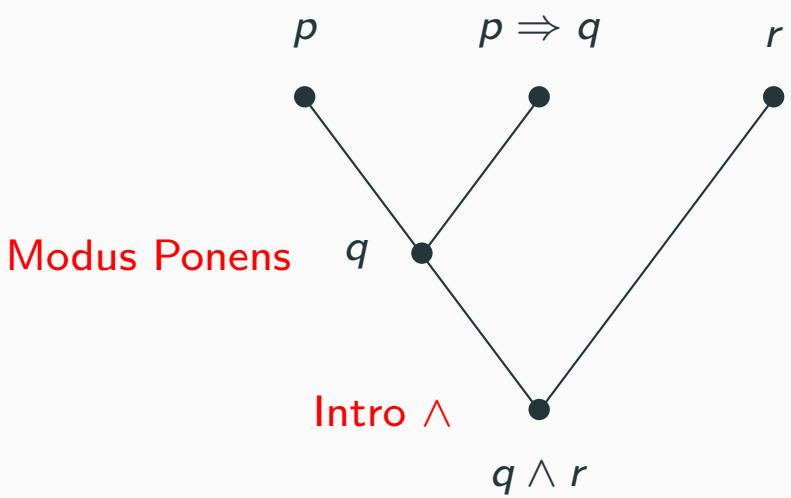
Stratégie ascendante

- Partir des éléments de la base de connaissance pour en chercher les conséquences, dont celle(s) qui nous intéresse(nt)
- Par exemple, est-ce que $q \wedge r$ est conséquence de $\{p, p \Rightarrow q, r\}$?



Stratégie ascendante

- Partir des éléments de la base de connaissance pour en chercher les conséquences, dont celle(s) qui nous intéresse(nt)
- Par exemple, est-ce que $q \wedge r$ est conséquence de $\{p, p \Rightarrow q, r\}$?



Stratégie guidée par les buts

- Partir des conséquences recherchées, et voir quelles règles d'inférence pourraient permettre de les démontrer

Stratégie guidée par les buts

- Partir des conséquences recherchées, et voir quelles règles d'inférence pourraient permettre de les démontrer
- Par exemple, est-ce que $q \wedge r$ est conséquence de $\{p, p \Rightarrow q, r\}$?

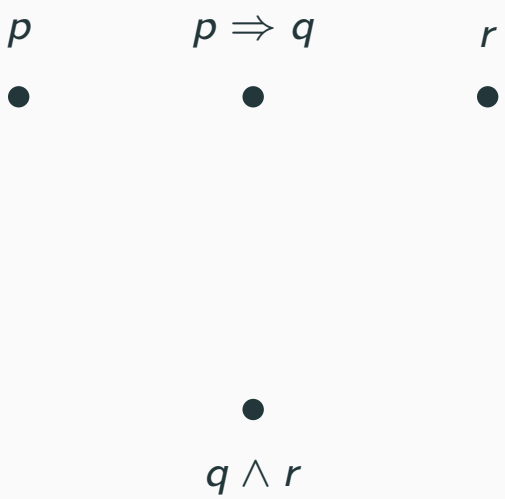
Stratégie guidée par les buts

- Partir des conséquences recherchées, et voir quelles règles d'inférence pourraient permettre de les démontrer
- Par exemple, est-ce que $q \wedge r$ est conséquence de $\{p, p \Rightarrow q, r\}$?

$$\bullet$$
$$q \wedge r$$

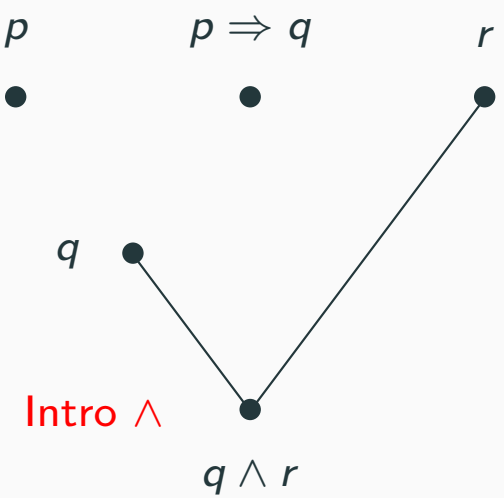
Stratégie guidée par les buts

- Partir des conséquences recherchées, et voir quelles règles d'inférence pourraient permettre de les démontrer
- Par exemple, est-ce que $q \wedge r$ est conséquence de $\{p, p \Rightarrow q, r\}$?



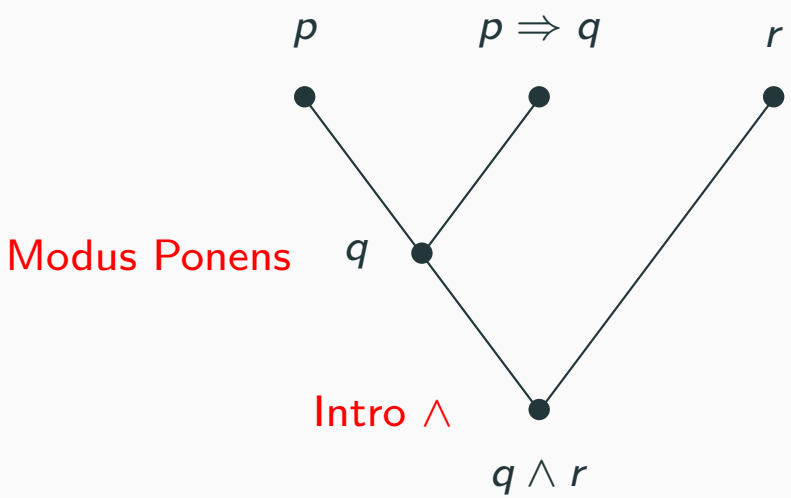
Stratégie guidée par les buts

- Partir des conséquences recherchées, et voir quelles règles d'inférence pourraient permettre de les démontrer
- Par exemple, est-ce que $q \wedge r$ est conséquence de $\{p, p \Rightarrow q, r\}$?



Stratégie guidée par les buts

- Partir des conséquences recherchées, et voir quelles règles d'inférence pourraient permettre de les démontrer
- Par exemple, est-ce que $q \wedge r$ est conséquence de $\{p, p \Rightarrow q, r\}$?



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 conjective (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 conjective (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. Eliminer les équivalences
2. Eliminer les implications
3. Faire migrer les négations "à l'intérieur"
4. Eliminer 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. Eliminer 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. Eliminer les implications

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

Traduction d'une wff en CNF : un exemple

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

$$\equiv \neg(\neg p \wedge (\neg q \Rightarrow r)) \vee s \qquad \text{2. Eliminer les implications}$$

$$\equiv \neg(\neg p \wedge (q \vee r)) \vee s \qquad \text{2. Eliminer les implications}$$

$$\equiv (\neg\neg p \vee \neg(q \vee r)) \vee s \qquad \text{3. Migrer les négations}$$

Traduction d'une wff en CNF : un exemple

$$
\begin{aligned}
& (\neg p \wedge (\neg q \Rightarrow r)) \Rightarrow s \\
\equiv & \neg(\neg p \wedge (\neg q \Rightarrow r)) \vee s \\
\equiv & \neg(\neg p \wedge (q \vee r)) \vee s \\
\equiv & (\neg\neg p \vee \neg(q \vee r)) \vee s \\
\equiv & (\neg\neg p \vee (\neg q \wedge \neg r)) \vee s
\end{aligned}
$$

2. Eliminer les implications
2. Eliminer les implications
    3. Migrer les négations
    3. Migrer les négations

Traduction d'une wff en CNF : un exemple

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

Traduction d'une wff en CNF : un exemple

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

Traduction d'une wff en CNF : un exemple

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

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

- Démonstration par l'absurde : 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

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
  - $(\neg a \vee \neg b \vee c)$ est une clause de Horn
  - $(\neg a \vee b \vee c)$ 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
  - $(\neg a \vee \neg b \vee c) = ((a \wedge b) \Rightarrow c)$
- 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 : exemple

R1. $p \Rightarrow q$

R2. $l \wedge m \Rightarrow p$

R3. $b \wedge l \Rightarrow m$

R4. $a \wedge p \Rightarrow l$

R5. $a \wedge b \Rightarrow l$

R6. a

R7. b

$\rightarrow$ On veut prouver q

Chaînage avant : exemple

R1. $p \Rightarrow q$

R2. $l \wedge m \Rightarrow p$

R3. $b \wedge l \Rightarrow m$

R4. $a \wedge p \Rightarrow l$

R5. $a \wedge b \Rightarrow l$

R6. a

R7. b

$\rightarrow$ On veut prouver q

```
--R6,R7--> a,b
```

Chaînage avant : exemple

R1. $p \Rightarrow q$

R2. $l \wedge m \Rightarrow p$

R3. $b \wedge l \Rightarrow m$

R4. $a \wedge p \Rightarrow l$

R5. $a \wedge b \Rightarrow l$

R6. a

R7. b

$\rightarrow$ On veut prouver q

```
--R6,R7--> a,b
           --R5--> a,b,l
```

Chaînage avant : exemple

R1. $p \Rightarrow q$

R2. $l \wedge m \Rightarrow p$

R3. $b \wedge l \Rightarrow m$

R4. $a \wedge p \Rightarrow l$

R5. $a \wedge b \Rightarrow l$

R6. a

R7. b

$\rightarrow$ On veut prouver q

```
--R6,R7--> a,b
           --R5--> a,b,l
                   --R3--> a,b,l,m
```

Chaînage avant : exemple

R1. $p \Rightarrow q$

R2. $l \wedge m \Rightarrow p$

R3. $b \wedge l \Rightarrow m$

R4. $a \wedge p \Rightarrow l$

R5. $a \wedge b \Rightarrow l$

R6. a

R7. b

$\rightarrow$ On veut prouver q

```
--R6,R7--> a,b
           --R5--> a,b,l
                   --R3--> a,b,l,m
                           --R2--> a,b,m,l,p
```

Chaînage avant : exemple

R1. $p \Rightarrow q$

R2. $l \wedge m \Rightarrow p$

R3. $b \wedge l \Rightarrow m$

R4. $a \wedge p \Rightarrow l$

R5. $a \wedge b \Rightarrow l$

R6. a

R7. b

$\rightarrow$ On veut prouver q

```
--R6,R7--> a,b
           --R5--> a,b,l
                   --R3--> a,b,l,m
                           --R2--> a,b,m,l,p
                                   --R1--> a,b,m,l,p,q
```

Preuve de complétude

- La procédure de chaînage avant permet d'obtenir toute wff atomique pouvant être déduite 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 : exemple

R1. $p \Rightarrow q$

R2. $l \wedge m \Rightarrow p$

R3. $b \wedge l \Rightarrow m$

R4. $a \wedge p \Rightarrow l$

R5. $a \wedge b \Rightarrow l$

R6. a

R7. b

$\rightarrow$ On veut prouver q

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 une wff 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
    - Une wff est fausse 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 `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

Conclusion

Conclusion

- **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'une wwf 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