Septième partie VII

Agents logiques

Plan

En bref ...

Motivations

- Agents fondés sur des 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. Par exemple :
  - 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

Agents fondés sur des connaissances

Base de connaissances

- 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 base de connaissances, 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 base de connaissances

Agent fondé sur des connaissances

- Un agent fondé sur des connaissances doit être capable de :
  1. Représenter les états, les actions
  2. Incorporer de nouvelles perceptions
  3. Mettre à jour sa représentation interne du monde
  4. Déduire les propriétés cachées du monde
  5. Déduire les actions appropriées

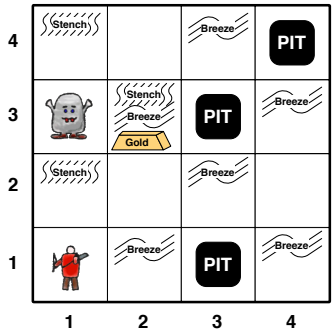
Agent fondé sur des connaissances

Algorithme

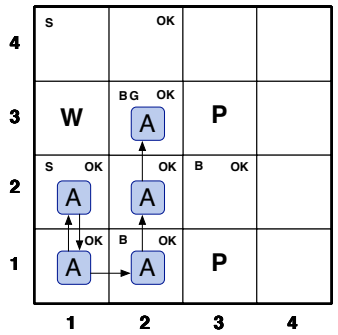
funtion KB-AGENT(*percept*) **return** *an action*
  static: *KB*, *a knowledge base*
    t, *a counter, initially 0, indicating time*
  TELL(*KB*,MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  $t \leftarrow t + 1$
  return *action*

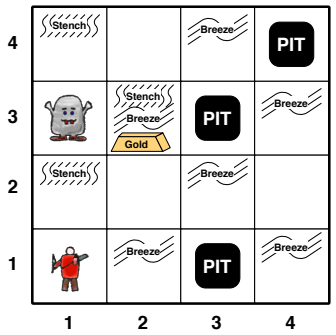
Exemple : le monde du Wumpus

- Mesures de performance :
  - or : +1000 ; mort : -1000 ;
  - action : -1 ; utiliser la flèche : -10
- Environnement
  - Agent commence en case [1,1]
  - Les 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, ouïe, touché
- Actions : tourne gauche, tourne droite, avance, attrappe, tire



Exemple : le monde du Wumpus





Exemple : le monde du Wumpus

- Totalement observable : Non. Perception locale uniquement
- Déterministe : Oui
- Épisodique : 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

Principes généraux de la logique

Principes 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 incorrrecte
  - $x + y = 4$ 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

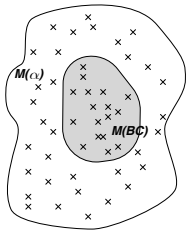
- Relation de conséquence : un énoncé découle logiquement d'un autre énoncé

$$\alpha \models \beta$$

- $\alpha \models \beta$ 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

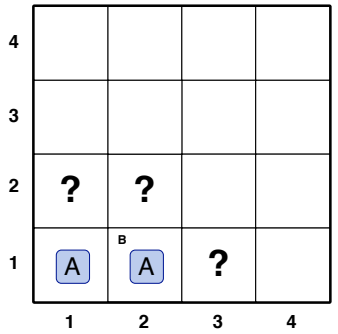
Les modèles

- 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 $M(BC) \subseteq M(\alpha)$

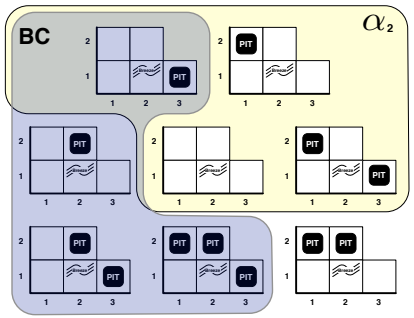


Relation de conséquence dans le mondes du Wumpus

- Situation après n'avoir rien détecté en [1,1]
  - Tourner à droite
  - Brise en [2,1]
- Considérons les modèles possible pour les cases ? en ne considérant que les puits
  - 3 choix booléens $\Rightarrow 2^3 = 8$ modèles possibles



Modèles du monde du Wumpus



- BC = règles du monde Wumpus + observations
- α_2 = "[2,2] est sans puits"
- $BC \not\models \alpha_2$, prouvé par vérification des modèles (model checking)

Logique propositionnelle

Inférence logique

- $BC \vdash_i \alpha$: l'énoncé α est dérivé de BC par la procédure i
- Validité (*soundness*) : i est valide si, lorsque $BC \vdash_i \alpha$ est vrai, alors $BC \models \alpha$ est également vrai
- Complétude (*completness*) : i est complète si lorsque $BC \models \alpha$ est vrai alors $BC \vdash_i \alpha$ 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 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 propositionnelle

- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
  - Soit 3 symboles propositionnels P_1, P_2 et P_3 tels que
    - $m = \{P_1 = vrai, P_2 = vrai, P_3 = faux\}$
- 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*
- L'évaluation d'un énoncé peut être réalisée par un simple processus récursif, e.g.,

$$\neg P_1 \wedge (P_2 \vee P_3) = vrai \wedge (faux \vee vrai) = vrai \wedge vrai = vrai$$

Table de vérité des connecteurs logiques

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

Base de connaissances du monde du Wumpus

- $P_{i,j}$ *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

- 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*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
vrai	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*	*vrai*	*faux*	*vrai*	*vrai*	*faux*	*vrai*	*faux*

Inférence par énumération

Algorithme

```
funtion TT-Entail ?(KB, α) return an true or false
    input: KB, a knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic
    symbols ← a list of propositions symbols in KB and α
    return TT-Check-All(KB, α, symbols, [])

funtion TT-Check-All(KB, α, symbols, model) return an true or false
    if Empty ?(symbols) then
        if PL-True ?(KB, model) then return PL-True ?(α, model)
        else return true
    else
        P ← First(symbols)
        rest ← Rest(symbols)
        return TT-Check-All(KB, α, rest, Extend(P, true, model)) and
               TT-Check-All(KB, α, rest, Extend(P, false, model))
```

Inférence par énumération

- L'énumération en profondeur d'abord est valide et complète
- Pour n symboles, la complexité
  - en temps est de $O(2^n)$
  - en mémoire est de $O(n)$

Équivalence logique

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

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

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

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 :

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

- 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 :

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

Raisonnement et preuve

Méthode de preuve

- Les méthodes de preuves sont de deux principaux types :
  1. 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
  2. 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)

Exemple de preuve

- Problème :
  - Soit la base de connaissances suivantes :
    - $R_1 : \neg P_{1,1}$;
    - $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
    - $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
    - $R_4 : \neg B_{1,1}$
    - $R_5 : B_{2,1}$
  - On veut prouver $\neg P_{1,2}$ (pas de puits en $[1, 2]$)
- Solution
  1. Élimination de l'équivalence à R_2 :

     $$R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

  2. Élimination de la conjonction à R_6 : $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
  3. Équivalence logique des contraposées : $R_8 : \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$
  4. Modus Ponens avec R_8 et R_4 : $R_9 : \neg(P_{1,2} \vee P_{2,1})$
  5. Règle de De Morgan : $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

Résolution par chaînage avant et arrière

- L'inférence peut être réalisée en utilisant une recherche en chaînage avant ou arrière
- Ces algorithmes sont très naturels et s'exécutent en temps linéaire
- Pour fonctionner, ils ont besoins que la base de connaissances soit structurée en conjonction de clauses de Horn
  - Une close clause de Horn =
    - une proprosition ou
    - (une conjonction de proposition) $\Rightarrow$ proposition
  - Par exemple :
    - $A \wedge B \Rightarrow C$ ou A
- L'inférence s'appuie sur le modus ponens

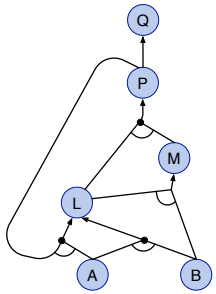
$$\frac{\alpha \wedge \beta \Rightarrow \beta}{\beta}$$

Elle consiste à affirmer une implication (si α alors β) et à poser ensuite l'antécédent (or α) pour en déduire le conséquent (donc β)

Résolution par chaînage avant

- Idée : Choisi n'importe quelle règle dont les prémisses sont satisfaites dans la base de connaissances et ajouter sa conclusion à la base de connaissance, jusqu'a ce que la propriété à vérifier soit trouvée

- Soit la base de connaissances suivantes :
  - $P \Rightarrow Q$
  - $L \wedge M \Rightarrow P$
  - $B \wedge P \Rightarrow M$
  - $A \wedge P \Rightarrow L$
  - $A \wedge B \Rightarrow L$
  - A
  - B

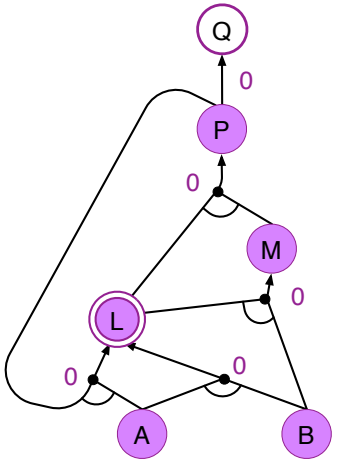


Algorithme d'inférence en chaînage avant

Algorithme

funtion PL-FC-ENTAILS ?(*KB*, *q*) **return** *a true of false*
- **input** : *KB*, *a knowledge base, a set of propositional Horn clauses*
  q, *the query, a proposition symbol*
- **local variables**: *count*, *a table, indexed by clause, initially the number of premises*
  inferred, *a table, indexed by symbol, each entry initially false*
  agenda, *a list of symbols, initially the symbols known to be true in KB*
- **while** *agenda is not empty* **do**
  - $p \leftarrow$ POP(*agenda*)
  - **if** $p = q$ **then return** *true*
  - **if** *inferred*$[p] = false$ **then**
    - *inferred*$[p] \leftarrow true$
    - **foreach** *Horn clause c in whose premise p appears* **do**
      - *decrement count*$[c]$
      - **if** *count*$[c] = 0$ **then**
        - PUSH(HEAD[*c*], *agenda*)
- **return** *false*

Exemple : Chaînage avant



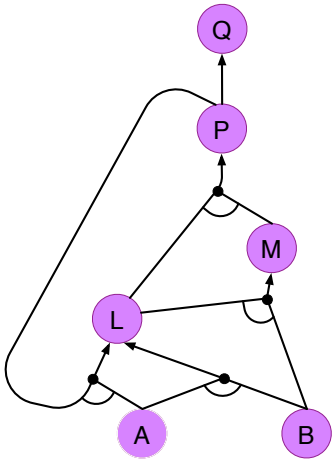
Chaînage avant

- La procédure de chaînage avant permet d'obtenir tout énoncé atomique pouvant être déduit de *BC*
  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 *BC* originellement sont vraies dans *m*
  4. Donc *m* est un modèle de *BC*
  5. Si $BC \models q$ est vrai, *q* est vrai dans tous les modèles de *BC*, donc dans *m*

Chaînage arrière

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

Exemple : Chaînage arrière



Chaînage avant versus chaînage arrière

- Chaînage avant : raisonnement piloté par les données
  - Conclusions à partir de perceptions 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

Forme Normal Conjontive

- La transformation d'un énoncé en CNF s'effectue en 4 étapes :

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

1. Élimination 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. Élimination de $\Rightarrow$: remplacementde $\alpha \Rightarrow \beta$ par $\neg\alpha \vee \beta$

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

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

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

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

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

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