

# Intelligence artificielle

## Agents logiques

---

Elise Bonzon

`elise.bonzon@u-paris.fr`

LIPADE - Université de Paris

<http://www.math-info.univ-paris5.fr/~bonzon/>

# Motivations

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

# Motivations

- Agents fondés sur les connaissances
    - Représentation des connaissances
    - Processus de raisonnement
- ⇒ Tirer parti de connaissances grâce à une capacité à combiner et recombinaison des informations pour les adapter à une multitude de fins.
- Mathématicien démontre un théorème
  - Astronome calcule la durée de vie de la Terre

# Motivations

- Agents fondés sur les connaissances
  - Représentation des connaissances
  - Processus de raisonnement
- ⇒ Tirer parti de connaissances grâce à une capacité à combiner et recombinaison 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**

1. Agents fondés sur les connaissances
2. Le monde du Wumpus
3. Principe généraux de la logique
4. Logique propositionnelle
5. Schémas de raisonnement en logique propositionnelle
6. Agents basés sur la logique propositionnelle
7. Conclusion

## **Agents fondés sur les connaissances**

---

# 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
  - au 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 la base de 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
- Dédire les propriétés cachées du monde
- Dédire 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

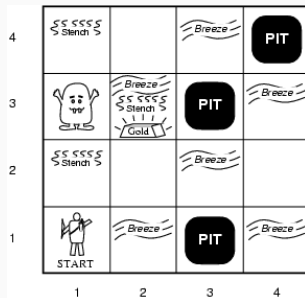
**retourner** *action*

# Le monde du Wumpus

---

# 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 les 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
- Percepteurs : liste de 5 symboles  
Ex : [odeur, brise, rien, rien, rien]
- Actions : tourne gauche, tourne droite, avance, attrape, 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**



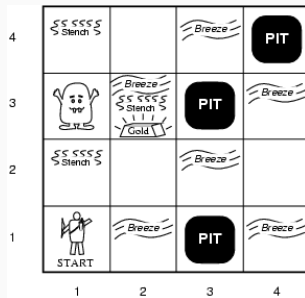
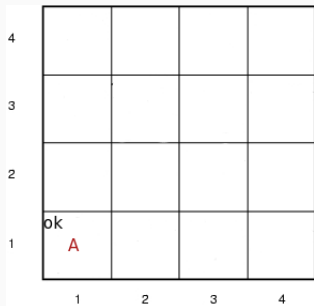
# Caractérisation du monde du Wumpus

- **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
- **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

# Explorer un monde du Wumpus

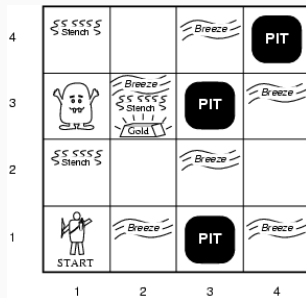
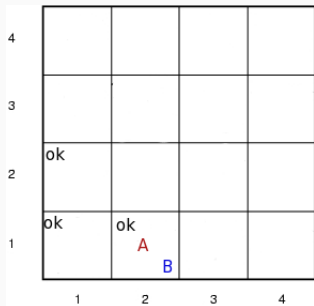


# Explorer un monde du Wumpus

4				
3				
2	ok			
1	ok A	ok		
	1	2	3	4




4	SSSSSS Stench		Breeze	PIT
3	 SSSSSS Stench	Breeze SSSSSS Stench Gold	PIT	Breeze
2	SSSSSS Stench		Breeze	
1	 START	Breeze	PIT	Breeze
	1	2	3	4

# Explorer un monde du Wumpus



# Explorer un monde du Wumpus

4				
3				
2	ok	P?		
1	ok	ok A B	P?	
	1	2	3	4

4	SSSSSS Stench		Breeze	PIT
3	 Breeze SSSSSS Stench 	Breeze	PIT	Breeze
2	SSSSSS Stench		Breeze	
1	 START	Breeze	PIT	Breeze
	1	2	3	4









# Explorer un monde du Wumpus

4				
3				
2	ok	P?		
1	ok A	ok B	P?	
	1	2	3	4

4	SSSSSS Stench		Breeze	PIT
3	Wumpus	Breeze SSSSSS Stench Gold	PIT	Breeze
2	SSSSSS Stench		Breeze	
1	START	Breeze	PIT	Breeze
	1	2	3	4

# Explorer un monde du Wumpus













4				
3				
2	ok A O	P?		
1	ok	ok B	P?	
	1	2	3	4

4	 Stench	Breeze	 PIT	
3	 Stench  Gold	Breeze Stench  PIT	Breeze	
2	 Stench	Breeze		
1	 START	Breeze  PIT	Breeze	
	1	2	3	4















# Explorer un monde du Wumpus

4				
3	W!			
2	ok A O	P?		
1	ok	ok	P?	
	1	2	3	4

4	 Stench	 Breeze	 PIT	
3	 Stench	 Stench	 PIT	
2	 Stench	 Gold	 Breeze	
1	 START	 Breeze	 PIT	
	1	2	3	4

# Explorer un monde du Wumpus

4				
3	W!			
2	ok A O	ok		
1	ok	ok B	P!	
	1	2	3	4

4	 Stench	 Breeze	 PIT	
3	 Stench	 Stench	 PIT	
2	 Stench	 Gold	 Breeze	
1	 START	 Breeze	 PIT	
	1	2	3	4
















# Explorer un monde du Wumpus

4				
3	W!	ok		
2	ok O	ok A	ok	
1	ok	ok B	P!	
	1	2	3	4

4	SSSSSS Stench		Breeze	PIT
3	Wumpus	Breeze SSSSSS Stench Gold	PIT	Breeze
2	SSSSSS Stench		Breeze	
1	START	Breeze	PIT	Breeze
	1	2	3	4

# Explorer un monde du Wumpus

4				
3	W!	ok A L O B		
2	ok O	ok	ok	
1	ok	ok B	P!	
	1	2	3	4

4	 Stench	 Breeze		
3		 Breeze  Stench  Gold		 Breeze
2	 Stench		 Breeze	
1	 START	 Breeze		 Breeze
	1	2	3	4

# Principe généraux de la logique

---

# Principe généraux de la logique

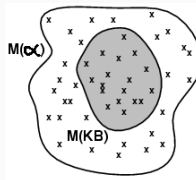
- **Logique** : langage formel permettant de représenter des informations à partir desquelles on peut tirer des conclusions
- La **syntaxe** désigne les phrases (ou énoncés) bien formées dans le langage
- La **sémantique** désigne la signification, le sens de ces phrases
- Par exemple, dans le langage 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  $\beta$  est vraie dans tous mondes où  $\alpha$  est vraie
  - Si  $\alpha$  est vraie,  $\beta$  doit être vraie
  - Par exemple,  $(x + y = 4) \models (x + y \leq 4)$
- Base 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  $\alpha$  si  $\alpha$  est vraie dans  $m$
- $M(\alpha)$  est l'ensemble de tous les modèles de  $\alpha$
- $BC \models \alpha$  si et seulement si  $M(BC) \subseteq M(\alpha)$



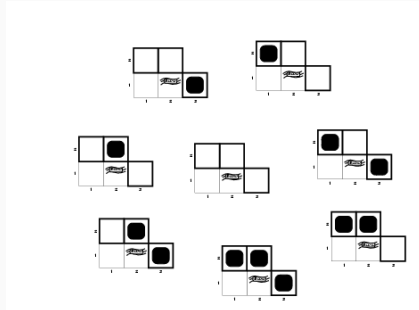


# Relation de conséquences dans le monde du Wumpus

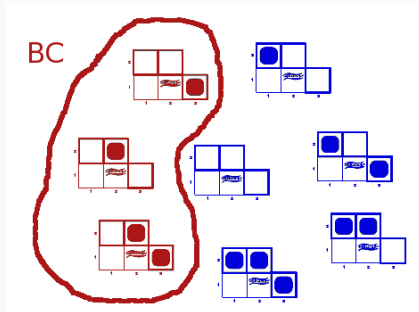
- Situation après avoir effectué
  - Rien en [1,1]
  - Droite
  - Brise en [2,1]
- Considérer les modèles possible pour la base de connaissances en ne considérant que les puits
- $2^3 = 8$  modèles possibles

4				
3				
2	?	?		
1	OK	OK A B	?	
	1	2	3	4

# Modèles du monde du Wumpus

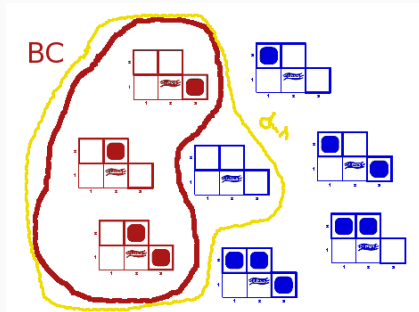


# Modèles du monde du Wumpus



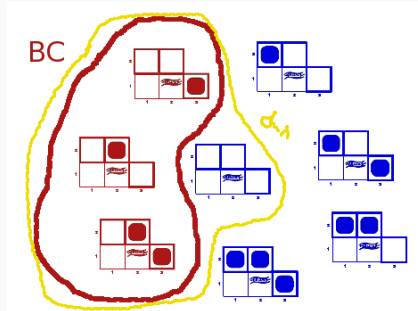
- $BC = \text{règles du monde Wumpus} + \text{observations}$

# Modèles du monde du Wumpus



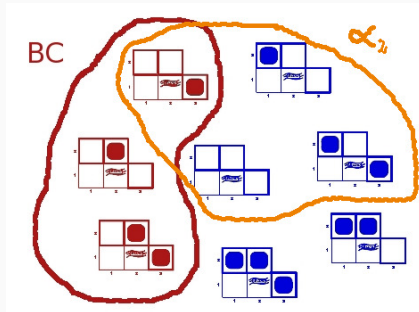
- $BC$  = règles du monde Wumpus + observations
- $\alpha_1$  = “[1,2] est sans puits”

# Modèles du monde du Wumpus



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

# Modèles du monde du Wumpus



- $BC = \text{règles du monde Wumpus} + \text{observations}$
- $\alpha_2 = "[2,2] \text{ est sans puits}"$
- $BC \neq \alpha_2$

- $KB \vdash_i \alpha$  : l'énoncé  $\alpha$  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** (**completeness**) :  $i$  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 propositionnelle

---

## Syntaxe

- Les **atomes** :
  - Constantes logiques  $\top$  (vrai) et  $\perp$  (faux)
  - **Symbole propositionnel** : proposition qui peut être vraie ou fausse  $a, b, c \dots$
- 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)

## Formule bien formée

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

- $P_{i,j}$  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 :  $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$

# Logique propositionnelle

---

## Sémantique

- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
  - 3 symboles propositionnels  $P_{1,1}$ ,  $P_{2,2}$  et  $P_{3,1}$
  - $m_1 = \{P_{1,1} = \text{Faux}, P_{2,2} = \text{Faux}, P_{3,1} = \text{Vrai}\}$

- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
  - 3 symboles propositionnels  $P_{1,1}$ ,  $P_{2,2}$  et  $P_{3,1}$
  - $m_1 = \{P_{1,1} = \text{Faux}, P_{2,2} = \text{Faux}, P_{3,1} = \text{Vrai}\}$
- $n$  symboles propositionnels =  $2^n$  modèles possibles

- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
  - 3 symboles propositionnels  $P_{1,1}$ ,  $P_{2,2}$  et  $P_{3,1}$
  - $m_1 = \{P_{1,1} = \text{Faux}, P_{2,2} = \text{Faux}, P_{3,1} = \text{Vrai}\}$
- $n$  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



- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
  - 3 symboles propositionnels  $P_{1,1}$ ,  $P_{2,2}$  et  $P_{3,1}$
  - $m_1 = \{P_{1,1} = \text{Faux}, P_{2,2} = \text{Faux}, P_{3,1} = \text{Vrai}\}$
- $n$  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

- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
  - 3 symboles propositionnels  $P_{1,1}$ ,  $P_{2,2}$  et  $P_{3,1}$
  - $m_1 = \{P_{1,1} = \text{Faux}, P_{2,2} = \text{Faux}, P_{3,1} = \text{Vrai}\}$
- $n$  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 <b>et</b> $E_2$ est vrai
$E_1 \vee E_2$	est vrai ssi	$E_1$ est vrai <b>ou</b> $E_2$ est vrai

- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
  - 3 symboles propositionnels  $P_{1,1}$ ,  $P_{2,2}$  et  $P_{3,1}$
  - $m_1 = \{P_{1,1} = \text{Faux}, P_{2,2} = \text{Faux}, P_{3,1} = \text{Vrai}\}$
- $n$  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

- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
  - 3 symboles propositionnels  $P_{1,1}$ ,  $P_{2,2}$  et  $P_{3,1}$
  - $m_1 = \{P_{1,1} = \text{Faux}, P_{2,2} = \text{Faux}, P_{3,1} = \text{Vrai}\}$
- $n$  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 <b>et</b> $E_2$ est vrai
$E_1 \vee E_2$	est vrai ssi	$E_1$ est vrai <b>ou</b> $E_2$ est vrai
$E_1 \Rightarrow E_2$	est vrai ssi	$E_1$ est faux <b>ou</b> $E_2$ est vrai
$E_1 \Rightarrow E_2$	est faux ssi	$E_1$ est vrai <b>et</b> $E_2$ est faux

- Un modèle : une valeur de vérité (*vrai* ou *faux*) pour chaque symbole propositionnel
  - 3 symboles propositionnels  $P_{1,1}$ ,  $P_{2,2}$  et  $P_{3,1}$
  - $m_1 = \{P_{1,1} = \text{Faux}, P_{2,2} = \text{Faux}, P_{3,1} = \text{Vrai}\}$
- $n$  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 <b>et</b> $E_2$ est vrai
$E_1 \vee E_2$	est vrai ssi	$E_1$ est vrai <b>ou</b> $E_2$ est vrai
$E_1 \Rightarrow E_2$	est vrai ssi	$E_1$ est faux <b>ou</b> $E_2$ est vrai
$E_1 \Rightarrow E_2$	est faux ssi	$E_1$ est vrai <b>et</b> $E_2$ est faux
$E_1 \Leftrightarrow E_2$	est vrai ssi	$E_1 \Rightarrow E_2$ est vrai <b>et</b> $E_2 \Rightarrow E_1$ est vrai

# Table de vérité des connecteurs logiques

$P$	$Q$	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>vrai</i>	<i>vrai</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>
<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>
<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>
<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>

# Table de vérité des connecteurs logiques

$P$	$Q$	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>vrai</i>	<i>vrai</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>
<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>
<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>
<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>

- 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**)

# 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
<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>
<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>
<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<u><i>vrai</i></u>
<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<u><i>vrai</i></u>
<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<u><i>vrai</i></u>
<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>faux</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>	<i>vrai</i>	<i>vrai</i>	<i>faux</i>	<i>vrai</i>	<i>faux</i>



# Logique propositionnelle

---

Inférence par énumération

## Enumération en profondeur d'abord de tous les modèles

**fonction** TT-Entails( $KB, \alpha$ ) **retourne** *vrai* ou *faux*

**variables statiques** :  $KB$ , base de connaissances

$\alpha$ , requête, énoncé propositionnel

$symboles \leftarrow$  liste de symboles propositionnels dans  $KB$  et  $\alpha$

**retourner** TT-Check-All( $KB, \alpha, symboles, []$ )

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

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

**si** PL-True?( $KB, modele$ ) **alors retourner** PL-True?( $\alpha, modele$ )

**sinon retourner** *vrai*

**sinon faire**

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

**retourner** TT-Check-All( $KB, \alpha, reste, Extend(P, vrai, modele)$ )

**et** TT-Check-All( $KB, \alpha, reste, Extend(P, faux, modele)$ )

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

# Logique propositionnelle

---

Equivalence, validité, satisfiabilité

# Equivalence logique

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

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

# Equivalence logique

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

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

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \text{ commutativité de } \wedge$$

# Equivalence logique

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

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

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \text{ commutativité de } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \text{ commutativité de } \vee$$

# Equivalence logique

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

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

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \text{ commutativité de } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \text{ commutativité de } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \text{ associativité de } \wedge$$



# Equivalence logique

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

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

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \text{ commutativité de } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \text{ commutativité de } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \text{ associativité de } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \text{ associativité de } \vee$$

# Equivalence logique

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

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

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \text{ commutativité de } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \text{ commutativité de } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \text{ associativité de } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \text{ associativité de } \vee$$

$$\neg(\neg\alpha) \equiv \alpha \text{ élimination de la double négation}$$

# Equivalence logique

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

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

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \text{ commutativité de } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \text{ commutativité de } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \text{ associativité de } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \text{ associativité de } \vee$$

$$\neg(\neg\alpha) \equiv \alpha \text{ élimination de la double négation}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \text{ contraposition}$$

# Equivalence logique

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

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

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativité de $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativité de $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativité de $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativité de $\vee$
$\neg(\neg\alpha)$	$\equiv$	$\alpha$	élimination de la double négation
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	élimination de l'implication

# Equivalence logique

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

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

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativité de $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativité de $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativité de $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativité de $\vee$
$\neg(\neg\alpha)$	$\equiv$	$\alpha$	élimination de la double négation
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	élimination de l'implication
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	élimination de l'équivalence

# Equivalence logique

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

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

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativité de $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativité de $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativité de $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativité de $\vee$
$\neg(\neg\alpha)$	$\equiv$	$\alpha$	élimination de la double négation
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	élimination de l'implication
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	élimination de l'équivalence
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	De Morgan

# Equivalence logique

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

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

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativité de $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativité de $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativité de $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativité de $\vee$
$\neg(\neg\alpha)$	$\equiv$	$\alpha$	élimination de la double négation
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	élimination de l'implication
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	élimination de l'équivalence
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta)$	$\equiv$	$(\neg\alpha \wedge \neg\beta)$	De Morgan

# Equivalence logique

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

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

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativité de $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativité de $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativité de $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativité de $\vee$
$\neg(\neg\alpha)$	$\equiv$	$\alpha$	élimination de la double négation
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	élimination de l'implication
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	élimination de l'équivalence
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta)$	$\equiv$	$(\neg\alpha \wedge \neg\beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma))$	$\equiv$	$((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivité de $\wedge$ par rapport à $\vee$



# Equivalence logique

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

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

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativité de $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativité de $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativité de $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativité de $\vee$
$\neg(\neg\alpha)$	$\equiv$	$\alpha$	élimination de la double négation
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	élimination de l'implication
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	élimination de l'équivalence
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta)$	$\equiv$	$(\neg\alpha \wedge \neg\beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma))$	$\equiv$	$((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivité de $\wedge$ par rapport à $\vee$
$(\alpha \vee (\beta \wedge \gamma))$	$\equiv$	$((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivité de $\vee$ par rapport à $\wedge$

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

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

## Théorème de la déduction

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

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

## 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 **valide** s'il est vrai dans **tous** les modèles. On dit aussi **tautologie**
  - Exemples :  $\top$  ;  $A \vee \neg A$  ;  $A \Rightarrow A$  ;  $(A \wedge (A \Rightarrow B)) \Rightarrow B$

## 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 :  $A \wedge \neg A$

# Validité et satisfiabilité

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

## 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 :  $A \wedge \neg A$

## Théorème de la déduction

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

# Schémas de raisonnement en logique propositionnelle

---

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  $\alpha$  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  $\alpha$  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  $BC \vdash_i \alpha$

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

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



# 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

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}{\alpha_i}$$

# 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

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}{\alpha_i}$$

Introduction de la conjonction

$$\frac{\alpha_1, \alpha_2, \dots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}$$

# 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

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}{\alpha_i}$$

Introduction de la conjonction

$$\frac{\alpha_1, \alpha_2, \dots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}$$

Introduction de la disjonction

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_n}$$

# 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	$\frac{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}{\alpha_i}$
Introduction de la conjonction	$\frac{\alpha_1, \alpha_2, \dots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}$
Introduction de la disjonction	$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_n}$
Elimination de la double négation	$\frac{\neg \neg \alpha}{\alpha}$

Elimination de l'équivalence

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

## 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

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



## Déductions

Une formule  $A$  **se déduit** d'un ensemble de formules  $\{B_1, B_2, \dots, B_n\}$ , noté  $B_1, B_2, \dots, B_n \vdash A$  s'il existe une suite finie  $(A_1, A_2, \dots, A_i, \dots, A)$ , 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  $[1, 2]$ )
- **Elimination de l'équivalence** à  $R_2$  :  
 $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

## 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  $R_6$  :  
 $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$

## 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** :  
 $R_8 : \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$

## 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  $R_4$  :  
 $R_9 : \neg(P_{1,2} \vee P_{2,1})$

## 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** :  
 $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

# 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

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

⇒ Besoin d'avoir une forme standardisée ou canonique

## Forme normale conjunctive (CNF)

Forme normale conjunctive (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

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

⇒ Besoin d'avoir une forme standardisée ou canonique

## Forme normale conjunctive (CNF)

Forme normale conjunctive (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

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

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

2. Eliminer les implications

## Traduction d'une wff en CNF : un exemple

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

2. Eliminer les implications

2. Eliminer les implications



## Traduction d'une wff en CNF : un exemple

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

2. Eliminer les implications

2. Eliminer les implications

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

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

2. Eliminer les implications

2. Eliminer les implications

3. Migrer les négations

3. Migrer les négations

4. Eliminer les doubles négations

## Traduction d'une wff en CNF : un exemple

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

2. Eliminer les implications

2. Eliminer les implications

3. Migrer les négations

3. Migrer les négations

4. Eliminer les doubles négations

5. Distribuer les  $\wedge$  sur les  $\vee$

## Traduction d'une wff en CNF : un exemple

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

2. Eliminer les implications

2. Eliminer les implications

3. Migrer les négations

3. Migrer les négations

4. Eliminer les doubles négations

5. Distribuer les  $\wedge$  sur les  $\vee$

5. Distribuer les  $\wedge$  sur les  $\vee$

- 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  $(p \vee r)$

- 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

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

- 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

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

- Le modus ponens est un cas particulier de la 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

## Algorithme de résolution

**fonction** PL-Resolution( $KB, \alpha$ ) **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*  $\leftarrow$  *clauses*  $\cup$  *nouveau*

# Schémas de raisonnement en logique propositionnelle

---

Systèmes à base de règles

- **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

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

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

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

- **Idée** : appliquer toutes les règles dont les prémisses sont satisfaites 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**

$p \leftarrow \text{Pop}(\text{agenda})$

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

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

$\text{infer}[p] \leftarrow \text{vrai}$

**pour chaque** clause de Horn *c* dans laquelle la prémisses *p* apparaît **faire**

$\text{compteur}[c] \leftarrow \text{compteur}[c] - 1$

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

**retourner** *faux*

## Chaînage avant : exemple

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

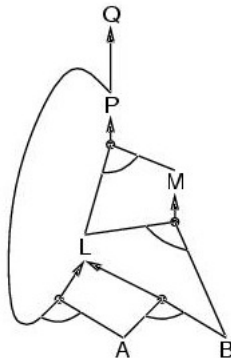
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

$A$

$B$





- 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  $m$

- **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

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

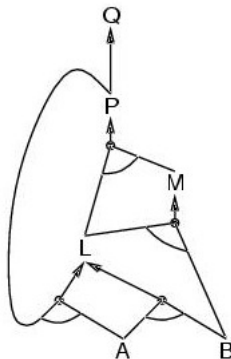
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

$A$

$B$



# 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

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

- Cet algorithme détermine si un énoncé logique en CNF est satisfiable

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



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

- 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

- 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

- 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**

- 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

- 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

- 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*

- 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**



- 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

- 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 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

**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*

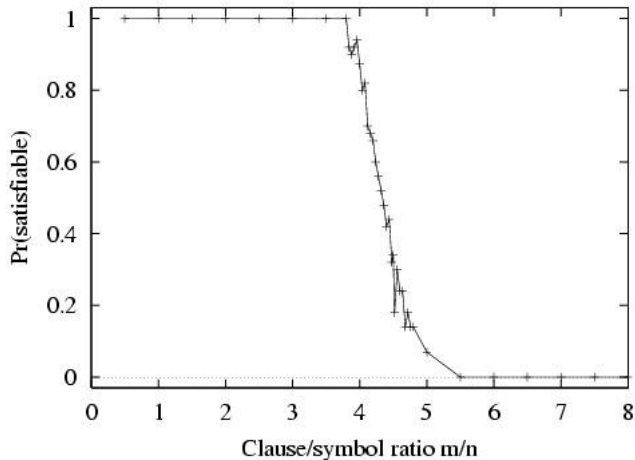
**sinon** basculer le symbole dans *clause* qui maximise le nombre de clauses  
satisfaites

**retourner** *erreur*

# Problèmes de satisfiabilité difficiles

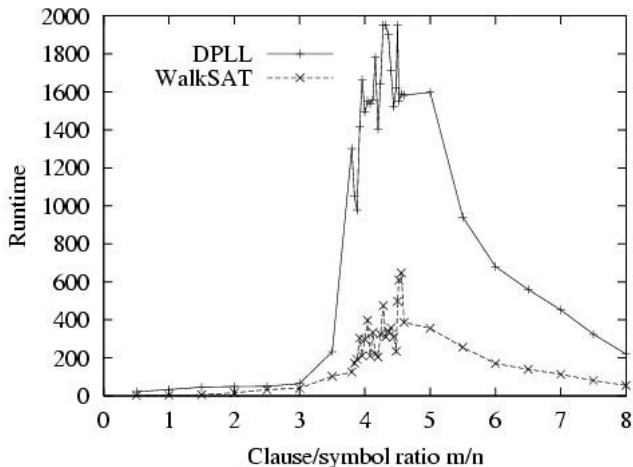
- Soit l'énoncé 3-CNF généré aléatoirement suivant :  
 $(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$
- 16 des 32 affectations possibles sont des modèles de cet énoncé  
→ en moyenne 2 tentatives aléatoires pour trouver un modèle
- Problème difficile : augmenter le nombre de clauses en laissant fixe le nombre de symboles  
→ 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

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

$\Rightarrow$  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

$$L_{x,y}^t \wedge \textit{droite}^t \textit{avance}^t \Rightarrow L_{x+1,y}^{t+1}$$

- 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
- 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**