

INF8175 - Intelligence artificielle

Méthodes et algorithmes

Module 5: Agents logiques



**POLYTECHNIQUE
MONTRÉAL**

Quentin Cappart

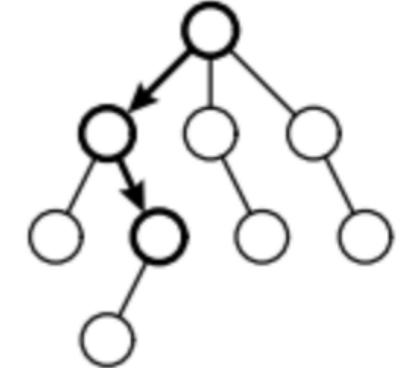
Contenu du cours

Raisonnement par recherche (essais-erreurs avec de l'intuition)

Module 1: Stratégies de recherche

Module 2: Recherche en présence d'adversaires

Module 3: Recherche locale



Raisonnement logique

Module 4: Programmation par contraintes

Module 5: Agents logiques



ΣΣΣΣΣ Stench		Breeze	PIT
	Breeze	ΣΣΣΣΣ Stench Gold	PIT
ΣΣΣΣΣ Stench		Breeze	
START	Breeze	PIT	Breeze

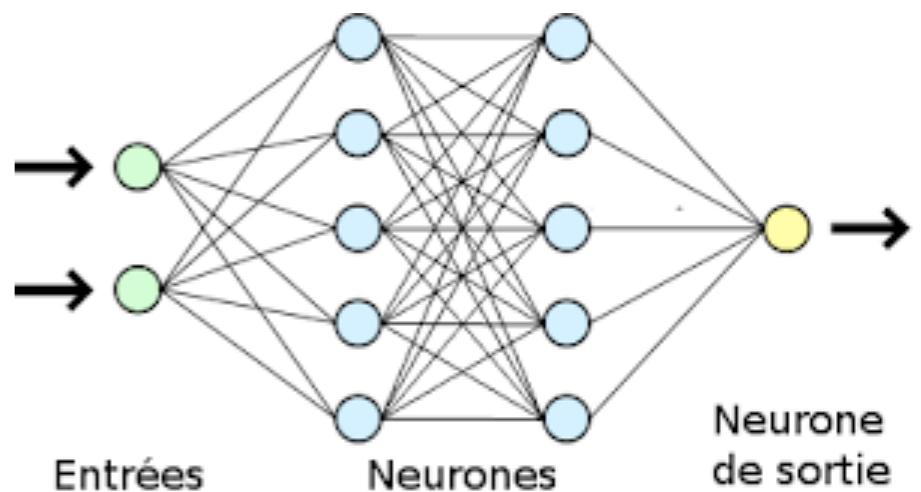
Raisonnement par apprentissage

Module 6: Apprentissage supervisé

Module 7: Réseaux de neurones et apprentissage profond

Module 8: Apprentissage non-supervisé

Module 9: Apprentissage par renforcement



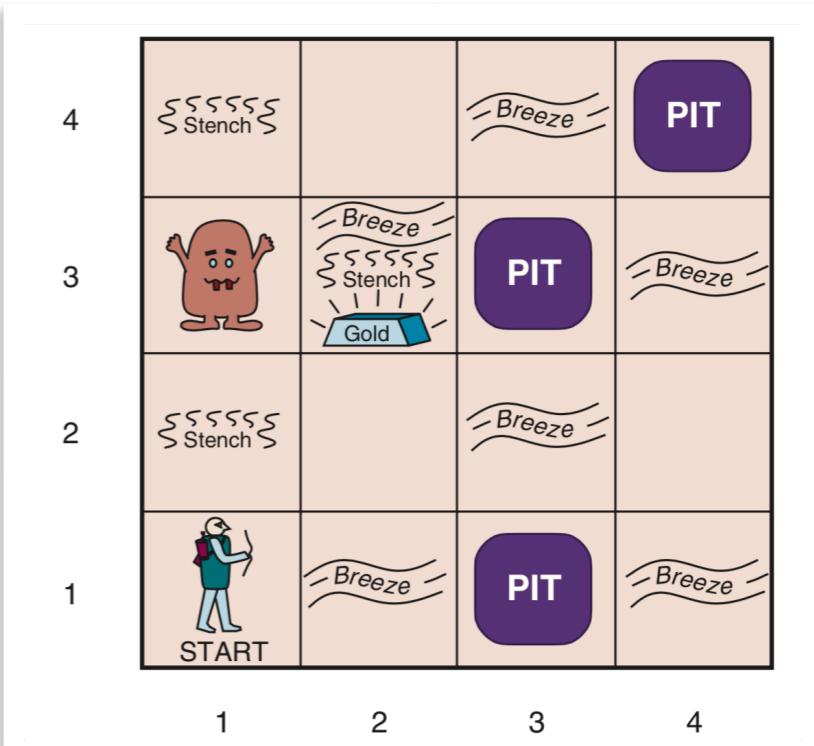
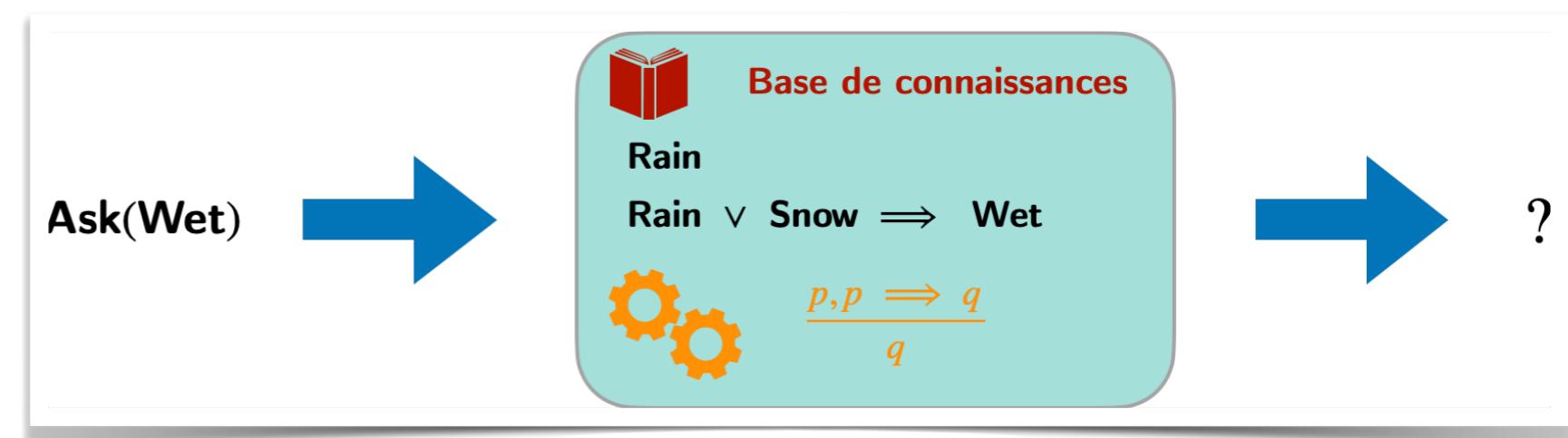
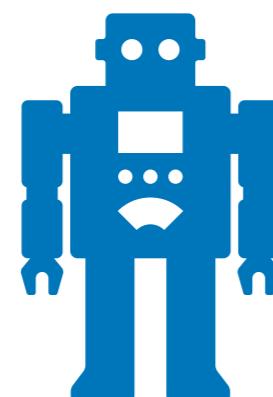
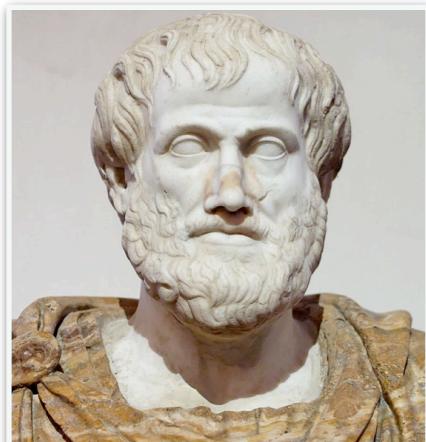
Considérations pratiques et sociétales

Module 10: Utilisation en industrie, éthique, et philosophie

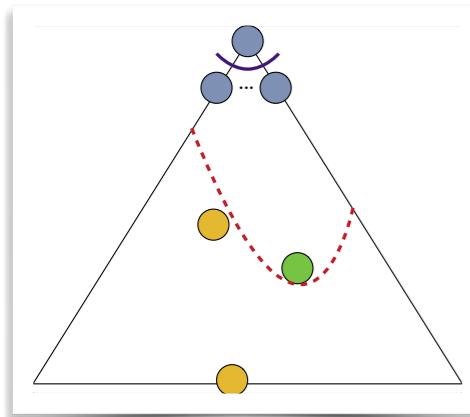
Table des matières

Agents logiques

1. Motivation et définition d'un agent logique
2. Représentation des connaissances
3. Système logique (syntaxe, sémantique, et règles d'inférences)
4. Formalisation de la logique des propositions
5. Raisonnement par *model-checking*
6. Inférences logiques et propriétés
7. Algorithme de résolution
8. Clauses de Horn
9. Extension à d'autres systèmes logiques



Retour sur les modules précédents

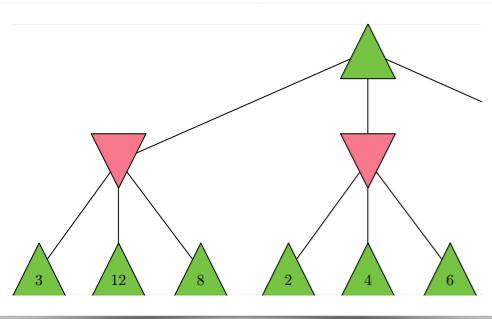


Stratégies de recherche (Module 1)

Objectif: trouver une séquence d'action de plus faible coût entre un état initial et final

Techniques vues: DFS, BFS, IDS, UCS, Greedy, A*

Principe: tester systématiquement des états jusqu'à trouver un chemin

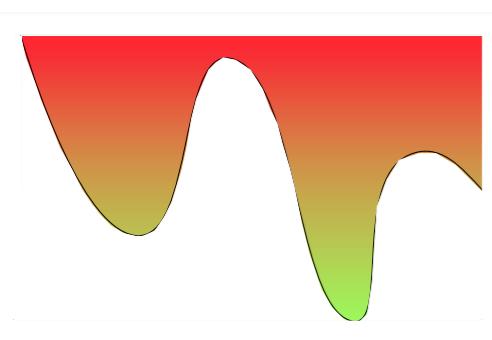


Stratégies de recherche avec adversaires (Module 2)

Objectif: trouver la meilleure action à effectuer, en présence d'adversaires

Techniques vues: Minimax, Alpha-beta pruning, MCTS, Expectiminimax

Principe: tester des scénarios, et prendre l'action du scénario maximisant notre score

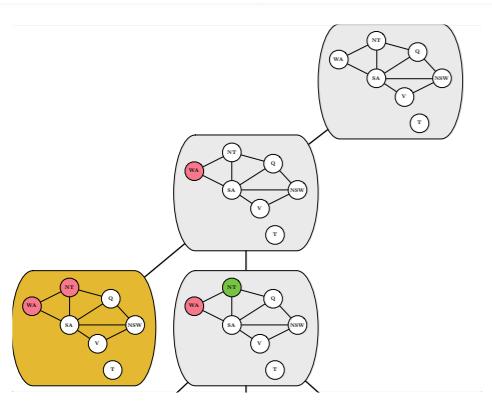


Recherche locale (Module 3)

Objectif: trouver la meilleure solution faisable, parmi un ensemble de solutions

Techniques vues: *hill climbing*, méthode des *restarts*, *simulated annealing*

Principe: tester des états en se déplaçant de solutions en solutions



Programmation par contraintes (Module 4)

Objectif: trouver la meilleure solution faisable, et obtenir des garanties

Techniques vues: *backtracking search*, *arc consistency*, algorithme du point fixe

Principe: tester des solutions, en assignant successivement les variables

Limitation des stratégies de recherche

Stratégies de recherche

Idée générale: résoudre le problème en testant, d'une façon ou d'une autre, différentes configurations

En pratique: intégration de mécanismes pour améliorer la recherche (heuristiques, propagation, etc.)



Est-ce que ce type de raisonnement est adapté dans toutes les situations ?

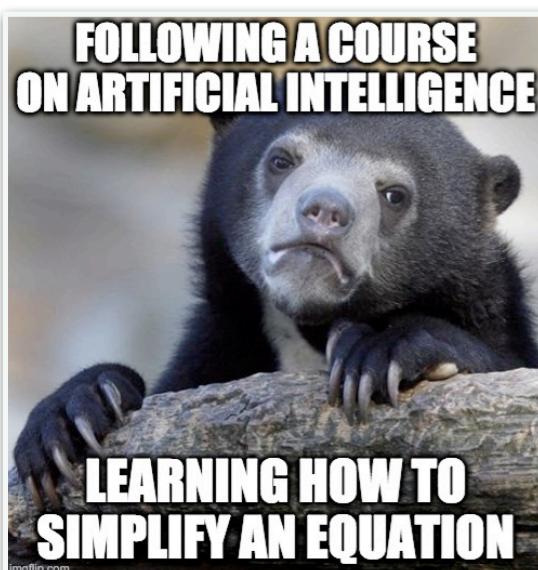
$$\begin{aligned} X_1 + X_2 &= 10 \\ X_1 - X_2 &= 4 \\ X_1 &=? \end{aligned}$$

Exemple: résolution d'un système d'équations simples

(1) Isolation d'un terme: $X_2 = X_1 - 4$

(2) Substitution d'un terme: $X_1 + X_1 - 4 = 10$

(3) Simplification algébrique: $X_1 = 7$



Résolution que vous savez faire depuis de nombreuses années

Algorithme: simple application de plusieurs règles arithmétiques

Limitation: très peu efficace d'exécuter une recherche pour ce problème !

Observation: certains problèmes se résolvent plus efficacement par d'autres stratégies

Ce module est dédié à la résolution de problèmes par raisonnements logiques automatisés

Programmation logique

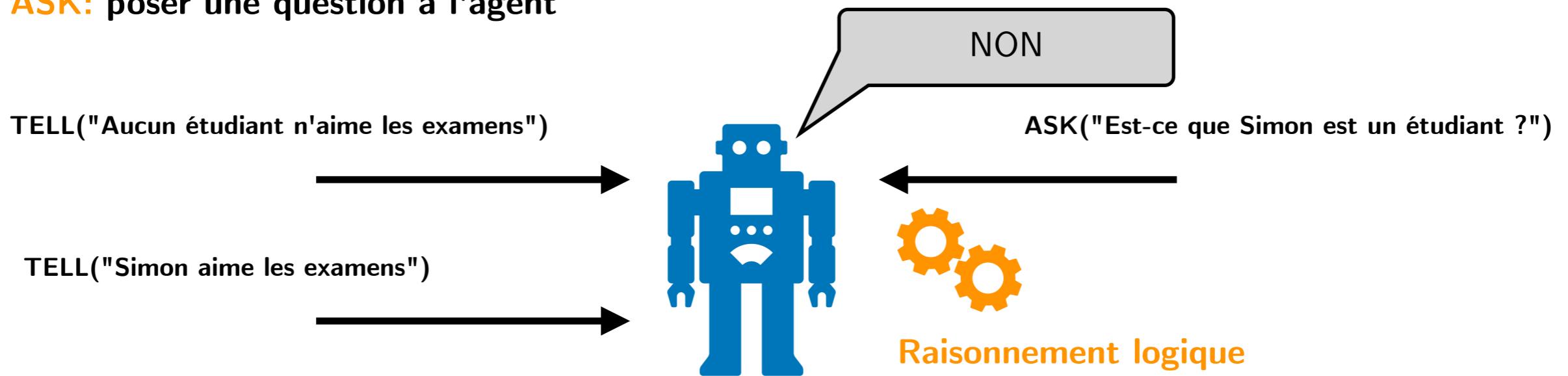
Agent logique

Motivation: créer un agent capable de raisonner et d'agir sur base d'un ensemble de connaissances

Souhait: deux interactions sont souhaitées avec l'agent

(1) **TELL:** donner des nouvelles connaissances à l'agent

(2) **ASK:** poser une question à l'agent



Base de connaissances

Aucun étudiant n'aime les examens

Simon aime les examens

Simon n'est pas un étudiant

Besoins de l'agent

- (1) Etre capable d'intégrer des connaissances très diverses
- (2) Etre capable de raisonner sur base des connaissances disponibles

Objectif 1: avoir un raisonnement supérieur à une simple énumération des connaissances disponibles

Objectif 2: concevoir un agent capable d'inférer par lui même de nouvelles connaissances

Intégration des connaissances



Quel langage utiliser pour représenter les connaissances ?

Votre langue maternelle (ou langue naturelle)

Objectif: créer un agent capable d'ingérer et de raisonner sur base de phrases en français (ou autre)

Objectif idéal: on souhaite interagir avec l'agent comme on le ferait avec une autre personne

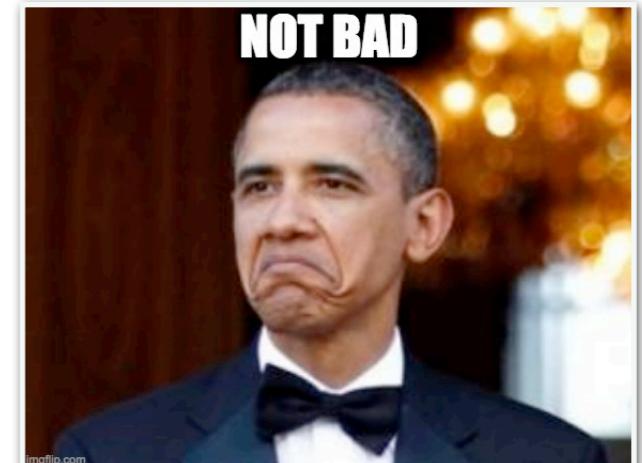
L'argent est plus précieux que le bronze

L'or est plus précieux que l'argent

L'or est plus précieux que le bronze



Raisonnement logique



A dollar is better than nothing

Nothing is better than world peace

A dollar is better than world peace



Raisonnement logique



Quel est le soucis ?

Le raisonnement logique en tant que tel est valide

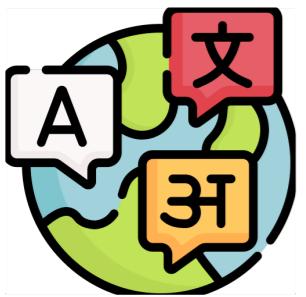
Difficulté: la langue naturelle est souvent ambiguë

Ambiguïté: *Nothing* a un sens à la fois concret et abstrait

Notion de langage

Un langage est un mécanisme pour exprimer quelque chose à une entité (communiquer)

Objectif: on souhaite que l'entité soit capable de comprendre ce qui lui est exprimé



Langue naturelle: langage informel, pour la communication dans notre vie quotidienne

Français: Les nombres pairs sont divisibles par 2

Anglais: Even numbers are divisible by 2

Langage de programmation: langage formel, pour la communication avec un ordinateur

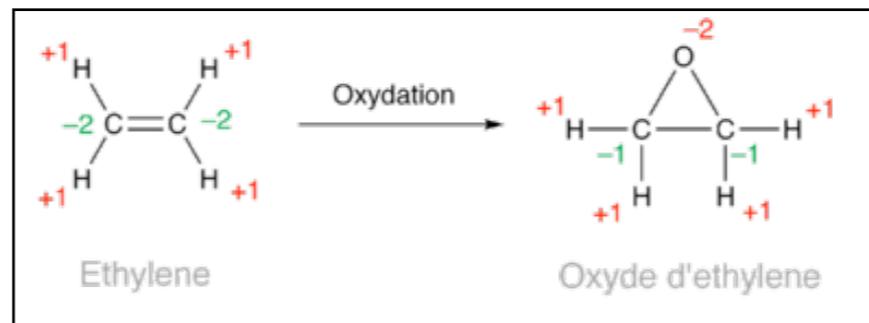
Python: `def even(x): return x % 2 == 0`

C++: `bool even(int x) { return x % 2 == 0; }`

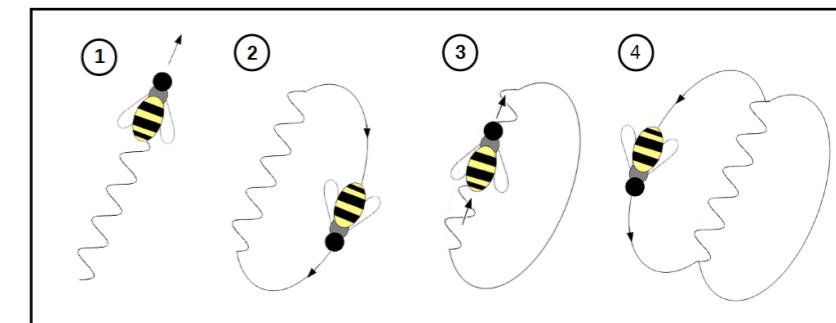
Ce ne sont que deux exemples, et il existe plein d'autres langages dans le monde

$$(\nabla_X Y)^k = X^i (\nabla_i Y)^k = X^i \left(\frac{\partial Y^k}{\partial x^i} + \Gamma_{im}^k Y^m \right)$$

Mathématique



Chimie

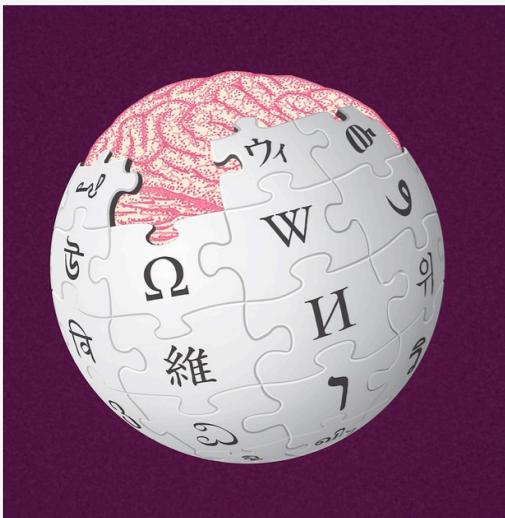


Danse des abeilles

Éléments caractéristiques: les forces des interlocuteurs, l'objet à communiquer, et l'objectif en découlant

Langage logique

Les langages logiques visent à remédier à l'ambiguïté des langues naturelles pour faire des raisonnements



Objectif 1: possibilité de représenter les connaissances relatives à l'environnement

Objectif 2: possibilité de raisonner à partir de ces connaissances

Attention: il n'existe pas un unique langage logique mais une multitude

(1) Logique des propositions

$$f : \text{Raining} \wedge \text{Snowing} \implies \text{Dangerous}$$



(2) Logique du premier ordre

$$f : \forall x \text{ EvenNumber}(x) \implies \text{Divides}(x,2)$$

(3) Logique du deuxième ordre, logique floue, etc.

Intérêt: permettre un équilibre différent entre efficacité et expressivité

En général: plus le langage est expressif, plus il est difficile de raisonner dessus



Langage naturel: très expressif, mais très difficile de raisonner dessus (p.ex., ChatGPT)

Langage mathématique: sans faille, mais limité à des calculs arithmétiques

Composants d'un langage logique



Langage logique

Un langage logique est un langage défini par 3 éléments:

- (1) Une syntaxe, qui définit l'ensemble de **formules valides**
- (2) Une sémantique, qui définit la signification d'une formule
- (3) Des règles d'inférences, qui permettent d'obtenir des nouvelles formules valides

Syntaxe et sémantique: notions courantes dans n'importe quel langage

Règles d'inférences: spécifiques aux langages logiques



Formule

Entité correspondant à l'expression d'une connaissance



Base de connaissances

Aucun étudiant n'aime les examens

Simon aime les examens

↑
Formule en
langue naturelle

Base de connaissances: ensemble des formules connues de l'agent



Syntaxe

Elément d'un langage permettant de définir quelles sont les **formules valides**

Formule valide: formule correctement écrite au sens grammatical

Formule syntaxiquement correcte dans le langage mathématique: $x + y = 4$

Formule syntaxiquement incorrecte dans le langage mathématique: $x + y + = 4 -$

Composants d'un langage logique: sémantique



Sémantique

Elément d'un langage permettant de donner une signification aux formules

Exemple: $x = 3 + 2 \rightarrow$ signifie que x vaut 5

Attention: sans sémantique, les formules ne sont qu'une suite de caractères sans aucune signification !

Conseil: pour ce module, il est très important de bien faire la distinction entre syntaxe et sémantique



Tâche non intuitive: on a tendance à associer mentalement ces deux notions

"Je mange une pomme"

Très difficile de lire cette phrase sans associer le mot "*pomme*" au fruit

Exemples

Formule 1 : $3 + 2$

Formule 2 : $2 + 3$

Formule 1 : `return 3 / 2` (Python 2.7)

Formule 2 : `return 3 / 2` (Python 3)

Formule 1 : Je vais chez le dépanneur (Québécois)

Formule 2 : Je vais chez le dépanneur (Belge)

Syntaxe différente, même sémantique

Syntaxe identique, sémantique différente (1 et 1.5)

Syntaxe identique, sémantique différente

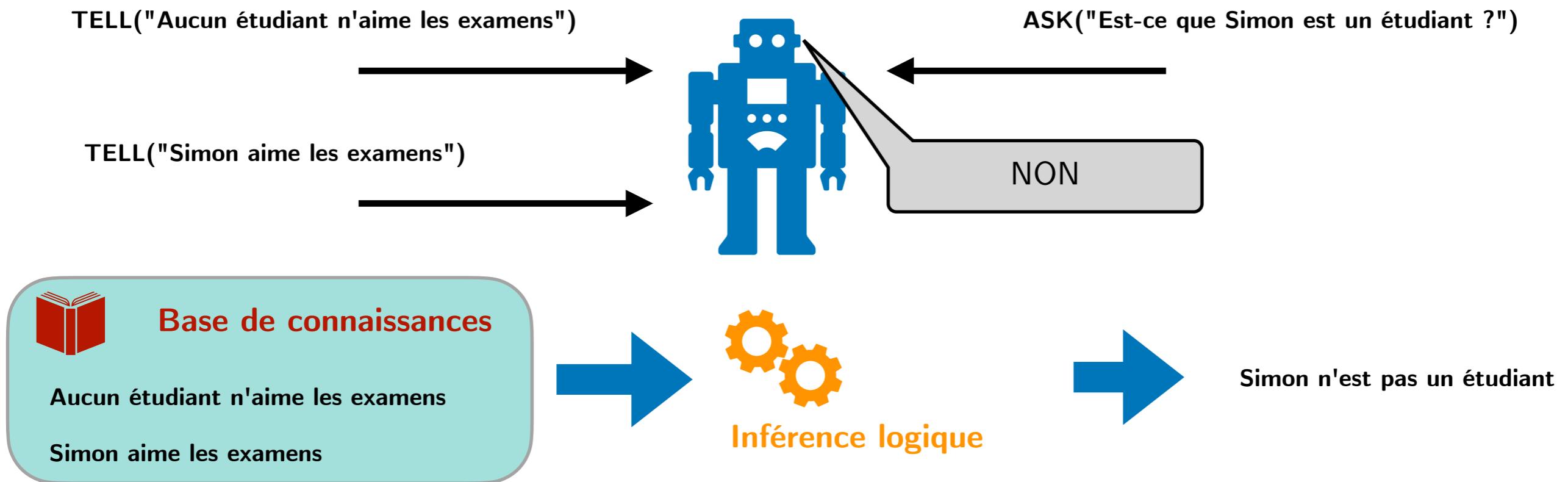
Composants d'un langage logique: règles d'inférence



Objectif: accroître les connaissances d'un agent

Il s'agit du cœur du raisonnement d'un agent logique

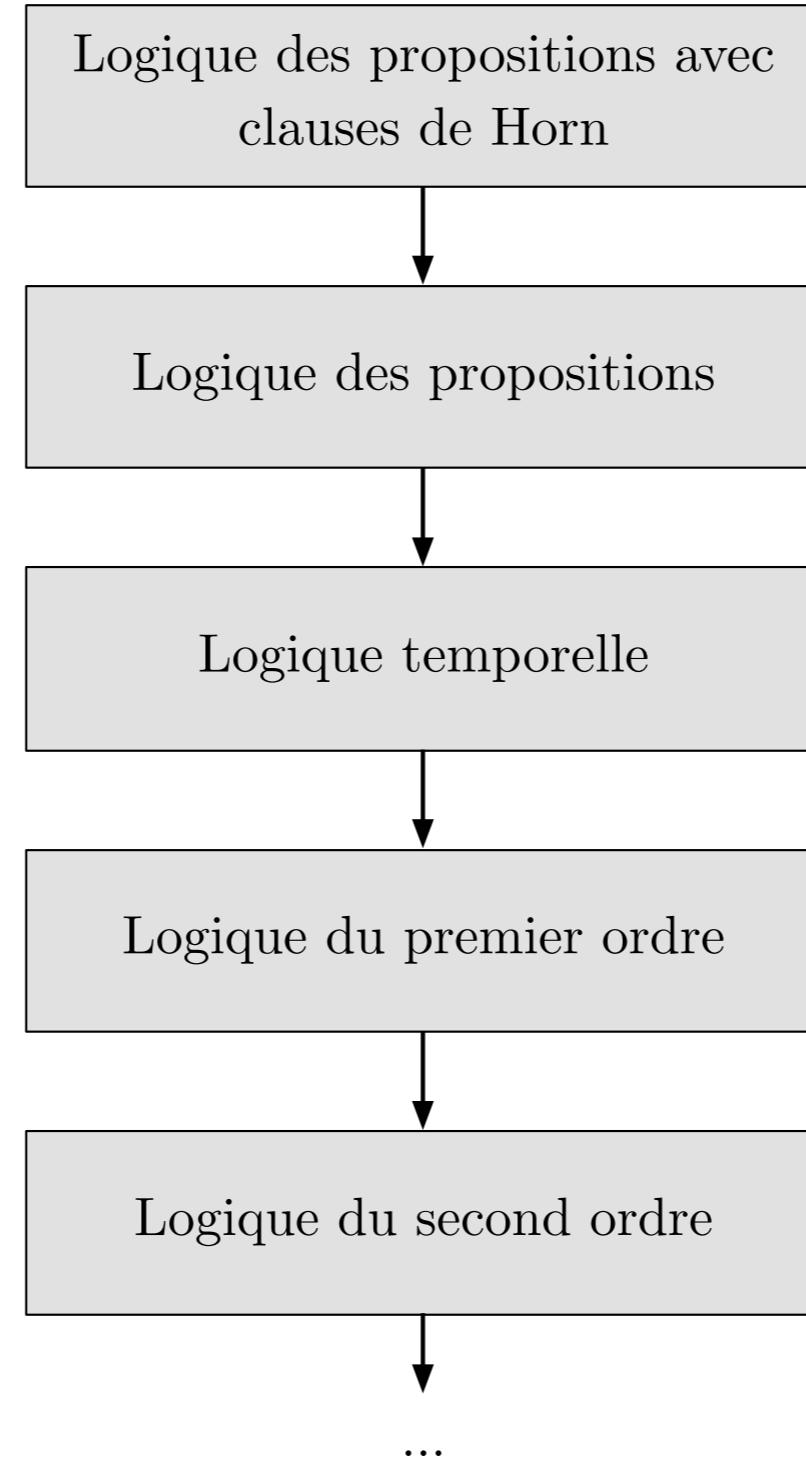
- (1) L'inférence se fait sur base des connaissances de l'agent (obtenue via TELL)
- (2) L'inférence est généralement exécutée lors d'une requête à l'agent (via ASK)



Mise à jour des connaissances: la nouvelle formule est ensuite ajoutée à la base de connaissances

Types de langages logiques

Gain en expressivité



Gain en performance



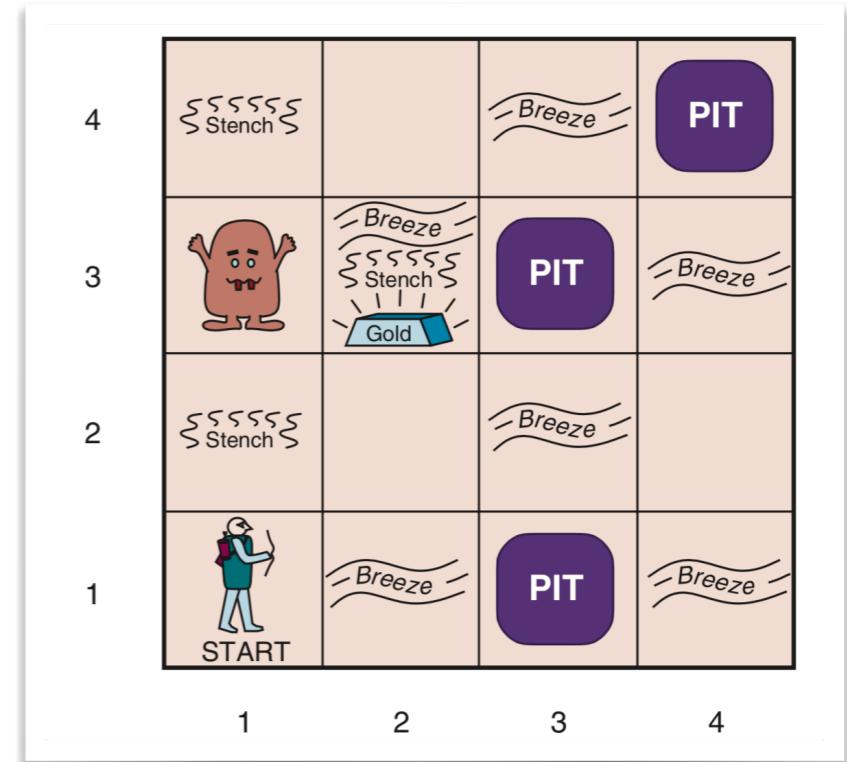
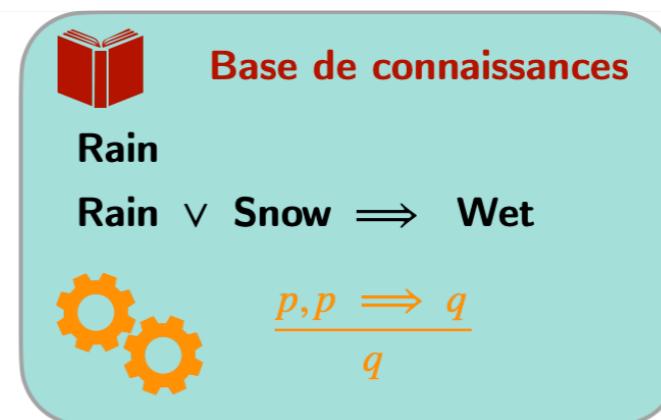
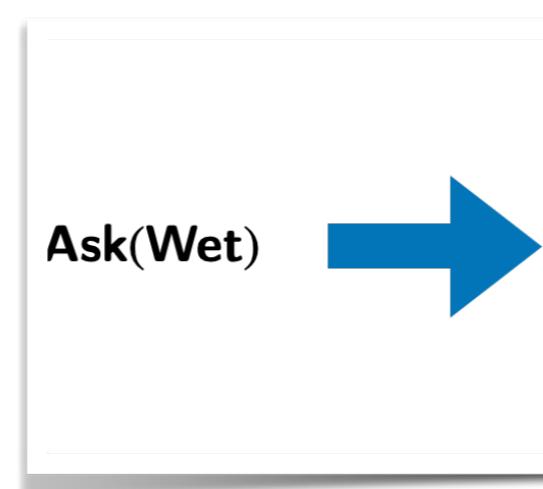
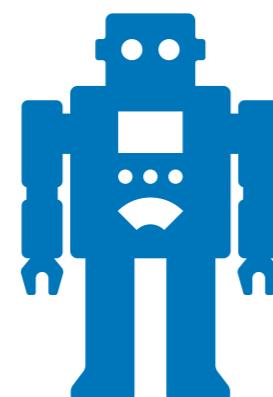
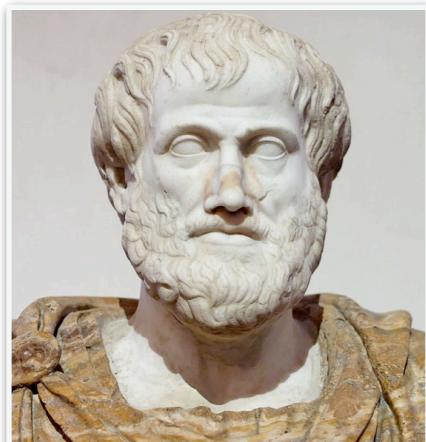
Expressivité: qualité des connaissances pouvant être intégrées dans la logique

Performance: qualité et efficacité des raisonnements logiques pouvant être faits

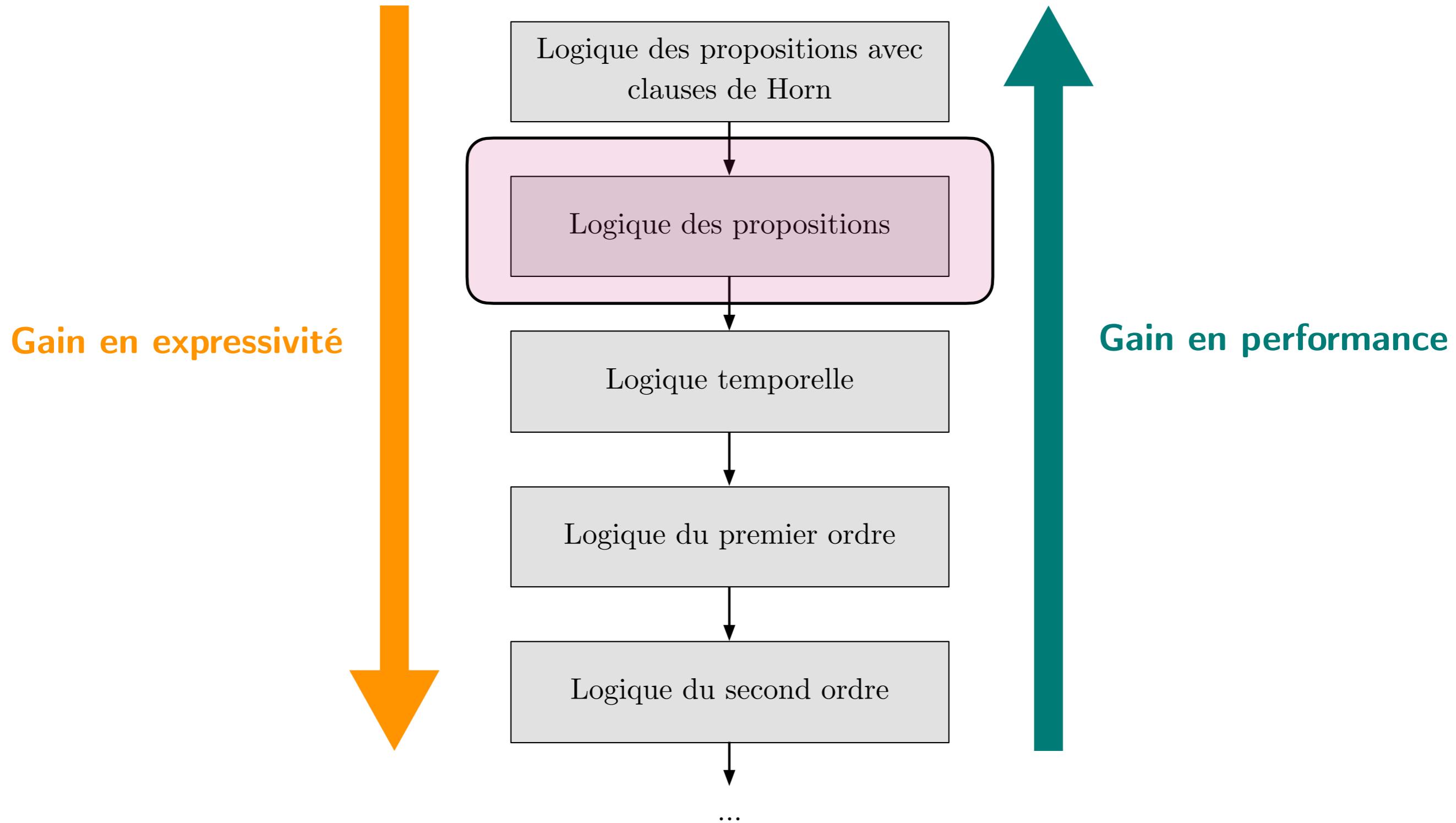
Table des matières

Agents logiques

- ✓ 1. Motivation et définition d'un agent logique
- ✓ 2. Représentation des connaissances
- ✓ 3. Système logique (syntaxe, sémantique, et règles d'inférences)
- 4. Formalisation de la logique des propositions
- 5. Raisonnement par model-checking
- 6. Inférences logiques et propriétés
- 7. Algorithme de résolution
- 8. Clauses de Horn
- 9. Extension à d'autres systèmes logiques



Logique des propositions



Logique des propositions: syntaxe

B

Element 1: formules atomiques (symboles)

Nature: formules constituées d'un seul symbole

Convention: les symboles commencent par une lettre majuscule

Exemples: A, B, Q, P, Raining, Wet, Dangerous, etc.

Deux symboles particuliers: True et False (mots-clé réservés)

V

Element 2: connecteurs logiques

Nature: 5 connecteurs sont disponibles dans la logique de propositions

Objectif: permettre d'assembler les formules pour en former des plus complexes

Liste des connecteurs: \neg , \wedge , \vee , \Rightarrow , \Leftrightarrow

$B \vee C$

Element 3: formules complexes

Nature: formules construites récursivement en utilisant les opérateurs logiques

Règles de composition: permet de créer une formule, à partir de deux formules f et g

Négation: $\neg f$

Conjonction: $f \wedge g$ (et)

Disjonction: $f \vee g$ (ou)

Implication: $f \Rightarrow g$ (également avec les symboles \rightarrow , \supset)

Équivalence: $f \Leftrightarrow g$ (si et seulement si, également avec le symbole \equiv)

Toutes les formules ne respectant pas ces règles ne sont pas syntaxiquement valides

Syntaxe: exemple

Négation: $\neg f$

Conjonction: $f \wedge g$

Disjunction: $f \vee g$

Implication: $f \implies g$

Équivalence: $f \iff g$

Priorité des opérateurs: donne un ordre de lecture aux opérateurs

Ordre de priorité : $\neg, \wedge, \vee, \implies, \iff$

Note: Il est permis d'ajouter des parenthèses pour établir les priorités



Est-ce que les formules suivantes sont syntaxiquement valides ?

P

Oui

Raining $\neg \wedge$ Wet

Non (composition non valide)

Raining \wedge Wet

Oui

$\neg \vee B$

Non (composition non valide)

$(P \wedge Q) \vee (\neg(R \implies \text{False}) \vee A)$

Oui

Raining $\wedge \neg \neg \neg$ Wet

Oui

Raining + Wet

Non (connecteur inconnu)

Attention: les formules et les opérateurs n'ont aucune signification ! Ce sont juste des symboles !

Un agent basé sur la logique des propositions ne peut ingérer que des formules respectant cette syntaxe

Logique des propositions: sémantique

L'objectif de la sémantique est d'associer à chaque formule valide une signification

Langage logique: la signification d'une formule passe par la notion de modèle et de fonction d'interprétation



Modèle (logique)

Dans le domaine de la logique, un modèle m est une assignation possible d'une valeur de vérité (**true (1)** ou **false (0)**) à chaque symbole présent dans un ensemble de formules

Exemple: considérons 3 symboles (Raining, Snowing, Dangerous)

Nombre de modèle: on a 8 modèles disponibles ($2 * 2 * 2$)

$m_1 = \{\text{Raining} : \text{true}, \text{Snowing} : \text{true}, \text{Dangerous} : \text{true}\}$

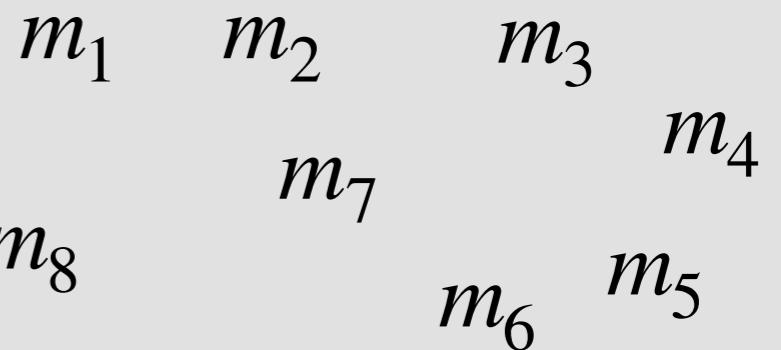
$m_2 = \{\text{Raining} : \text{false}, \text{Snowing} : \text{true}, \text{Dangerous} : \text{true}\}$

$m_3 = \{\text{Raining} : \text{true}, \text{Snowing} : \text{false}, \text{Dangerous} : \text{true}\}$

...

$m_8 = \{\text{Raining} : \text{false}, \text{Snowing} : \text{false}, \text{Dangerous} : \text{false}\}$

Ensemble des modèles



En général, on a 2^k modèles pour k symboles

Intuition: un modèle correspond à une réalisation possible du monde, caractérisé par la valeur des symboles

Attention: a ce stade, aucun lien avec les formules n'est établi

Point piégeux: aucune relation n'est encore faite entre un **symbole** (syntaxe) et son **sens** (sémantique)

Logique des propositions: sémantique



Fonction d'interprétation

Fonction qui associe la valeur de vérité d'une formule (f) selon un modèle spécifique (m)

$$I(f, m) : F \times M \rightarrow \{\text{false, true}\}$$

$$I(f, m) : F \times M \rightarrow \{0, 1\}$$

Objectif: donner une signification (un sens) à une formule, étant donné un modèle

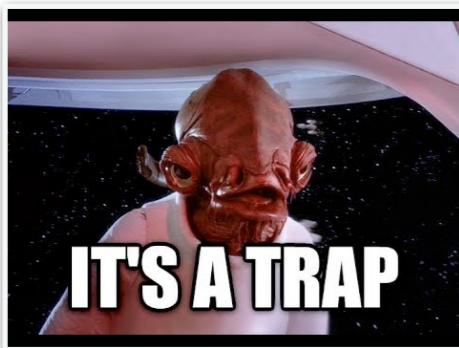
Valeur de vérité: autre nom donné à la signification donnée à une formule

La prochaine étape est de définir cette fonction pour la logique des propositions

Cas 1: formules atomiques

Règle 1: le symbole **True** (élément de syntaxe) a toujours la valeur de vérité **true** (sémantique)

Règle 2: le symbole **False** (élément de syntaxe) a toujours la valeur de vérité **false** (sémantique)



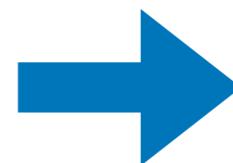
Attention: grosse source de confusion possible entre ces deux notions

Différence: le symbole commence par une majuscule, contrairement à la valeur de vérité

Notation: pour éviter les confusions, je vais remplacer les valeurs de vérité par 1/0

Règle 3: tout autre symbole a directement la valeur de vérité associée à celle définie dans le modèle

$$m = \{\text{Raining : true (1), Snowing : true (1), Dangerous : false (0)}\}$$



Raining : **true (1)**

Snowing : **true (1)**

Dangerous : **false (0)**

Logique des propositions: sémantique

Cas 2: formules complexes

Connecteurs logiques: chaque connecteur implique deux formules (sauf la négation qui en implique une)

Nombre de modèles: on a 4 modèles qui sont issus des 2 formules impliquées par le connecteur

Table de vérité: définition des valeurs de vérités d'une formule complexe, pour les 4 possibilités de modèle

P	Q
<i>false</i>	<i>false</i>
<i>false</i>	<i>true</i>
<i>true</i>	<i>false</i>
<i>true</i>	<i>true</i>

4 modèles possibles

Négation : $\neg P$ est vrai si et seulement si P est faux (et inversément)

Conjonction : $P \wedge Q$ est vrai si et seulement si P et Q sont tous les deux vrais

Disjonction : $P \vee Q$ est vrai si et seulement si au moins P ou Q est vrai

Implication : $P \implies Q$ est vrai sauf si P est vrai et Q est faux

Équivalence : $P \iff Q$ est vrai si P à la même valeur que Q



Sémantique: exemple

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

? Quelle est la valeur sémantique de cette formule étant donné ces modèles ? $f = (\neg P \wedge Q) \Leftrightarrow R$

Cas 1: $m_1 = \{P : 1, Q : 1, R : 0\}$

$P = 1$

Résultat: la formule f est vraie sous m_1

$Q = 1$

Cas 2: $m_2 = \{P : 0, Q : 1, R : 0\}$

$R = 0$

Résultat: la formule f est fausse sous m_2

$P = 0$

$Q = 1$

On a un mécanisme donnant une signification à chacune des formules valides en logique des propositions

Modèles d'une formule



Modèles d'une formule

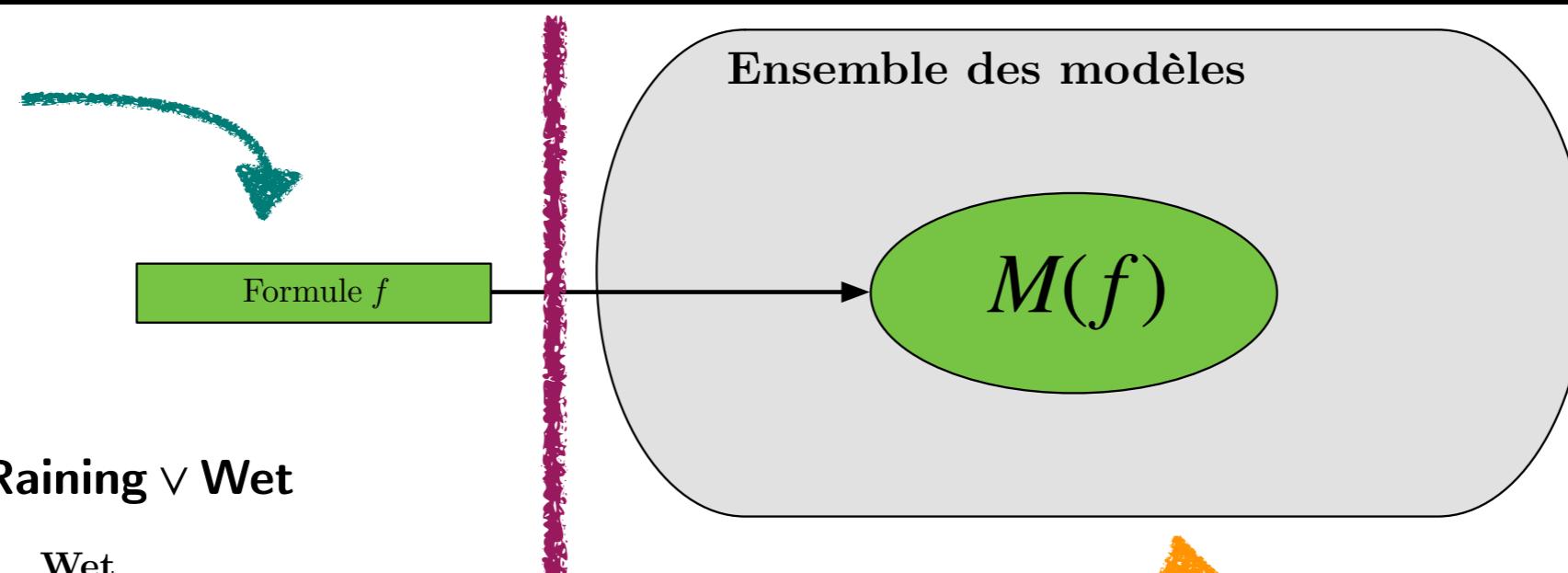
Un modèle m est un modèle pour une formule f , si f est vrai sous l'interprétation de m

$I(f, m) = 1$ signifie que m est un modèle pour f

On définit également $M(f)$, l'ensemble des modèles de la formule f

$M(f) = \text{tous les modèles } m \text{ tels que } I(f, m) = 1$

Syntaxe



Exemple: $f = \text{Raining} \vee \text{Wet}$

$M(f) =$

		Wet
		0 1
Raining	0	0
	1	1

La formule a 3 modèles parmi les 4 modèles existants

Intuition: on peut voir une formule comme une représentation compacte d'un ensemble de modèles

Note: ce n'est que réellement maintenant qu'on fait le lien entre sémantique et syntaxe

Base de connaissances (*knowledge base*)



Base de connaissances (*knowledge base - KB*)

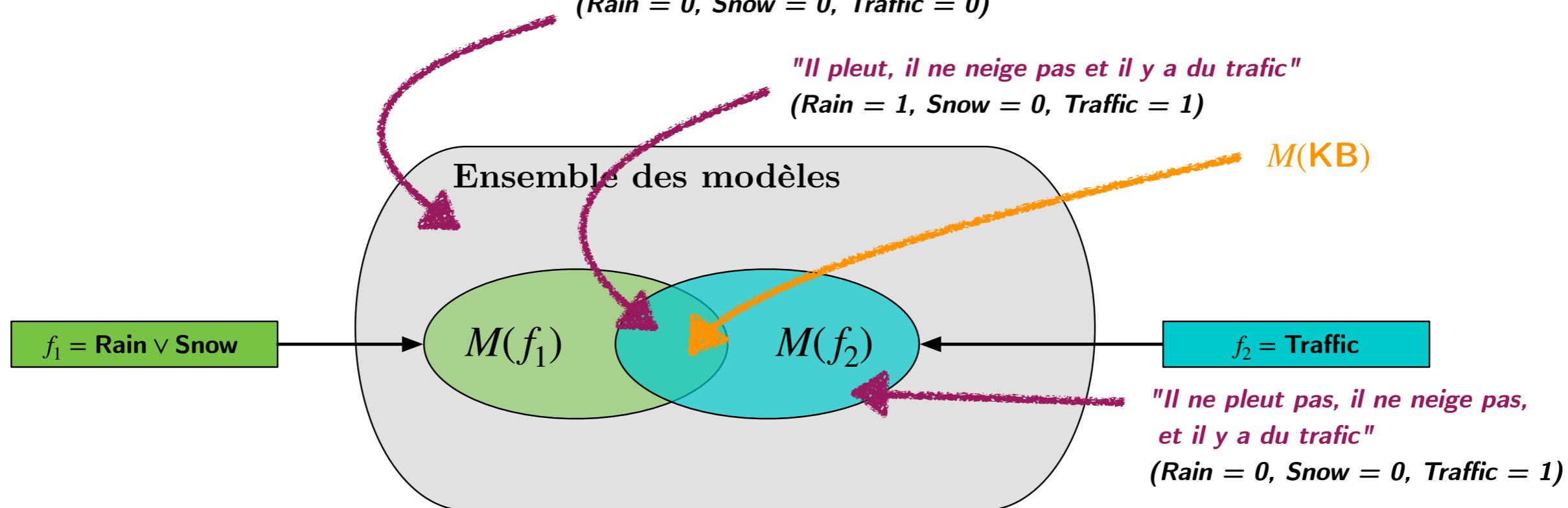
Ensemble de formules prises de façon conjointe. Les modèles de la base de connaissances correspondent à l'intersection des modèles de toutes les formules qui la composent

$$M(KB) = \bigcap_{f \in KB} M(f)$$

Exemple: $KB = \{Rain \vee Snow, Traffic\}$

"Il ne pleut pas, il ne neige pas, et il n'y a pas de trafic"
($Rain = 0, Snow = 0, Traffic = 0$)

"Il pleut, il ne neige pas et il y a du trafic"
($Rain = 1, Snow = 0, Traffic = 1$)



Intuition: la base de connaissances indique un ensemble de contraintes sur le monde (ce qu'on connaît)

$M(KB)$ donne ainsi l'ensemble des réalisations possibles selon les contraintes (nos connaissances)

Base de connaissances (*knowledge base*)

Détaillons les modèles d'une base de connaissances à travers un exemple

Base de connaissances: $KB = \{Rain, Rain \implies Wet\}$

Etape 1: détailler le modèle de chaque connaissance

$$M(Rain) =$$

		Wet				
		0 1				
Rain	0	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>				
1		<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>				

"Il pleut"

$$M(Rain \implies Wet) =$$

		Wet				
		0 1				
Rain	0	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>				
1		<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>				

"Lorsqu'il pleut, le sol est mouillé"



Etape 2: prendre l'intersection pour avoir le modèle de la base de connaissances

$$M(KB) =$$

		Wet				
		0 1				
Rain	0	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>				
1		<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>				

"Il pleut et le sol est mouillé"

		Wet				
		0 1				
Rain	0	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>				
1		<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>				

\cap

		Wet				
		0 1				
Rain	0	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>				
1		<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>				

Enrichir la base de connaissances

Un des intérêts d'une base de connaissances est de pouvoir y ajouter des nouvelles formules

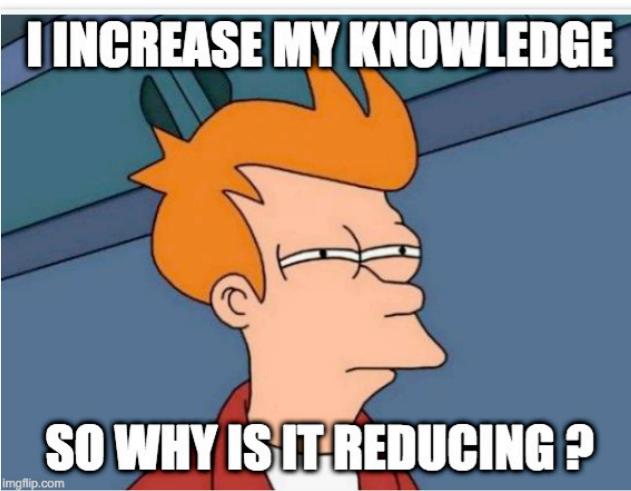
$$\text{KB} \rightarrow \text{KB} \cup \{f\}$$

? Quel est l'impact sur les modèles de la base de connaissances ?

$$M(\text{KB}) \rightarrow M(\text{KB}) \cap M(f)$$

Mise à jour: l'intersection des deux ensembles donnent les nouveaux modèles de la base de connaissances

Conséquence: l'intersection va réduire le nombre de modèles



Attention: cette réduction peut paraître contre-intuitive !

Observation: acquérir des connaissances réduit (ou garde) le nombre de modèles

Interprétation: on a une vision plus précise du monde (moins de possibilités)

$\text{KB} = \{\text{Rain} \implies \text{Wet}\}$ "Je sais que lorsqu'il pleut, le sol est mouillé"

$\text{KB} = \text{KB} \cup \{\text{Rain}\}$ "Je sais maintenant qu'il pleut"

"Dès lors, je sais maintenant que le sol est mouillé"

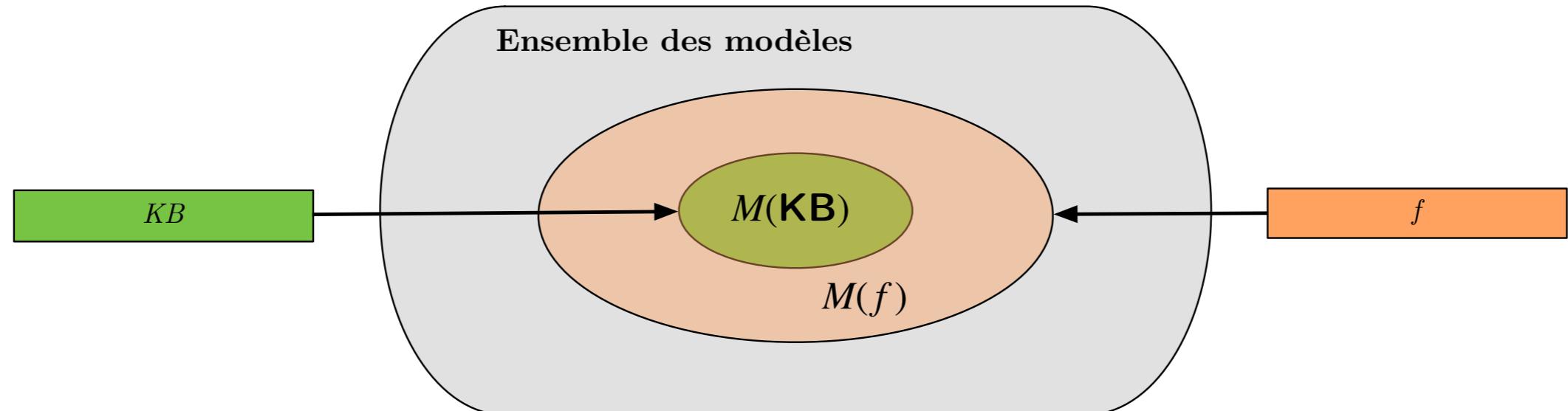
		Wet
		0 1
Rain	0	
	1	

? Quel est l'impact d'ajouter une formule à la base de connaissance ?

On a 3 situations qui peuvent se produire

Cas 1: Conséquence logique (*entailment*)

Cas 1: situation où tous les modèles de KB sont inclus dans les modèles de la nouvelle formule



Intuition: la formule f n'ajoute aucune information à la base de connaissance

Explication: toutes les connaissances de f étaient déjà contenues dans la base de connaissances



Conséquence logique (*entailment*)

- Soit KB et une formule f , f est une conséquence logique de KB si et seulement si f est vrai pour chaque modèle où KB est vrai

$$KB \models f \text{ si et seulement si } M(KB) \subseteq M(f)$$

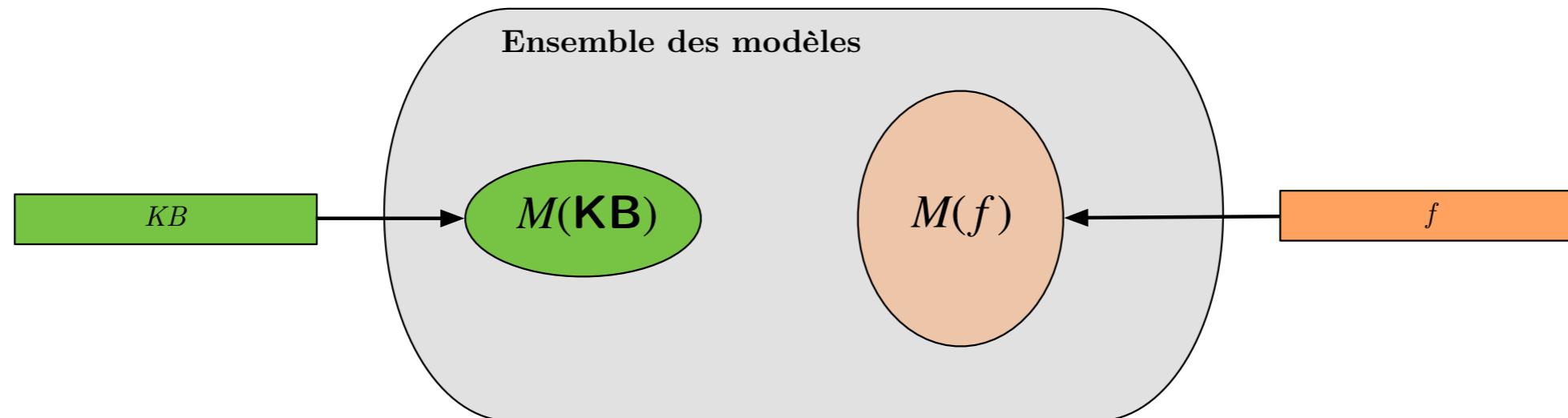
Terminologie: on dit également que KB implique logiquement (*entails*) la formule f

Utilité: permet de vérifier si une formule f est toujours vraie si la KB est vraie

Exemple: $\{\text{Raining} \wedge \text{Wet}\} \models \{\text{Wet}\}$ *S'il pleut et que le sol est mouillé... alors, il pleut également*

Cas 2: Contradiction

Cas 2: situation où les modèles de KB sont tous différents des modèles de f



Intuition: la formule f ajoute une information contradictoire avec la base de connaissance

Explication: la nouvelle connaissance obtenue est incompatible avec ce qu'on connaît déjà



Contradiction

- Soit KB et une formule f , f est en contradiction avec KB s'ils n'ont aucun modèle en commun
- KB est en contradiction avec f si et seulement si** $M(KB) \cap M(f) = \emptyset$

Utilité: le principe de contradiction est très fort utilisé pour des raisonnements logiques par l'absurde

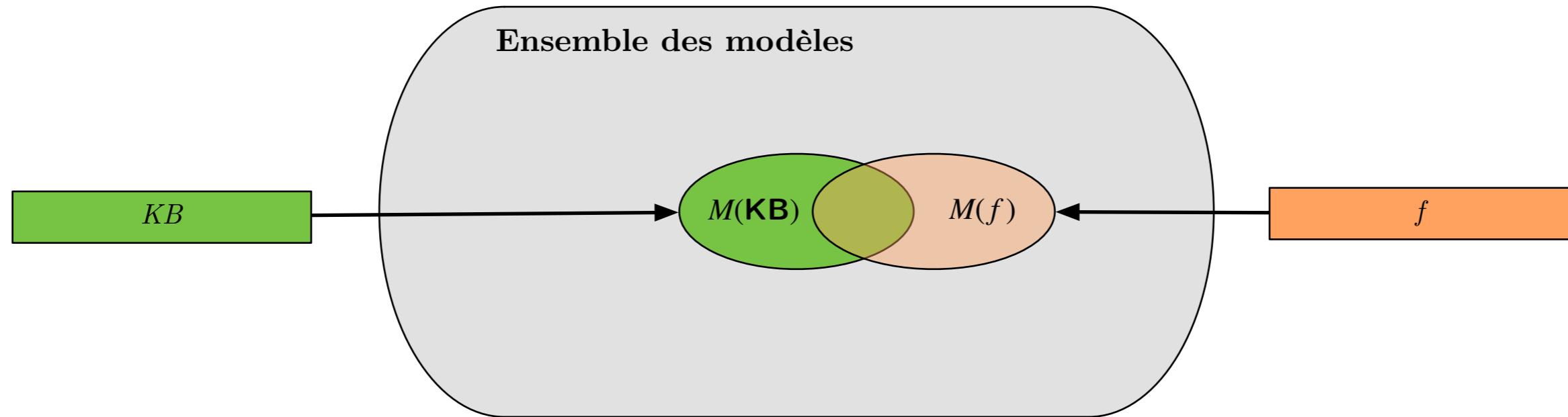
Intuition: si on est dans un monde où KB est vrai, alors il est impossible de satisfaire f

Exemple: {Raining \wedge Wet} est en contradiction avec { \neg Raining}

*Il pleut et le sol est mouillé...
et il ne pleut pas... : contradiction*

Cas 3: Contingence

Cas 3: situation où le nombre de modèles de KB est réduit, sans qu'on ait une contradiction



Intuition: la formule f est une nouvelle information, non contradictoire, à la base de connaissance

Explication: une nouvelle connaissance non triviale a été ajoutée à la base de connaissance

 **Contingence**

Soit KB et une formule f , f est en contingence avec KB si et seulement si elle n'est ni une conséquence logique de KB , ni en contradiction

$$\emptyset \subset M(KB) \cap M(f) \subset M(KB)$$

Utilité: une contingence permet de raffiner les connaissances que l'agent a à disposition

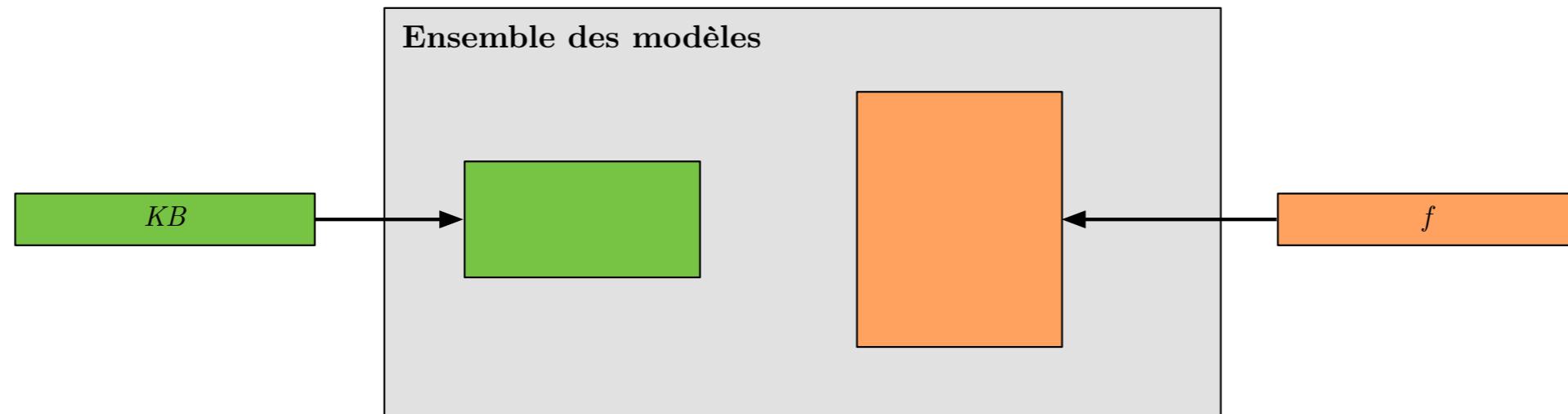
Exemple: {Raining} est en contingence avec {Wet}

*Je sais qu'il pleut...
j'ai appris également que le sol est mouillé*

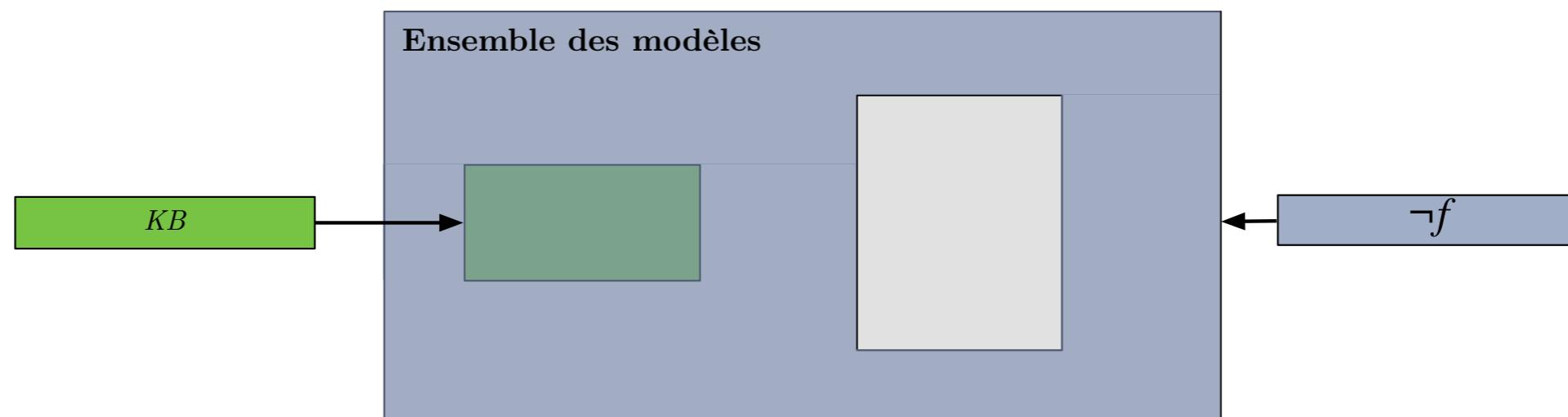
Relation entre conséquence logique et contradiction

Il existe une relation intéressante entre conséquence logique et contradiction

(1) Point de vue d'une contradiction: supposons qu'on soit dans une situation où f contredit KB



(2) Point de vue d'une conséquence logique: on considère la négation de la formule



Que peut-on en conclure ?

KB contredit f si et seulement si $KB \models \neg f$

Ce résultat va servir de base à plusieurs raisonnements logiques par la suite

Lien avec l'opération TELL



Opération TELL

Opération consistant à donner une nouvelle formule f à la base de connaissances

Fonction: enrichir les connaissances de l'agent

Exemple: Je veux intégrer la connaissance "*Il pleut*" à l'agent

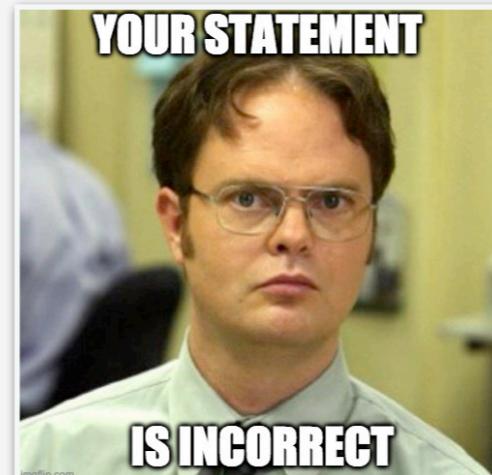


Quelles sont les réactions possibles de la part de KB ?



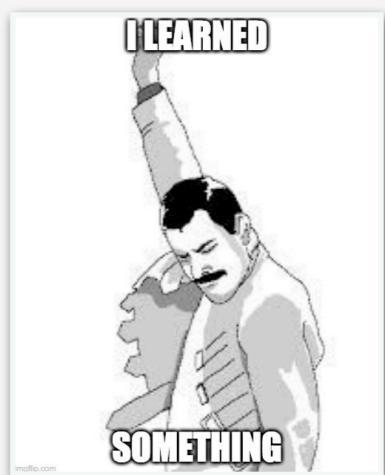
f est une conséquence logique
de KB

$$KB \models f$$



f est en contradiction
avec KB

$$KB \models \neg f$$



f est en contingence
avec KB

KB est mis à jour

Lien avec l'opération ASK



Opération consistant à demander à la KB si une formule est vraie ou fausse

Fonction: obtenir de l'information venant de l'agent

Exemple: Je veux savoir si il pleut



Quelles sont les réactions possibles de la part de KB ?

f est une conséquence logique
de KB

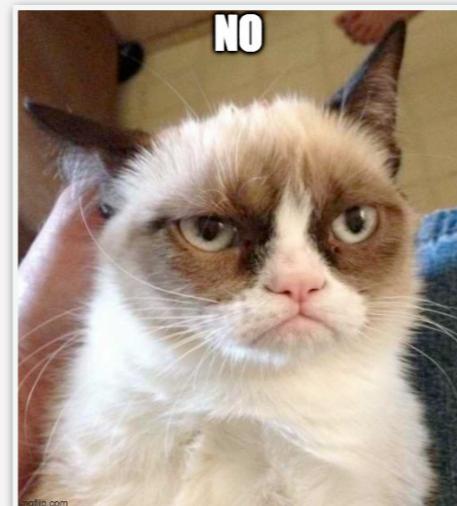
$$\text{KB} \models f$$



La réponse est OUI

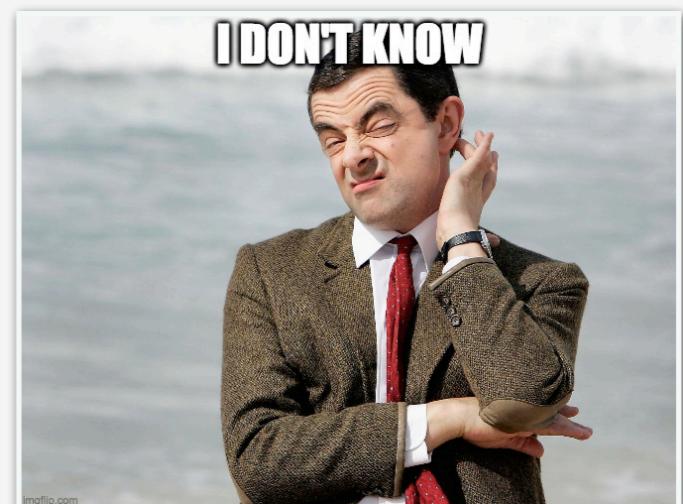
f est en contradiction
avec KB

$$\text{KB} \models \neg f$$



La réponse est NON

f est en contingence avec KB



L'agent ne connaît pas la
réponse

Satisfiabilité d'une base de connaissance



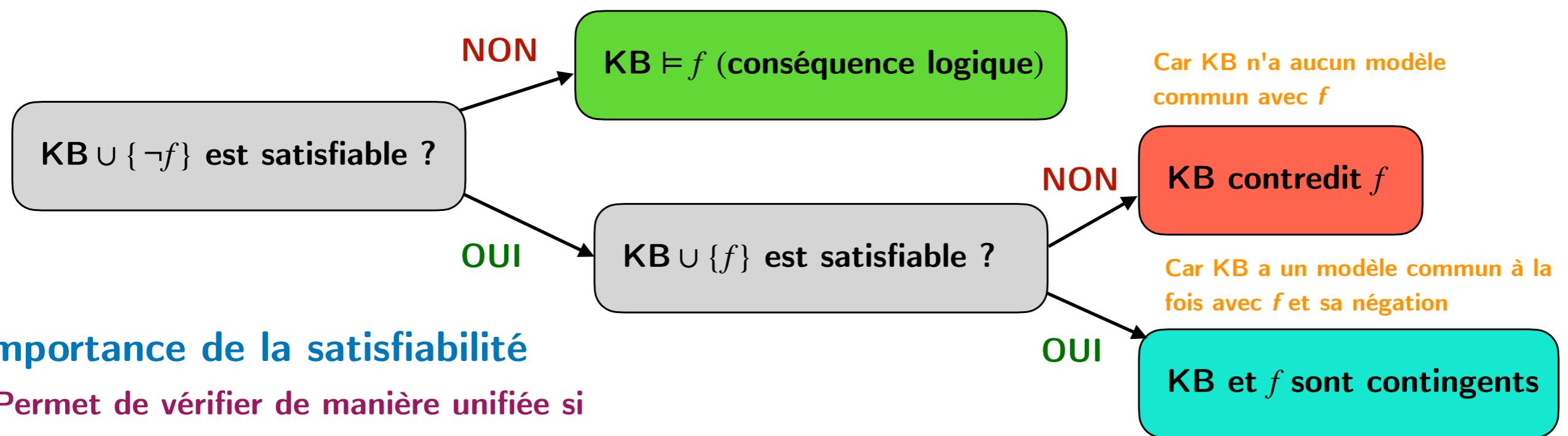
Satisfiabilité

Une base de connaissance KB est satisfiable si elle contient au moins un modèle
 $M(KB) \neq \emptyset$

Illustration: supposons une base de connaissances KB et une formule f

Principe: on effectue deux requêtes impliquant f à KB

Car la négation de f n'a aucun modèle commun avec KB



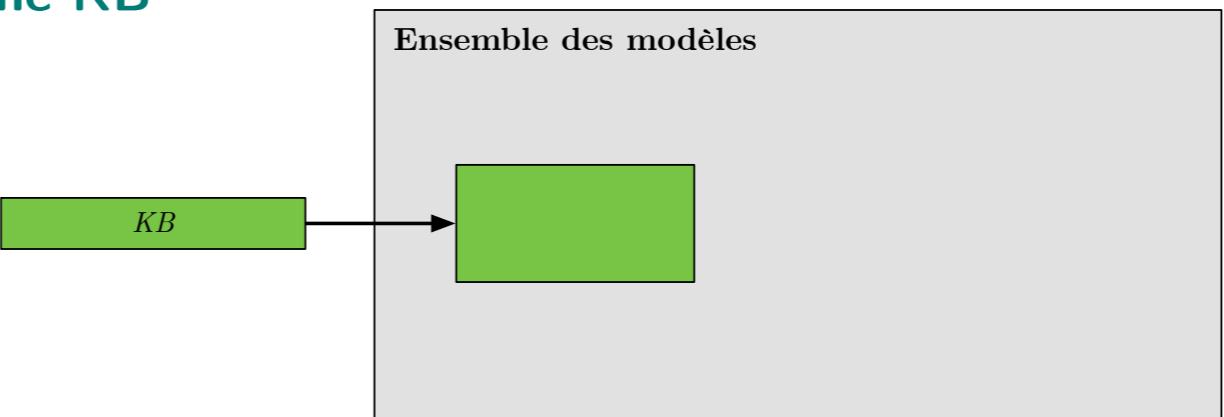
Importance de la satisfiabilité

Permet de vérifier de manière unifiée si

- (1) Une formule est une conséquence logique d'une KB
- (2) Une formule est en contradiction avec KB
- (3) Une formule est en contingence avec KB

Applicable à la fois pour les opérations TELL et ASK

Tâche importante: pouvoir calculer la satisfiabilité



Vérification de satisfiabilité



Comment vérifier la satisfiabilité d'une KB ?



Base de connaissances

Raining \implies Wet

Wet \wedge Snowing \implies Dangerous

Snowing

Raining

Requête ASK: Dangerous?



Base de connaissances

$P \implies Q$

$L \wedge M \implies P$

$B \wedge L \implies M$

$A \wedge P \implies L$

$A \wedge B \implies L$

A

B

Requête ASK: Q?

Model checking: méthode pour vérifier la satisfiabilité d'une KB

Satisfiable: Il s'agit de trouver un moins un modèle

Non-satisfiable: il s'agit de prouver qu'il n'en existe aucun

Mauvaise nouvelle: il s'agit d'un problème NP-complet

Bonne nouvelle: on a vu des algorithmes pour résoudre cela !

Principe: on peut modéliser ce problème comme un CSP

Symboles: variables de décision à domaine binaire

Formules: contraintes

Modèle: assignation complète des variables

Algorithme DPLL: résolution par *backtracking search* et *pruning*

WalkSat: résolution par recherche locale intégrant de l'aléatoire

Vous pouvez même faire un modèle Minizinc !

Description de ces algorithmes en lectures complémentaires

Exemple: utilisation de la satisfiabilité



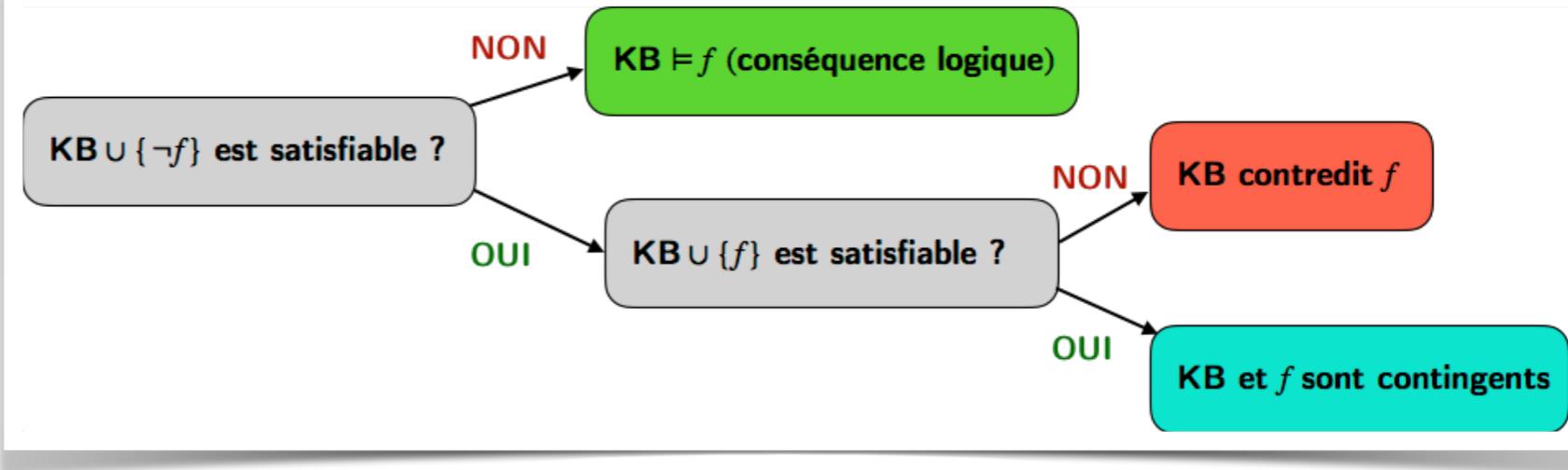
Base de connaissances

Raining \implies Wet

Wet \wedge Snowing \implies Dangerous

Snowing

Raining



Requête: Dangerous?

Question 1: est ce que $KB \cup \{\neg \text{Dangerous}\}$ est satisfiable ?

Résolution: exécution du CSP pour obtenir la réponse

```
Hide all

▼ Running model_checking_1.mzn
Warning: model inconsistency detected
=====UNSATISFIABLE=====
Finished in 47msec.
```

Réponse: NON

```
1 % Variables
2 var bool: Raining;
3 var bool: Wet;
4 var bool: Dangerous;
5 var bool: Snowing;
6
7 % Base de connaissances
8
9 constraint Raining -> Wet;
10 constraint (Wet /\ Snowing) -> Dangerous;
11 constraint Snowing;
12 constraint Raining;
13
14 % Requête
15
16 constraint not Dangerous;
17
18 solve satisfy;
```

Résultat: on sait que Dangerous est une conséquence logique de KB

Exemple: utilisation de la satisfiabilité



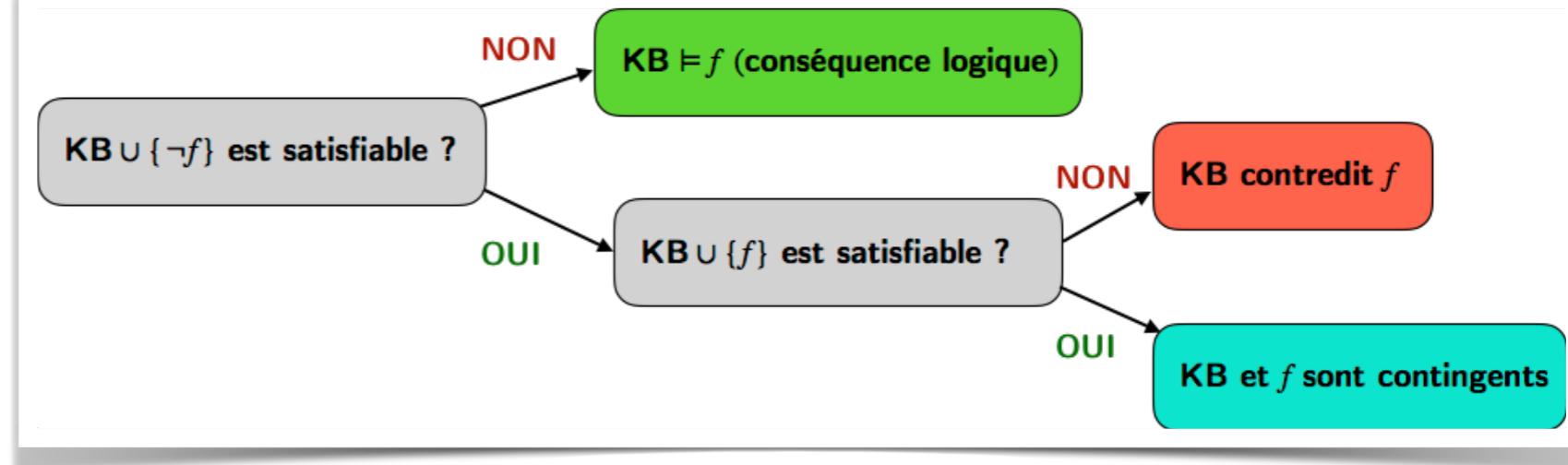
Base de connaissances

$$P \implies \neg Q$$

$$L \wedge M \implies P \quad A \wedge P \implies L$$

$$B \wedge L \implies M \quad A \wedge B \implies L$$

$$A \quad B$$



Requête: Q?

Question 1: est ce que KB ∪ {¬Q} est satisfiable ?

```
-----  
Finished in 53msec.  
Running model_checking_2.mzn  
P = true;  
Q = false;  
L = true;  
M = true;  
B = true;  
A = true;  
-----  
Finished in 54msec.
```

OUI

Question 2: est ce que KB ∪ {Q} est satisfiable ?

```
-----  
Finished in 54msec.  
Running model_checking_2.mzn  
Warning: model inconsistency detected  
=====UNSATISFIABLE=====  
Finished in 51msec.
```

NON

```
1 % Variables  
2 var bool: P;  
3 var bool: Q;  
4 var bool: L;  
5 var bool: M;  
6 var bool: B;  
7 var bool: A;  
8  
9 % Base de connaissances  
10  
11 constraint P -> not Q;  
12 constraint (L /\ M) -> P;  
13 constraint (B /\ L) -> M;  
14 constraint (A /\ P) -> L;  
15 constraint (A /\ B) -> L;  
16 constraint A;  
17 constraint B;  
18  
19 % Requête  
20  
21 constraint not Q;  
22  
23 solve satisfy;
```

```
1 % Variables  
2 var bool: P;  
3 var bool: Q;  
4 var bool: L;  
5 var bool: M;  
6 var bool: B;  
7 var bool: A;  
8  
9 % Base de connaissances  
10  
11 constraint P -> not Q;  
12 constraint (L /\ M) -> P;  
13 constraint (B /\ L) -> M;  
14 constraint (A /\ P) -> L;  
15 constraint (A /\ B) -> L;  
16 constraint A;  
17 constraint B;  
18  
19 % Requête  
20 |  
21 constraint Q;  
22  
23 solve satisfy;
```

Résultat: on sait que Q est en contradiction avec KB

Table des matières

Agents logiques

✓ 1. Motivation et définition d'un agent logique

✓ 2. Représentation des connaissances

✓ 3. Système logique (syntaxe, sémantique, et règles d'inférences)

✓ 4. Formalisation de la logique des propositions

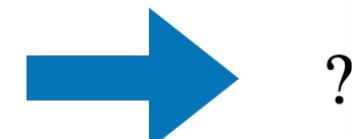
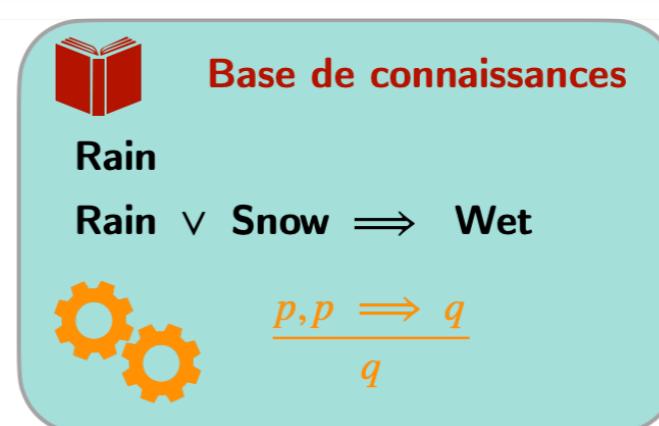
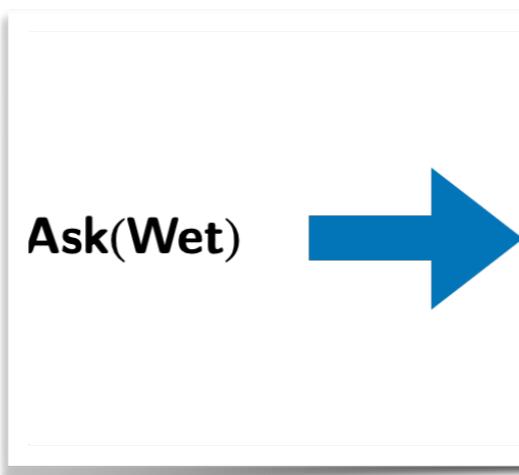
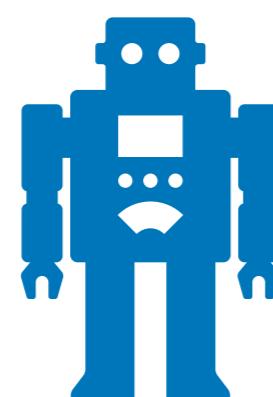
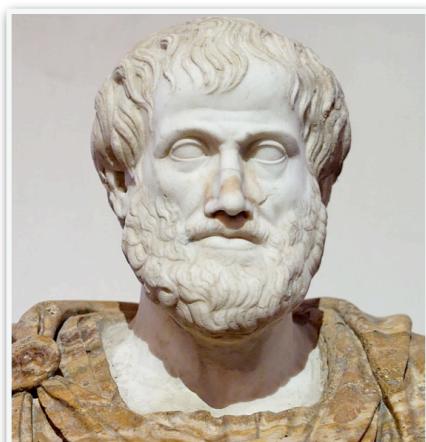
✓ 5. Raisonnement par model-checking

6. Inférences logiques et propriétés

7. Algorithme de résolution

8. Clauses de Horn

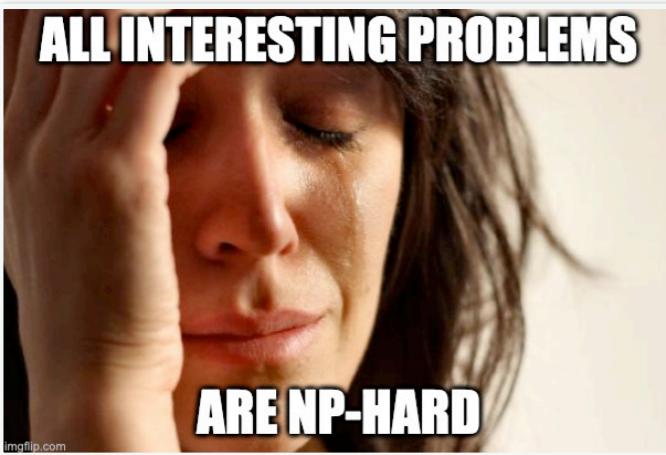
9. Extension à d'autres systèmes logiques



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

Complexité de la vérification de satisfiabilité en logique des proposition

Malheureusement, trouver un modèle qui satisfait les formules d'une KB est NP-complet (problème SAT)



Historique: SAT est le premier problème prouvé NP-complet (Cook, 1971)

Intuition: on a un nombre exponentiel de modèles à tester

Implication: on utilise des algorithmes de recherche pour sa résolution

?

Peut-on faire mieux ?

Certitude: on aura toujours une complexité exponentielle dans le pire des cas pour une résolution exacte



Résolution actuelle: basée sur une stratégie de recherche

Particularité du problème: on a affaire à des formules logiques

Formules logiques: forme spécifique de contraintes

?

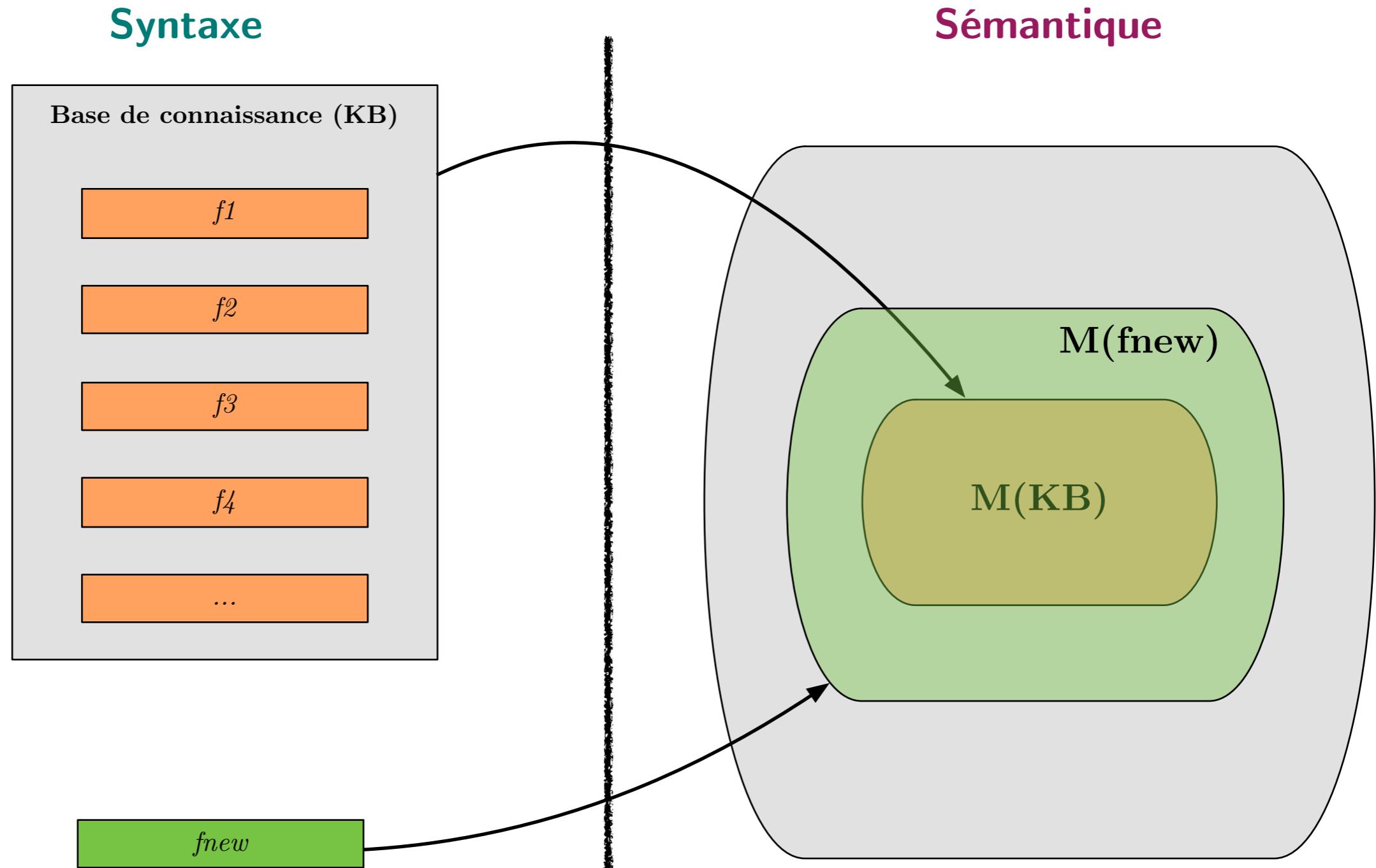
Peut-on exploiter cela dans notre processus de résolution ?

Oui: on peut utiliser un autre type de processus de résolution, à savoir l'inférence logique

Bonne nouvelle: permet d'obtenir de meilleures performances en pratique

Regardons comment exploiter au mieux le fait qu'on travaille avec des formules logiques

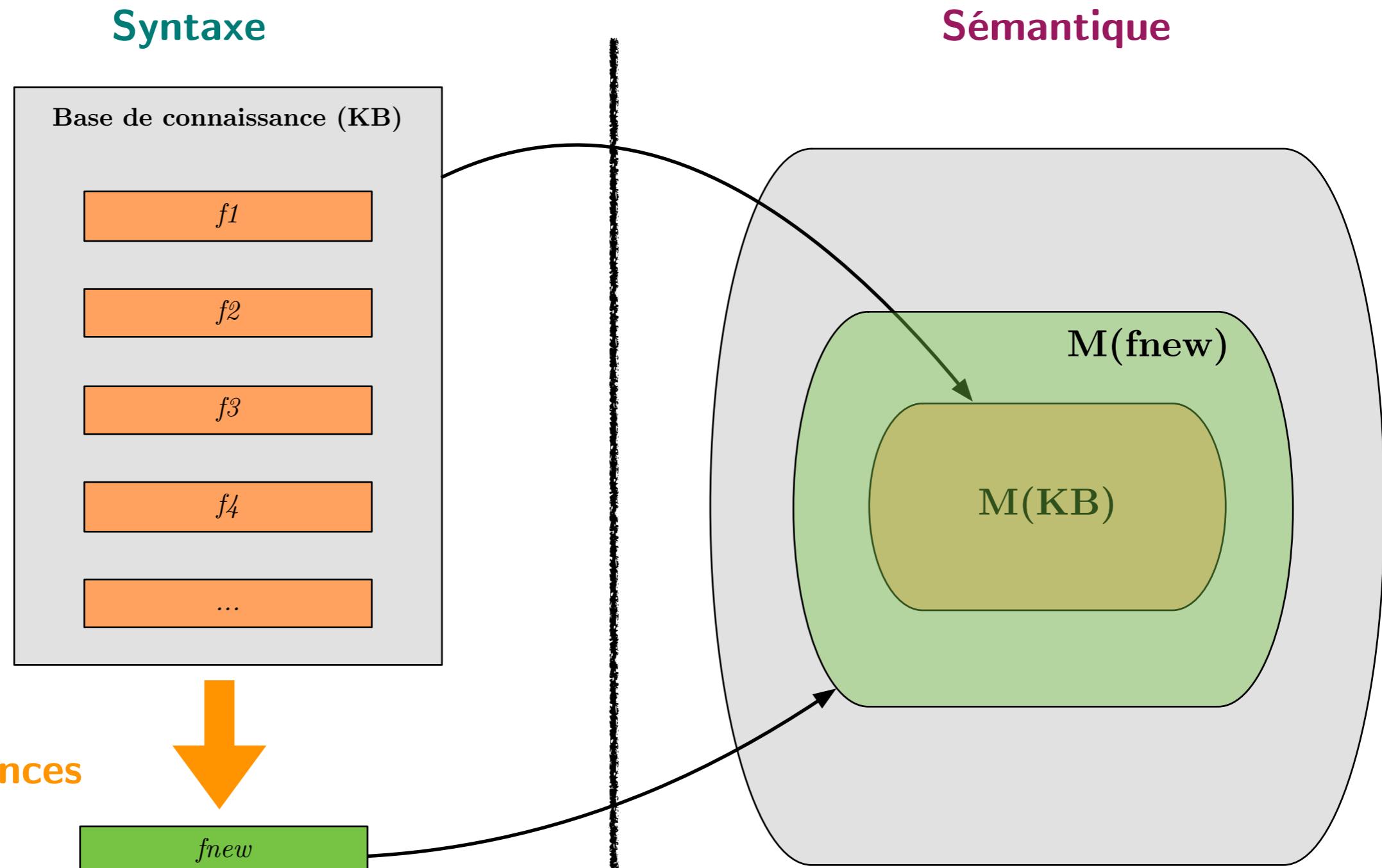
Schéma de la situation actuelle



Résultat: si les les modèles d'une formule englobent les modèles de KB, alors elle en conséquence logique

Model checking: raisonnement possible pour vérifier cela **en restant dans le monde de la sémantique**

Règles d'inférence



Peut-on obtenir un résultat similaire en restant dans le monde de la syntaxe ?

Oui: en utilisant des règles d'inférence (troisième élément central d'un système logique)

Règles d'inférence

Une inférence logique est un processus par lequel on obtient une nouvelle formule via des formules connues

Intuition: une règle d'inférence est une règle qui définit la façon dont une nouvelle règle peut être générée



Règle d'inférence

Soit un ensemble de formules f_1, f_2, \dots, f_k et g . Une règle d'inférence à la forme suivante:

$$\frac{f_1, \dots, f_k}{g}$$

Elle indique que la formule g peut être générée à partir des formules f_1, \dots, f_k

Prémices: formules déjà connues

Conclusion: formule générée en appliquant la règle d'inférence

prémices
conclusion

Il existe un très grand nombre de règles d'inférence (certaines étant plus utiles que d'autres)

(1) Modus Ponens

(2) Règle d'élimination

(3) Règle de résolution

Ces règles peuvent être utilisées dans la conception d'algorithmes pour obtenir des conséquences logiques

Règle d'inférence: Modus ponens et règle d'élimination

Modus Ponens

Règle d'inférence sous la forme suivante:

$$\frac{p, p \Rightarrow q}{q}$$

avec p et q deux formules

Règles d'élimination

Règles d'inférence sous la forme suivante:

$$\frac{p \wedge q}{q}$$

$$\frac{p \wedge q}{p}$$

avec p et q deux formules

Rain, Rain \implies Wet

Exemple:

Wet

Prémisse 1: il pleut

Prémisse 2: quand il pleut, le sol est mouillé

Conclusion: le sol est mouillé (Wet)

Rain \wedge Snow

Exemple:

Rain

Prémisse 1: Il pleut et il neige

Conclusion: il pleut

Fonctionnement: on ne fait qu'appliquer des règles opérant sur des formules

Attention: les règles d'inférences appartiennent au monde de la syntaxe

Conséquence: à ce stade, elles n'ont aucune relation avec la sémantique

Autrement dit, elles n'ont aucune interprétation !

IT'S A TRAP

Algorithme générique d'inférences logiques

logicalInference(KB, R) :

while KB is modified :

$f_1, \dots, f_k = \text{selectFormulaFrom}(KB)$

if a matching rule $\frac{f_1, \dots, f_k}{g}$ exists in R :

$KB = KB \cup g$

return KB

Objectif: on souhaite appliquer un ensemble de règles d'inférence (R) à KB

Tant qu'on peut rajouter une formule à KB

On sélectionne un ensemble de formules de KB

Si une règle d'inférence existe sur les règles sélectionnées

On ajoute la formule inférée (g) à KB

Attention: cet algorithme est encore incomplet pour une utilisation pratique

(1) Quelles règles d'inférences utiliser ?

(2) Comment sélectionner les formules pertinentes (potentiellement des millions dans une KB)



Dérivation de formules

S'il est possible d'obtenir une formule f par application d'un algorithme d'inférence A sur une base de connaissance KB , on dit que f est dérivé de KB par A

$KB \vdash_A f$

Attention: on est toujours dans le monde de la syntaxe, sans relation faite avec celui de la sémantique

A ce stade, cet algorithme ne fait que dériver mécaniquement de nouvelles formules

Exemple de dérivation de formules



Base de connaissances

Rain

Rain \implies Wet

Wet \implies Slippery



Base de connaissances

Rain

Wet

Rain \implies Wet

Wet \implies Slippery



Base de connaissances

Rain

Wet

Rain \implies Wet

Slippery

Wet \implies Slippery

Example: algorithme (A) utilisant la règle: $\left\{ \frac{p, p \implies q}{q} \right\}$ (mod ponens)



Quelles nouvelles formules peut-on dériver ?

Dérivation 1: première application du modus ponens

$$\frac{\text{Rain}, \text{Rain} \implies \text{Wet}}{\text{Wet}}$$

$KB \vdash_A \text{Wet}$

Dérivation 2: deuxième application du modus ponens

$$\frac{\text{Wet}, \text{Wet} \implies \text{Slippery}}{\text{Slippery}}$$

$KB \vdash_A \text{Slippery}$

Convergence: on ne sait plus rien ajouter

En pratique: on souhaite générer des règles qui ont un sens sémantique correct



Est-ce que les règles qu'on génère ont du sens ?

Lien avec la sémantique

$KB \vdash f_{new}$

$KB \models f_{new}$

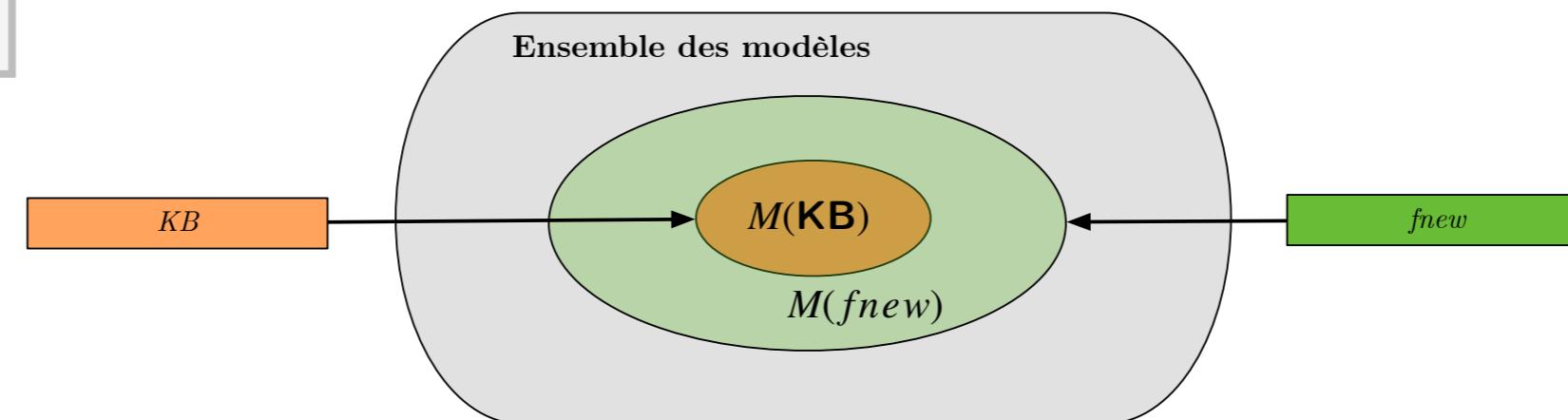
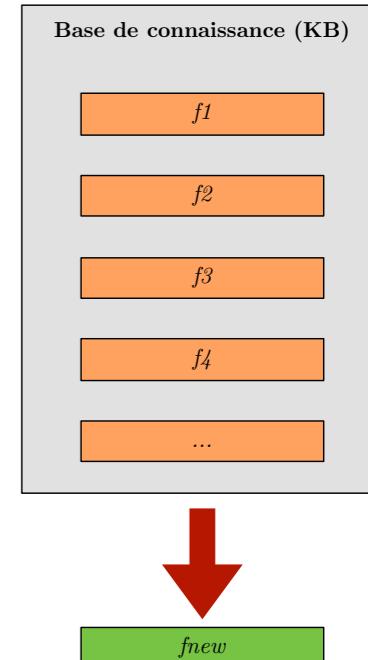
Monde de la syntaxe

Element 1: définition de formules correctes grammaticalement

Element 2: moyen de dériver de nouvelles formules (règle d'inférence)

Monde de la sémantique

Element: notion de conséquence logique entre KB et une formule



Intérêt: donne la garantie que pour chaque modèle de KB, la formule f est également vraie

Vérification actuelle: model checking (algorithme de recherche)



Pour une formule f , quelles sont les relations possibles entre $KB \vdash f$ et $KB \models f$?

Intuition: lorsqu'on dérive une formule, on souhaite ne pas dériver n'importe quoi

On souhaite dériver des règles également correctes d'un point de vue sémantique

Propriété de cohérence (*soundness*)



Ensemble des formules vraies (définition sémantique)

Ensemble de toutes les formules en conséquence logique de KB

$\{f \mid KB \models f\}$: **ensemble des formules vraies dans le monde de KB**

Visualisation: on peut considérer que toutes ces formules permettent de remplir exactement un récipient

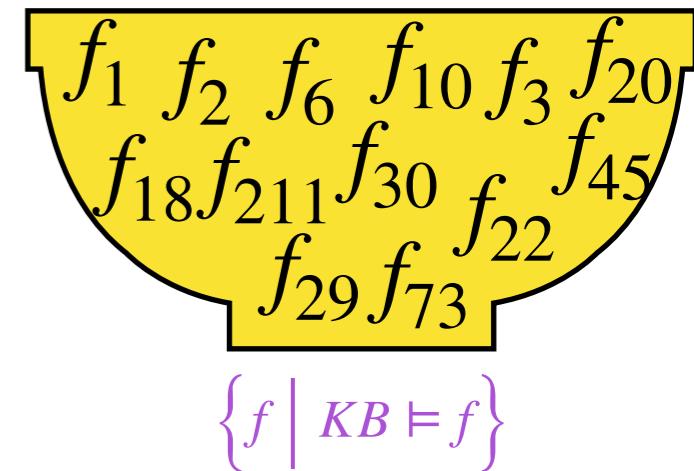
Intuition: ce récipient contient **UNIQUEMENT** et **TOUTES** les formules en conséquence logique avec KB



Propriété de cohérence (*soundness*)

Un algorithme d'inférence A est cohérent (*sound*) s'il ne génère QUE des formules en conséquence logique avec KB

$\{f \mid KB \vdash_A f\} \subseteq \{f \mid KB \models f\}$



Signification: on veut générer **UNIQUEMENT** les formules vraies, et rien d'autre

Visualisation: on génère **UNIQUEMENT** des formules appartenant au récipient

Propriété indispensable d'un algorithme d'inférence et des règles liées

Intérêt: nous assure qu'on va jamais générer des formules fausses

Limitation: aucune assurance que toutes les formules vraies peuvent être générées



Propriété de complétude (*completeness*)



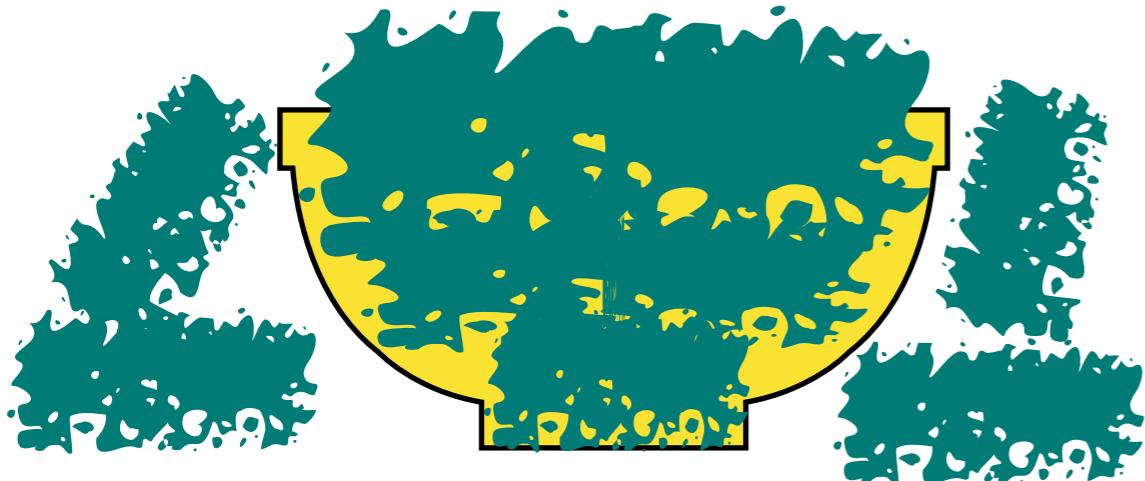
Propriété de complétude (*completeness*)

Un algorithme d'inférence A est complet s'il peut générer TOUTES les formules en conséquence logique avec KB

$$\{f \mid KB \vdash_A f\} \supseteq \{f \mid KB \models f\}$$

Signification: on veut pouvoir générer **TOUTES** les formules vraies

Visualisation: on génère **TOUTES** les formules du récipient, et potentiellement d'autres



Propriété désirable d'un algorithme d'inférence et des règles liées

Intérêt: assure qu'on peut générer toutes les formules qui sont vraies



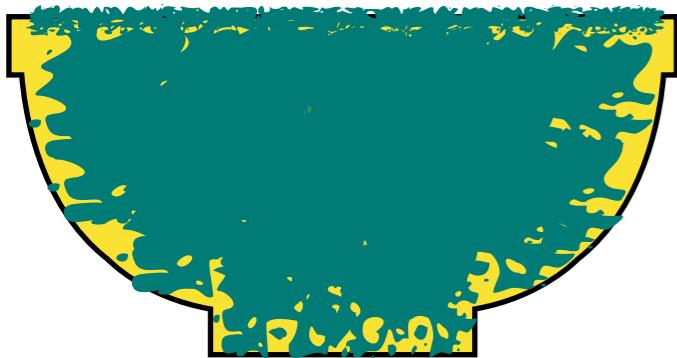
Limitation: risque de générer des formules qui ne sont pas vraies



Que souhaiterions-nous à l'idéal pour notre algorithme d'inférence (A) ?

Cohérence et complétude

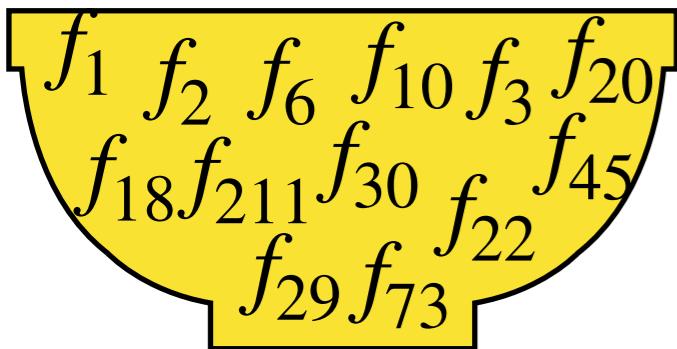
"Je veux la vérité, toute la vérité, rien que la vérité"



Complétude: nous assure de générer toutes les formules vraies

Cohérence: nous assure de ne générer que des formules vraies

Idéalement, on souhaite que notre algorithme possède ces deux propriétés



$$\{f \mid KB \vdash_A f\} \subseteq \{f \mid KB \models f\} \text{ ET } \{f \mid KB \vdash_A f\} \supseteq \{f \mid KB \models f\}$$

Visuellement: on souhaite couvrir tout et uniquement le récipient



Est-ce un objectif réaliste ?



Mauvaise nouvelle: ce n'est pas toujours facile (ni possible) à avoir

Prix à payer: perte de performances ou perte d'expressivité

En pratique: on doit souvent se contenter d'une, en préférant la cohérence

Ainsi, la complétude est parfois sacrifiée

Vérification de la cohérence

logicalInference(KB, R) :

while KB is modified :

$f_1, \dots, f_k = \text{selectFormulaFrom}(KB)$

if a matching rule $\frac{f_1, \dots, f_k}{g}$ exists in R :

$KB = KB \cup g$

return KB

?

Comment s'assurer qu'un algorithme d'inférence est cohérent ?

Principe: s'assurer que chaque règle utilisée est cohérente

$$\left\{ f \mid KB \vdash_A f \right\} \subseteq \left\{ f \mid KB \vDash f \right\}$$

Procédure: on peut vérifier directement les modèles des formules

$$M(\text{prémisses}) \subseteq M(\text{conclusion})$$

Définition d'une
conséquence logique

?

Est-ce que la règle du modus ponens $(\frac{p, p \Rightarrow q}{q})$ est cohérente ?

$$\frac{\text{Rain, Rain} \implies \text{Wet}}{\text{Wet}}$$

Vérification de modèles: $M(\text{Rain}) \cap M(\text{Rain} \implies \text{Wet}) \subseteq M(\text{Wet})$?

	Wet	0	1
0			
1			

\cap

	Wet	0	1
0			
1			

=

	Wet	0	1
0			
1			

	Wet	0	1
0			
1			

\cap

	Wet	0	1
0			
1			

?

OK

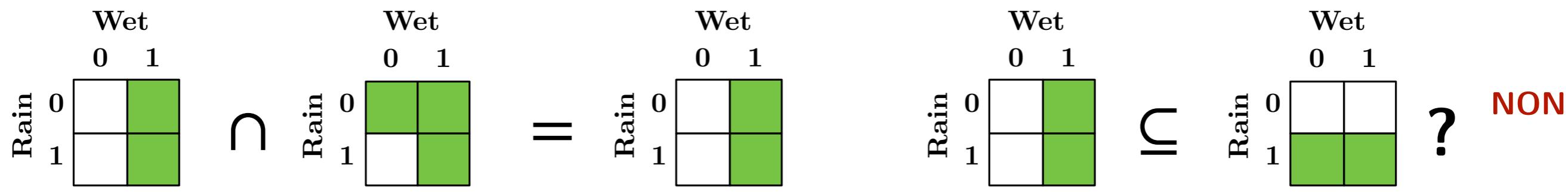
En conclusion, la règle du modus ponens est cohérente ! (exemple générique)

Vérification de la cohérence

? Est-ce que la règle $\frac{q, p \Rightarrow q}{p}$ est cohérente ?

$$\frac{\text{Wet, Rain} \Rightarrow \text{Wet}}{\text{Rain}}$$

Vérification de modèles: $M(\text{Wet}) \cap M(\text{Rain} \Rightarrow \text{Wet}) \subseteq M(\text{Rain})$?



Conclusion: la règle n'est pas cohérente !

Vérification de la cohérence

Cohérence: propriété fondamentale qui nous assure que les formules générées sont vraies

Bonne nouvelle: généralement facile à vérifier dans la logique des propositions

Procédure: on doit vérifier que les modèles des prémisses sont inclus dans ceux de la conclusion

Vérification: peut se faire via model checking car le nombre de possibilités est généralement faible

4. Soit la règle d'inférence $\frac{f \rightarrow g, \neg f \rightarrow \neg g}{f \equiv g}$. Cette règle est cohérente (*sound*).

Question de l'examen A2022

Vérification de la complétude

logicalInference(KB, R) :

while KB is modified :

$f_1, \dots, f_k = \text{selectFormulaFrom}(KB)$

if a matching rule $\frac{f_1, \dots, f_k}{g}$ exists in R :

$KB = KB \cup g$

return KB

?

Comment s'assurer qu'un algorithme d'inférence est complet ?

$$\left\{ f \mid KB \vdash_A f \right\} \supseteq \left\{ f \mid KB \models f \right\}$$

Besoin: l'algorithme doit pouvoir générer tout ce qui est vrai

"Ce qui est vrai": les formules en conséquences logiques de KB

?

Est-ce qu'un algorithme basé uniquement sur la règle du modus ponens ($\frac{p, p \Rightarrow q}{q}$) est complet ?

Soit la KB suivante: $KB = \{\text{Rain}, \text{Rain} \vee \text{Snow} \Rightarrow \text{Wet}\}$

Considérons la formule: $f = \text{Wet}$

Règle utilisable: $\frac{p, p \Rightarrow q}{q}$

?

Est-ce que f est une conséquence logique de KB ?

Oui: vérification possible par modèle checking

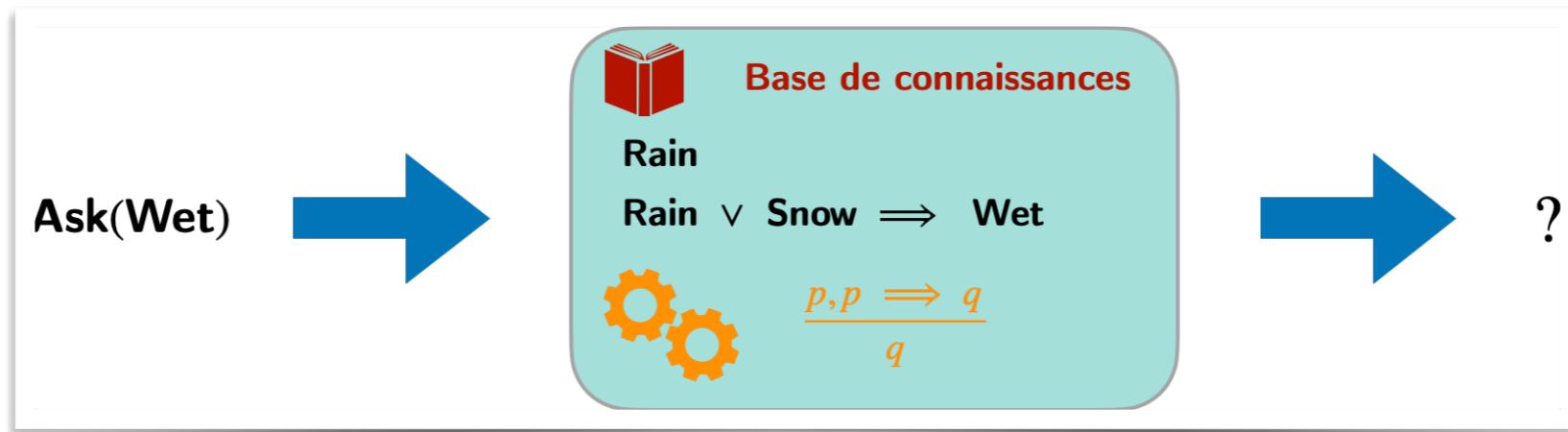
?

Est-ce que f est dérivable par l'algorithme ?

Non: le modus ponens n'est pas applicable dans KB

Un algorithme basé juste sur le modus ponens n'est pas complet dans la logique des propositions

Impact d'un algorithme incomplet



Quel est l'impact d'un algorithme d'inférence incomplet ?

Conséquence: l'agent sera incapable de répondre, même si WET est une conséquence logique de KB



Que peut-on faire pour pallier cette difficulté ?



Option 1: considérer des règles d'inférences plus expressives

Option 2: se rabattre sur une logique plus simple

Table des matières

Agents logiques

✓ 1. Motivation et définition d'un agent logique

✓ 2. Représentation des connaissances

✓ 3. Système logique (syntaxe, sémantique, et règles d'inférences)

✓ 4. Formalisation de la logique des propositions

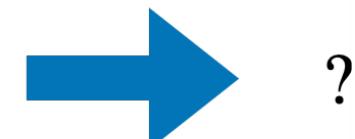
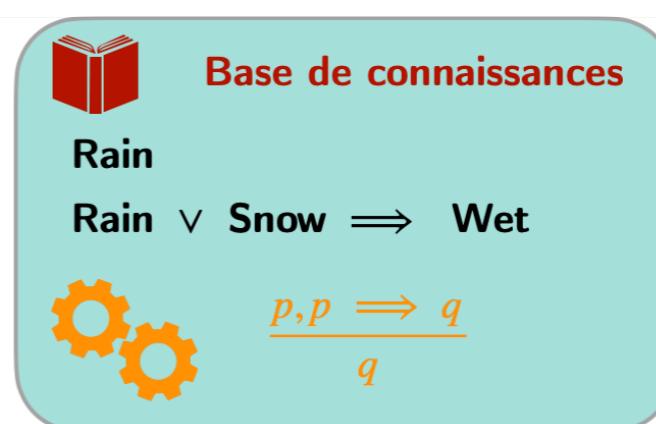
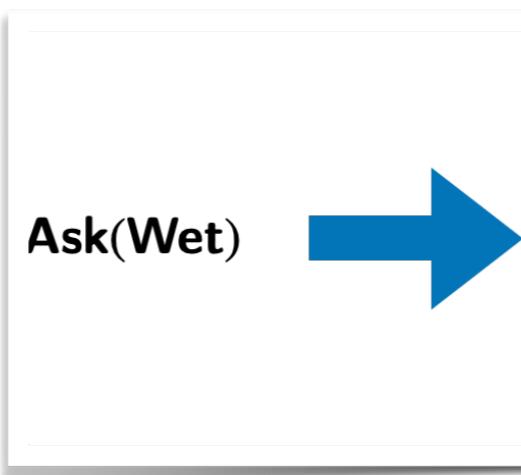
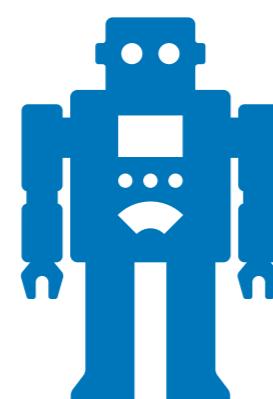
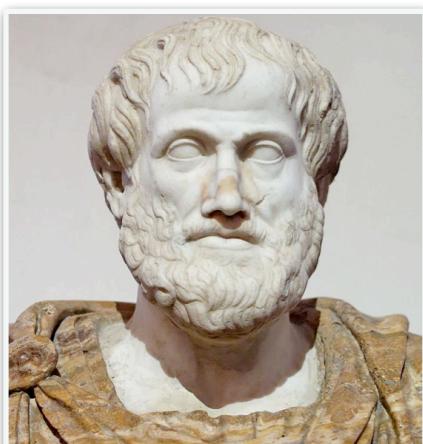
✓ 5. Raisonnement par model-checking

✓ 6. Inférences logiques et propriétés

7. Algorithme de résolution

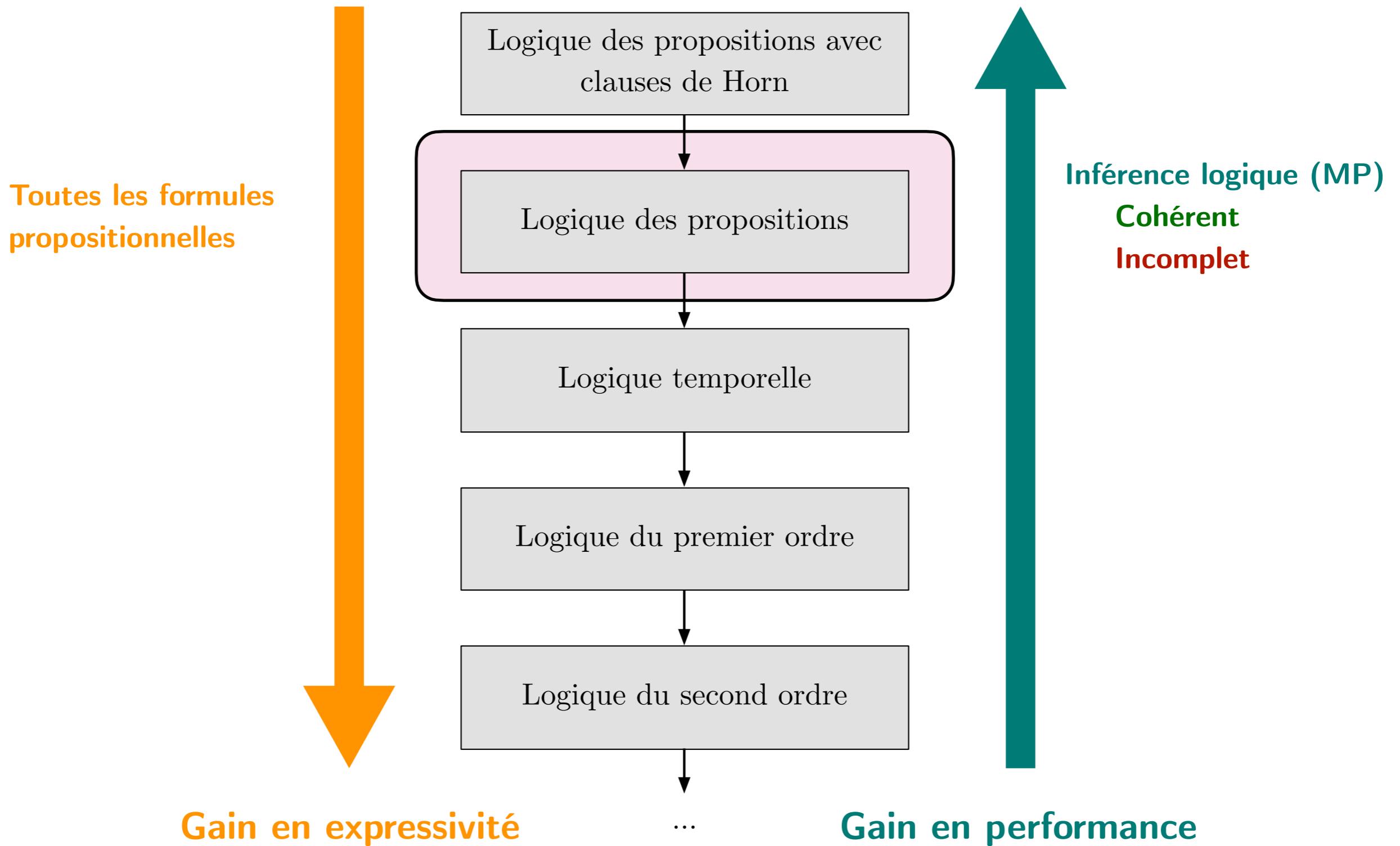
8. Clauses de Horn

9. Extension à d'autres systèmes logiques



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

Etat de la situation



Règle de résolution



Le Modus Ponens n'est complet. Pourrait-on envisager d'autres règles d'inférence ?

 **Règle de résolution**

Règle d'inférence sous la forme suivante:

$$\frac{(f_1 \vee \dots \vee f_n \vee p) \quad (\neg p \vee g_1 \vee \dots \vee g_m)}{f_1 \vee \dots \vee f_n \vee g_1 \vee \dots \vee g_m}$$

Prémices: supposons qu'une formule (p) et sa négation (non p) apparaissent dans deux disjonctions de KB

Conclusion: alors on peut inférer une nouvelle règle en combinant la conjonction et en retirant la formule

Exemple: illustration de la règle de résolution

$$\frac{\text{Rain} \vee \text{Snow} \quad \neg \text{Snow} \vee \text{Wind}}{\text{Rain} \vee \text{Wind}}$$

Je sais qu'il pleut ou qu'il neige

Je sais qu'il vente ou qu'il neige pas

Si il ne neige pas: alors il pleut obligatoirement

Si il neige: alors il vente obligatoirement

Je peux alors conclure qu'il pleut ou qu'il vente (ou les deux)



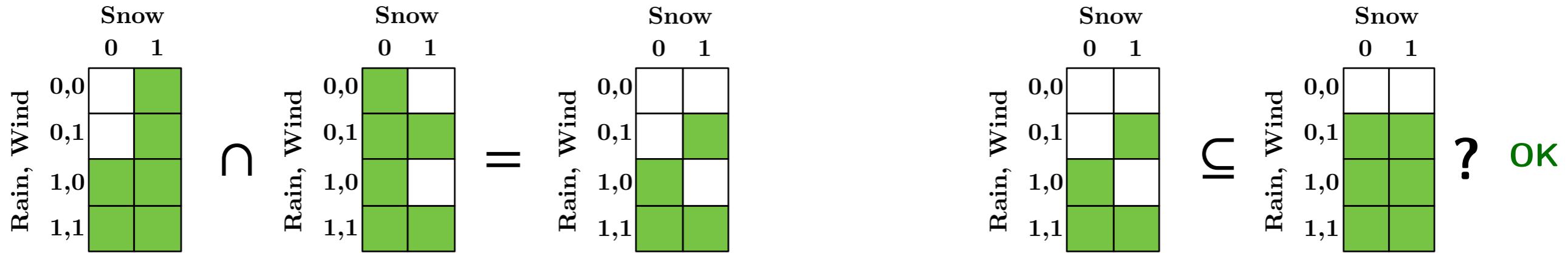
Règle de résolution



Est-ce que la règle de résolution est cohérente ?

$$\frac{\text{Rain} \vee \text{Snow} \quad \neg \text{Snow} \vee \text{Wind}}{\text{Rain} \vee \text{Wind}}$$

Vérification de modèles: $M(\text{Rain} \vee \text{Snow}) \cap M(\neg \text{Snow} \vee \text{Wind}) \subseteq M(\text{Rain} \vee \text{Wind})$?



Conclusion: la règle de résolution est cohérente ! (formalisation de l'exemple précédent)

Note: le principe est similaire pour des conjonctions impliquant plus de formules



Est-ce que la règle de résolution est complète ?

Première intuition: il semble que non !

$$KB = \{\text{Rain} \vee \text{Snow}, \neg \text{Snow} \vee \text{Wind}\} \quad f: \text{Snow} \implies \text{Wind}$$

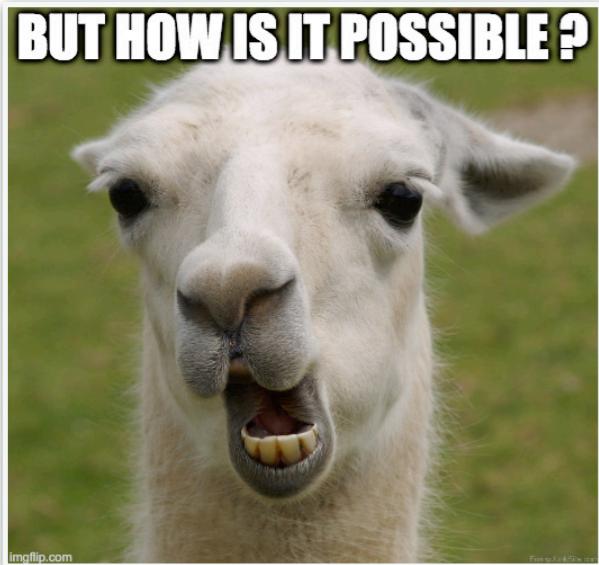
Observation: La formule f est bien une conséquence logique de KB

Limitation: la règle de résolution ne fonctionne pas directement avec des implications

Propriétés de la règle de résolution

 **Propriétés de la règle de résolution**

La règle de résolution est **complète** et **cohérente** en logique des propositions

$$\frac{(f_1 \vee \dots \vee f_n \vee p) \quad (\neg p \vee g_1 \vee \dots \vee g_m)}{f_1 \vee \dots \vee f_n \vee g_1 \vee \dots \vee g_m}$$


Cohérence: démonstration à la diapositive précédente

Complétude: cela semble conte-intuitif !

Limitation: la règle est limitée aux règles n'ayant que des OU logiques

Cela empêche notamment les inférences sur des formules ayant des implications

Astuce: on peut transformer n'importe quelle formule en une forme simplifiée

Exigence sur la forme requise: la règle de résolution ne fonctionne que sur une KB constituées de **clauses**

Clause: une disjonction (\vee) de littéraux

Littéral: un symbole propositionnel ou sa négation (P ou $\neg P$)

Idée: élaborer des règles de transformations permettant de ramener n'importe quelle formule en clause

Forme normale conjonctive et simplification



Forme normale conjonctive (CNF - *Conjonctive normal form*)

Une formule CNF est une conjonction (\wedge) de clauses

Précédemment: prendre l'ensemble des formules d'une KB est identique à prendre leur conjonction

$\{f_1, f_2, \dots, f_n\}$ est équivalent à la (grande) formule $f_1 \wedge f_2 \wedge \dots \wedge f_n$

Exemple: $KB = \{\text{Rain} \vee \text{Snow}, \neg \text{Snow} \vee \text{Wind}\} \iff KB = (\text{Rain} \vee \text{Snow}) \wedge (\neg \text{Snow} \vee \text{Wind})$

Conséquence: une KB peut être représentée comme en CNF si toutes les formules sont des clauses



Simplification de formules en logique des propositions

N'importe quelle formule f peut être ré-exprimée en une formule CNF équivalente g

$$M(f) = M(g)$$

Note: formellement, cela signifie que les deux formules ont exactement les mêmes modèles

Conséquence: n'importe quelle KB peut être transformée en une formule CNF

Intérêt: assure que la règle de résolution peut être utilisée sur n'importe quelle KB (après simplification)

Attention: on n'a pas prouvé que la règle de résolution est complète (preuve en lecture complémentaire)

La base de connaissance est représentée comme une conjonction de deux clauses



Transformation en CNF



Comment réaliser cette transformation ?

Idée: exploiter successivement des équivalences logiques connues pour transformer la formule

Transformation: fonctionne en 5 étapes successives

Etape 1: Elimination des équivalences logiques (\Leftrightarrow)

(une équivalence est une implication dans les deux sens)

$$\text{Règle: } \frac{(f \Leftrightarrow g)}{((f \Rightarrow g) \wedge (g \Rightarrow f))}$$

Etape 2: Elimination des implications (\Rightarrow)

(vérification triviale par model checking)

$$\text{Règle: } \frac{(f \Rightarrow g)}{(\neg f \vee g)}$$

Etape 3: Avancer les négations (\neg) pour les coller aux littéraux

(application de la Loi de De Morgan)

$$\text{Règles: } \frac{\neg(f \vee g)}{\neg f \wedge \neg g} \text{ et } \frac{\neg(f \wedge g)}{\neg f \vee \neg g}$$

Etape 4: Eliminer les doubles négations ($\neg\neg$)

(deux négations s'annulent mutuellement)

$$\text{Règle: } \frac{\neg\neg f}{f}$$

Etape 5: Distribuer les disjonctions (\vee) dans les conjonctions (\wedge)

$$\text{Règle: } \frac{f \vee (g \wedge h)}{(f \vee g) \wedge (f \vee h)}$$

Exercice supplémentaire: vérifier l'exactitude de ces formules via model checking

On est ainsi capable de transformer n'importe quelle formule en CNF (logique des propositions)

Transformation en CNF: exemple

Exemple de transformation pour une formule $f: P \iff (Q \vee R)$

Etape 1: Elimination de l'équivalence logique

$$f: (P \implies (Q \vee R)) \wedge ((Q \vee R) \implies P)$$

Règle: $\frac{(f \iff g)}{((f \implies g) \wedge (g \implies f))}$

Etape 2: Elimination des implications

$$f: (\neg P \vee (Q \vee R)) \wedge (\neg(Q \vee R) \vee P)$$

Règle: $\frac{(f \implies g)}{(\neg f \vee g)}$

Etape 3: Avancer les négations

$$f: (\neg P \vee (Q \vee R)) \wedge ((\neg Q \wedge \neg R) \vee P)$$

Règles: $\frac{\neg(f \vee g)}{\neg f \wedge \neg g}$ et $\frac{\neg(f \wedge g)}{\neg f \vee \neg g}$

Etape 4: Retirer les doubles négations (rien à faire)

Règle: $\frac{\neg \neg f}{f}$

Etape 5: Distribuer les disjonctions

$$f: (\neg P \vee (Q \vee R)) \wedge (\neg Q \vee P) \wedge (\neg R \vee P)$$

Règle: $\frac{f \vee (g \wedge h)}{(f \vee g) \wedge (f \vee h)}$

Formule finale (après retrait des parenthèses inutiles):

$$f: (\neg P \vee Q \vee R) \wedge (\neg Q \vee P) \wedge (\neg R \vee P) \text{ (CNF de 3 clauses)}$$

Rappels de concepts



Objectif initial: vérifier si une formule en requête (ASK) est en conséquence logique avec KB



Comment utiliser tous ces mécanismes pour concevoir notre agent ?

Précédemment: on a vu qu'il existait un lien entre conséquence logique et satisfiabilité (slide 32)



Relation 1: prouver la conséquence logique ($KB \models f$) revient à prouver la non satisfiabilité de $KB \cup \{\neg f\}$

Relation 2: de manière équivalente, cela revient à prouver que KB et $\neg f$ sont en contradiction

Idée: exploiter ces relations pour établir un algorithme consistant à calculer si la formule peut être inférée

Note: pour avoir la certitude d'une réponse correcte, il faut que l'algorithme soit cohérent et complet

On va ainsi construire un algorithme basé sur ces principes pour vérifier la conséquence logique

Algorithme de résolution

L'objectif est de prouver qu'une formule f est une conséquence logique de KB

Principes de l'algorithme: procédure en 4 étapes:

Etape 1: ajouter temporairement $\neg f$ à la base de connaissance (KB)

Etape 2: convertir toutes les formules de KB en CNF

Etape 3: appliquer répétitivement la règle de résolution à chaque paire de clauses

Etape 4: si la clause vide est générée (non-satisfiabilité), alors on a prouvé la conséquence logique

resolutionAlgorithm(KB, f) :

$F = \text{formulaFrom}(KB) \cup \{\neg f\}$

$C = \text{transformIntoCNF}(F)$

while true :

$N = \{\}$

for each pair $(c_i, c_j) \in (C, C)$:

$c_{new} = \text{resolutionRule}(c_i, c_j)$

if $c_{new} = \emptyset$: return true

$N = N \cup \{c_{new}\}$

if $N \subseteq C$: return false

$C = C \cup N$

Etape 1: ajout temporaire de *non* f à KB

Etape 2: transformation en CNF, ce qui donne un ensemble de clauses

Etape 3: tant que l'algorithme n'a pas su confirmer la conséquence logique

Ensemble des nouvelles clauses générées par l'itération actuelle

On considère toutes les paires de clauses

On applique la règle de résolution dessus

Etape 4: si l'ensemble vide est généré, alors la cons. logique est prouvée (contradiction trouvée)

On rajoute la nouvelle clause à notre ensemble

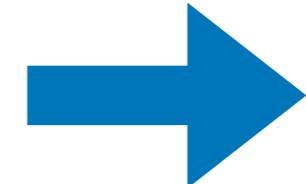
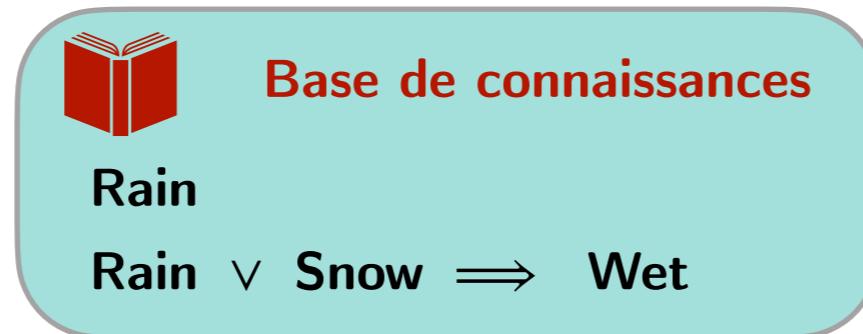
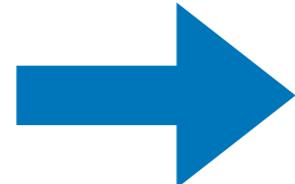
Si on a rien généré de nouveau, l'algorithme a convergé: f n'est pas une conséquence logique

On rajoute les clauses générées à notre ensemble de clause pour la prochaine itération

Intuition: appliquer la règle de résolution en boucle jusqu'à avoir une réponse (vrai ou faux)

Algorithme de résolution: exemple

Ask(Wet)



? Est-ce que *Wet* est une cons. logique de KB ?

$$KB = \{\text{Rain}, (\text{Rain} \vee \text{Snow}) \implies \text{Wet}\}$$
$$f = \text{Wet}$$

Etape 1: ajouter $\neg f$ à la base de connaissance (KB)

$$KB = \{\text{Rain}, (\text{Rain} \vee \text{Snow}) \implies \text{Wet}, \neg \text{Wet}\}$$

Etape 2: convertir toutes les formules de KB en CNF

$$KB = \{\text{Rain}, \neg \text{Rain} \vee \text{Wet}, \neg \text{Snow} \vee \text{Wet}, \neg \text{Wet}\}$$

Etape 3: appliquer répétitivement la règle de résolution à chaque paire de clauses

Inférence 1:

$$\frac{\text{Rain} \quad \neg \text{Rain} \vee \text{Wet}}{\text{Wet}}$$

$$KB = \{\text{Rain}, \neg \text{Rain} \vee \text{Wet}, \neg \text{Snow} \vee \text{Wet}, \neg \text{Wet}, \text{Wet}\}$$

Note: toutes les inférences ne sont pas montrées

Inférence 2:

$$\frac{\neg \text{Wet} \quad \text{Wet}}{\emptyset}$$

Contradiction obtenue: Wet est une conséquence logique de KB

Pour rappel, on n'était pas capable de prouver cela avec le modus ponens

Algorithme de résolution: propriétés

Cohérence

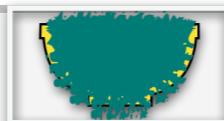
$$\{f \mid KB \vdash_A f\} \subseteq \{f \mid KB \models f\}$$

Complétude

$$\{f \mid KB \vdash_A f\} \supseteq \{f \mid KB \models f\}$$

Règle de résolution

$$\frac{(f_1 \vee \dots \vee f_n \vee p) \quad (\neg p \vee g_1 \vee \dots \vee g_m)}{f_1 \vee \dots \vee f_n \vee g_1 \vee \dots \vee g_m}$$



Qu'en est-il de la complexité temporelle ?

I HAVE A BAD FEELING

ABOUT THIS

Considérons que l'on ait n symboles propositionnels impliqués dans KB

Règle de résolution: ajout potentiel de clauses d'au plus $n-1$ symboles

Conséquence: un nombre exponentiel de clauses peut ainsi être généré

Mauvaise nouvelle: l'algorithme a une complexité temporelle exponentielle

Note: cela n'est pas étonnant, sachant que c'est un problème NP-complet

Algorithme de résolution: visualisation

Base de connaissances: $KB = \{\neg P \vee Q, \neg Q \vee R \vee P, \neg R \vee Q, \neg Q\}$

Règle à prouver: $f = \neg R$

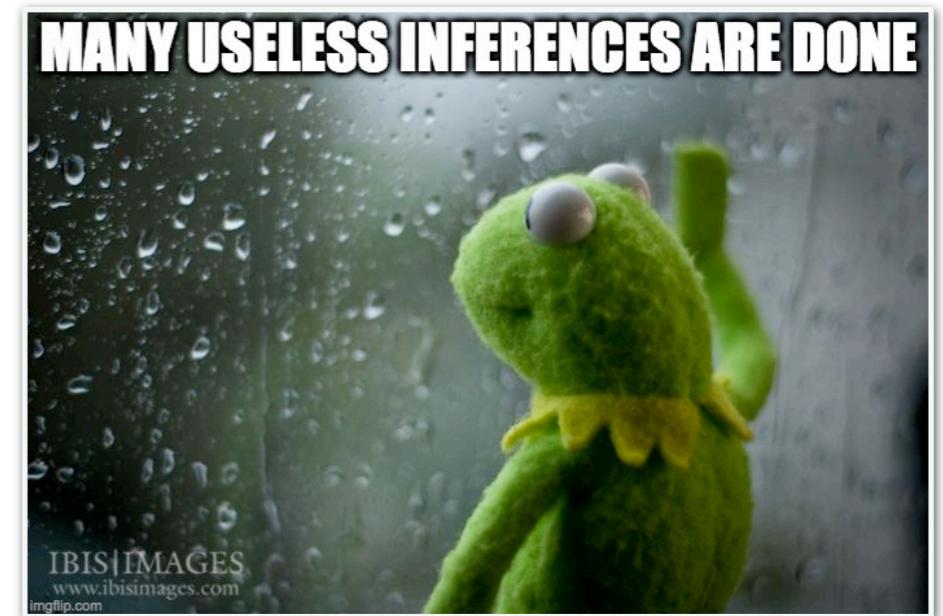
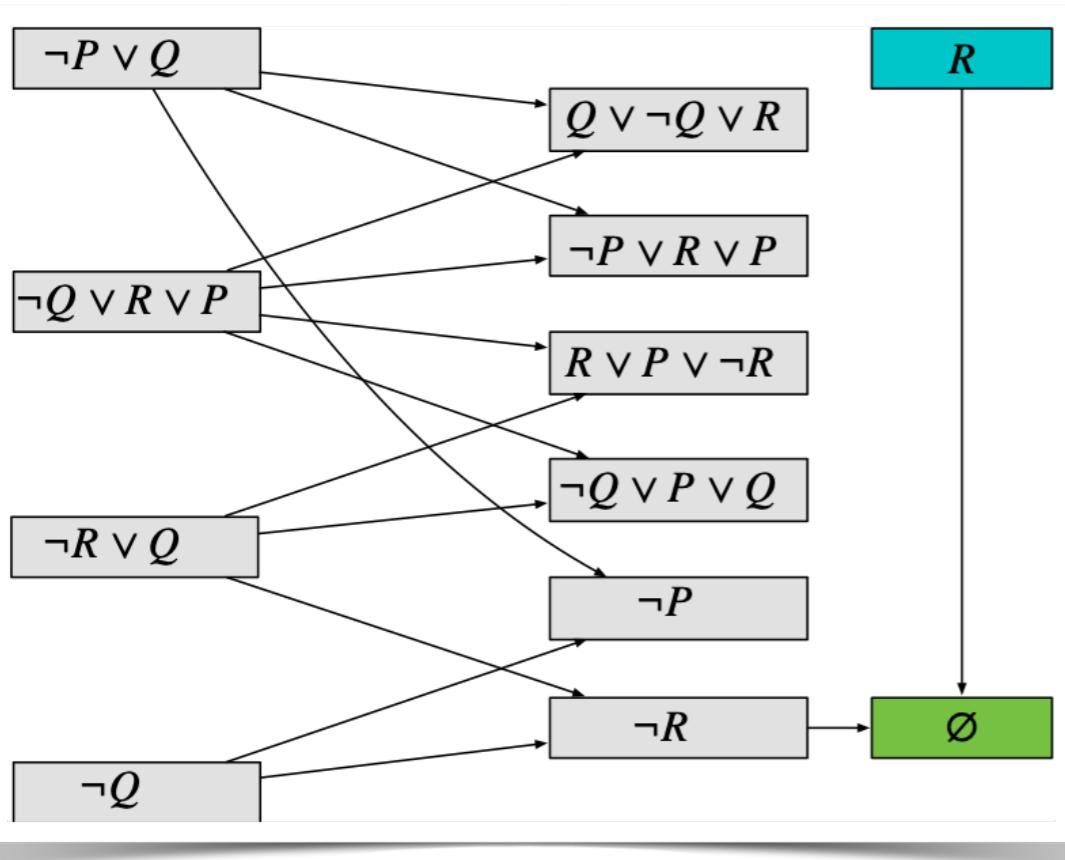
Base intermédiaire: $KB' = \{\neg P \vee Q, \neg Q \vee R \vee P, \neg R \vee Q, \neg Q, \textcolor{red}{R}\}$

Que pensez-vous de cette exécution ?

Visualisation: comme un graphe dirigé

(1) Chaque arête correspond à une application de la règle de résolution

(2) Chaque noeud correspond à une clause dans KB (présente initialement ou générée)



Peut-on gagner en efficacité ?

Amélioration possible: choisir adéquatement les clauses pour appliquer la règle (heuristique de sélection)

Table des matières

Agents logiques

✓ 1. Motivation et définition d'un agent logique

✓ 2. Représentation des connaissances

✓ 3. Système logique (syntaxe, sémantique, et règles d'inférences)

✓ 4. Formalisation de la logique des propositions

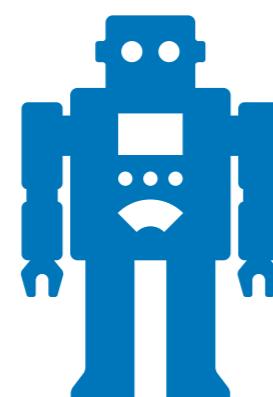
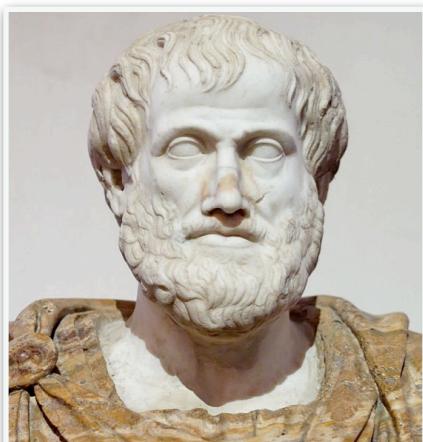
✓ 5. Raisonnement par model-checking

✓ 6. Inférences logiques et propriétés

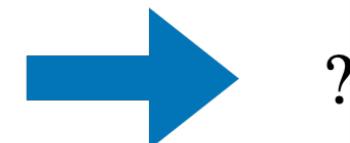
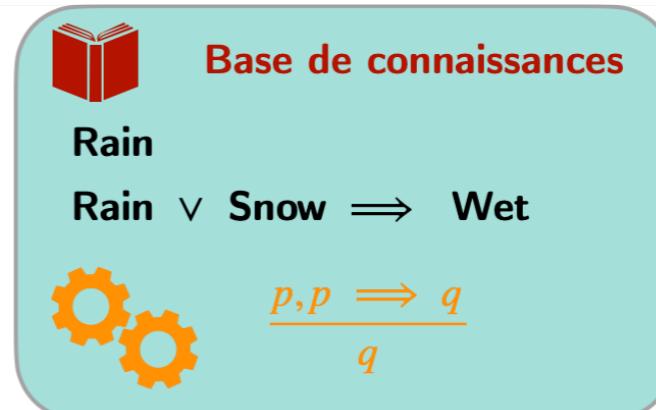
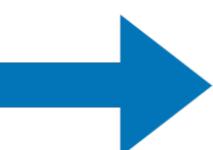
✓ 7. Algorithme de résolution

8. Clauses de Horn

9. Extension à d'autres systèmes logiques

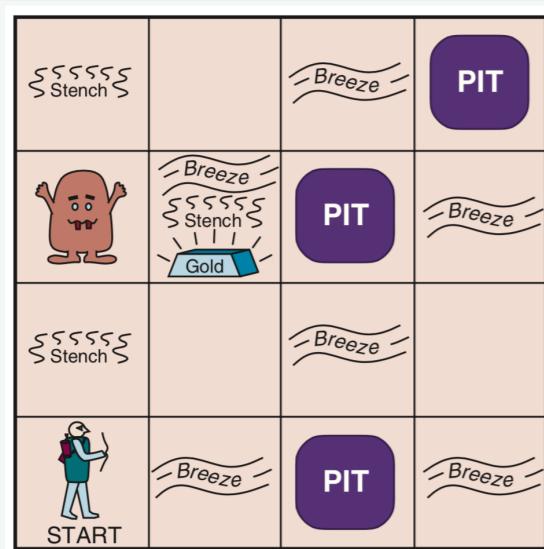


Ask(Wet)



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

Cas d'étude: le monde de Wumpus



Mise en situation: vous entrez dans une grotte en vue de trouver un trésor
Difficultés: cette grotte présente plusieurs dangers susceptibles de vous tuer...
Environnement: statique (ne change pas avec le temps), et partiellement observable

Grotte: environnement constitué de 16 cases disposées en carré

Wumpus: monstre qui vous dévore si vous entrez dans sa case

Puanteur (stench): information présente aux cases adjacentes du Wumpus

Trous (pits): pièges vous tuant si vous entrez dans la case

Air frais (breeze): information présente aux cases adjacentes aux trous

Lingot d'or: trésor que doit trouver l'agent

Objectif de l'agent: trouver le lingot d'or sans mourir

Position initiale: en bas à gauche - case (1,1)

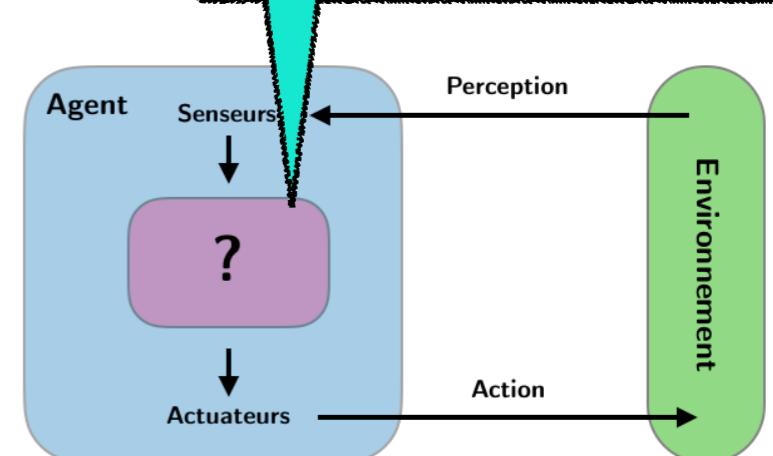
Actions: déplacement en haut, bas, gauche, ou droite

Capteurs sensoriels: présence d'or, de puanteur, ou d'air frais sur la case

L'agent n'a connaissance que de sa case, et de ce qu'il a déjà visité

Difficulté: on doit absolument éviter les cases amenant à la mort de l'agent

Raisonnement logique



Ainsi, le raisonnement par essais-erreurs pour découvrir les cases n'est pas possible dans ce contexte

Le monde de Wumpus: approche de résolution

Etape 1: faire l'état de toutes nos connaissances concernant l'environnement

Etape 2: les exprimer dans un système logique

Etape 3: effectuer des inférences pour connaître les cases sécuritaires

Etape 4: déplacer l'agent dans ces cases pour obtenir de nouvelles connaissances

Connaissance 1: présence de trous, de Wumpus, et d'un lingot dans la grotte

Connaissance 2: les trous relâchent de l'air frais sur les cases adjacentes

Connaissance 3: les Wumpus relâchent de la puanteur sur les cases adjacentes

Chaque case peut contenir: wumpus, trou, lingot, air frais, et puanteur

Situation initiale de l'agent: l'agent n'a observé que sa case de départ (1,1)

Connaissances observées: rien sur la case (pas de Wumpus, etc., sur la case (1,1))

Connaissance du monde: les relations entre air frais/puanteur et Wumpus/trou

Le monde de Wumpus: encodage des connaissances

Encodons toutes ces connaissances dans la logique des propositions

$P_{1,1}$: Vrai s'il y a un trou (pit) dans la case (1,1)

$W_{1,1}$: Vrai s'il y a un Wumpus dans la case (1,1)

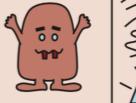
$B_{1,1}$: Vrai s'il y a de l'air frais (breeze) dans la case (1,1)

$S_{1,1}$: Vrai s'il y a de la puanteur (stench) dans la case (1,1)

$L_{1,1}$: Vrai s'il y a un lingot dans la case (1,1)

Autres cases: même encodage (16 cases au total)

Nombre de symboles: $5 \times 16 = 80$

$\Sigma\Sigma\Sigma\Sigma\Sigma$ Stench			PIT
	 $\Sigma\Sigma\Sigma\Sigma\Sigma$ Stench		PIT
$\Sigma\Sigma\Sigma\Sigma\Sigma$ Stench			
 START		PIT	

? Combien de symboles propositionnels avons-nous ?

Etat actuel: on a une base de connaissances pour l'agent



Base de connaissances

Faits: $\neg P_{1,1}, \neg W_{1,1}, \neg B_{1,1}, \neg S_{1,1}, \neg L_{1,1}$

R1: $B_{1,1} \iff (P_{1,2} \vee P_{2,1})$

R2: $S_{1,1} \iff (W_{1,2} \vee W_{2,1})$

La case (1,1) est vide

Il y a un trou en (1,2) ou (2,1) si et seulement s'il y a de l'air frais en (1,1)

Il y a un Wumpus en (1,2) ou (2,1) si et seulement s'il y a de la puanteur en (1,1)

? Quelle(s) action(s) l'agent peut-il prendre pour obtenir plus d'informations sécuritairement ?

Le monde de Wumpus: acquisition de connaissances

Obtenons de nouvelles connaissances via l'algorithme de résolution!

Requête ASK: assurons nous qu'il n'y a pas de trous en (2,1)

Formule à prouver: $\neg P_{2,1} (f)$

Etape 1: ajouter $\neg f$ à la base de connaissance (KB)

Etape 2: convertir toutes les formules de KB en CNF

Etape 3: appliquer répétitivement la règle de résolution à chaque paire de clauses

Etape 4: si la clause vide est générée (non-satisfiabilité), alors on a prouvé la conséquence logique



Base de connaissances

Faits: $\neg P_{1,1}, \neg W_{1,1}, \neg B_{1,1}, \neg S_{1,1}, \neg L_{1,1}$

R1-C1: $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$

R1-C2: $\neg P_{1,2} \vee B_{1,1}$

R1-C3: $\neg P_{2,1} \vee B_{1,1}$

R2-C1: $\neg S_{1,1} \vee W_{1,2} \vee W_{2,1}$

R2-C2: $\neg W_{1,2} \vee S_{1,1}$ **N-C1:** $\neg P_{2,1}$

R2-C3: $\neg W_{2,1} \vee S_{1,1}$

O: $P_{2,1}$

$$\frac{(f_1 \vee \dots \vee f_n \vee p) \quad (\neg p \vee g_1 \vee \dots \vee g_m)}{f_1 \vee \dots \vee f_n \vee g_1 \vee \dots \vee g_m}$$

$\xi\xi\xi\xi\xi$ Stench		$\sim\!\!\!\sim$ Breeze	PIT
	$\sim\!\!\!\sim$ Breeze $\xi\xi\xi\xi\xi$ Stench Gold	PIT	$\sim\!\!\!\sim$ Breeze
$\xi\xi\xi\xi\xi$ Stench		$\sim\!\!\!\sim$ Breeze	
START	$\sim\!\!\!\sim$ Breeze	PIT	$\sim\!\!\!\sim$ Breeze
1	2	3	4

Inférence 1 (R1-C2 avec un fait): $\frac{\neg P_{2,1} \vee B_{1,1} \quad \neg B_{1,1}}{\neg P_{2,1}}$

Inférence 2 (N-C1 avec O): $\frac{\neg P_{2,1} \quad P_{2,1}}{\emptyset}$

Résultat: contradiction

On a prouvé **f**: certitude qu'il n'y a pas de trous en (2,1)

Même raisonnement: il n'y pas de Wumpus en (2,1)

Même raisonnement: il n'y pas de trous en (1,2)

Même raisonnement: il n'y pas de Wumpus en (1,2)

Le monde de Wumpus: acquisition de connaissances

On peut continuer le processus pour obtenir de nouvelles connaissances

Actions de l'agent: on peut découvrir dans risque ce qu'il y a en (1,2) et (2,1)

Case (1,2): présence uniquement de puanteur $\neg P_{1,2}, \neg W_{1,2}, \neg B_{1,2}, S_{1,2}, \neg L_{1,2}$

Case (2,1): présence uniquement d'air frais $\neg P_{2,1}, \neg W_{2,1}, B_{2,1}, \neg S_{2,1}, \neg L_{2,1}$

Connaissances: mise-à-jour avec les nouvelles informations

$$R3: B_{2,1} \iff (P_{1,1} \vee P_{3,1} \vee P_{2,2})$$

$$R4: S_{2,1} \iff (W_{1,1} \vee W_{3,1} \vee W_{2,2})$$

$$R5: B_{1,2} \iff (P_{1,1} \vee P_{1,3} \vee P_{2,2})$$

$$R6: S_{1,2} \iff (W_{1,1} \vee W_{1,3} \vee W_{2,2})$$

$\lesssim \lesssim \lesssim \lesssim \lesssim$ Stench		$\sim \sim \sim \sim$ Breeze	PIT
(Wumpus)	$\sim \sim \sim \sim$ Breeze $\lesssim \lesssim \lesssim \lesssim$ Stench (Gold)	PIT	$\sim \sim \sim \sim$ Breeze
$\lesssim \lesssim \lesssim \lesssim \lesssim$ Stench		$\sim \sim \sim \sim$ Breeze	
(START)	$\sim \sim \sim \sim$ Breeze	PIT	$\sim \sim \sim \sim$ Breeze
1	2	X	3
y	2		1

Faits: $\neg P_{1,1}, \neg W_{1,1}, \neg B_{1,1}, \neg S_{1,1}, \neg L_{1,1}$

Faits: $\neg P_{2,1}, \neg W_{2,1}, B_{2,1}, \neg S_{2,1}, \neg L_{2,1}$

Faits: $\neg P_{1,2}, \neg W_{1,2}, \neg B_{1,2}, S_{1,2}, \neg L_{1,2}$

R1-C1: $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$

R1-C2: $\neg P_{1,2} \vee B_{1,1}$

R1-C3: $\neg P_{2,1} \vee B_{1,1}$

R2-C1: $\neg S_{1,1} \vee W_{1,2} \vee W_{2,1}$

R2-C2: $\neg W_{1,2} \vee S_{1,1}$

R2-C3: $\neg W_{2,1} \vee S_{1,1}$

R3-C1: $\neg B_{2,1} \vee P_{1,1} \vee P_{3,1} \vee P_{2,2}$

R3-C2: $\neg P_{1,1} \vee B_{2,1}$

R3-C3: $\neg P_{3,1} \vee B_{2,1}$

R3-C4: $\neg P_{2,2} \vee B_{2,1}$

R4-C1: $\neg S_{2,1} \vee W_{1,1} \vee W_{3,1} \vee W_{2,2}$

R4-C2: $\neg W_{1,1} \vee S_{2,1}$

R4-C3: $\neg W_{3,1} \vee S_{2,1}$

R4-C4: $\neg W_{2,2} \vee S_{2,1}$



Base de connaissances

R5-C1: $\neg B_{1,2} \vee P_{1,1} \vee P_{1,3} \vee P_{2,2}$

R5-C2: $\neg P_{1,1} \vee B_{1,2}$

R5-C3: $\neg P_{1,3} \vee B_{1,2}$

R5-C4: $\neg P_{2,2} \vee B_{1,2}$

R6-C1: $\neg S_{1,2} \vee W_{1,1} \vee W_{1,3} \vee W_{2,2}$

R6-C2: $\neg W_{1,1} \vee S_{1,2}$

R6-C3: $\neg W_{1,3} \vee S_{1,2}$

R6-C4: $\neg W_{2,2} \vee S_{1,2}$

Le monde de Wumpus: acquisition de connaissances

Faits: $\neg P_{1,1}, \neg W_{1,1}, \neg B_{1,1}, \neg S_{1,1}, \neg L_{1,1}$

Faits: $\neg P_{2,1}, \neg W_{2,1}, B_{2,1}, \neg S_{2,1}, \neg L_{2,1}$

Faits: $\neg P_{1,2}, \neg W_{1,2}, \neg B_{1,2}, S_{1,2}, \neg L_{1,2}$

R1-C1: $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$

R1-C2: $\neg P_{1,2} \vee B_{1,1}$

R1-C3: $\neg P_{2,1} \vee B_{1,1}$

R2-C1: $\neg S_{1,1} \vee W_{1,2} \vee W_{2,1}$

R2-C2: $\neg W_{1,2} \vee S_{1,1}$

R2-C3: $\neg W_{2,1} \vee S_{1,1}$

R3-C1: $\neg B_{2,1} \vee P_{1,1} \vee P_{3,1} \vee P_{2,2}$

R3-C2: $\neg P_{1,1} \vee B_{2,1}$

R3-C3: $\neg P_{3,1} \vee B_{2,1}$

R3-C4: $\neg P_{2,2} \vee B_{2,1}$

R4-C1: $\neg S_{2,1} \vee W_{1,1} \vee W_{3,1} \vee W_{2,2}$

R4-C2: $\neg W_{1,1} \vee S_{2,1}$

R4-C3: $\neg W_{3,1} \vee S_{2,1}$

R4-C4: $\neg W_{2,2} \vee S_{2,1}$



Base de connaissances

R5-C1: $\neg B_{1,2} \vee P_{1,1} \vee P_{1,3} \vee P_{2,2}$

R5-C2: $\neg P_{1,1} \vee B_{1,2}$

R5-C3: $\neg P_{1,3} \vee B_{1,2}$

R5-C4: $\neg P_{2,2} \vee B_{1,2}$

R6-C1: $\neg S_{1,2} \vee W_{1,1} \vee W_{1,3} \vee W_{2,2}$

R6-C2: $\neg W_{1,1} \vee S_{1,2}$

R6-C3: $\neg W_{1,3} \vee S_{1,2}$

R6-C4: $\neg W_{2,2} \vee S_{1,2}$



Est-il sécuritaire de se déplacer en (2,2) ?

Difficulté: information est non triviale à obtenir

Observation 1: Par l'air frais en (2,1), il est possible qu'il y ait un trou en (2,2)

Observation 2: Par la puanteur en (1,2), il est possible qu'il y ait un Wumpus en (2,2)

	1	2	3	4
4	Stench		$Breeze$	PIT
3	Wumpus	$Breeze$ Stench	Gold	$Breeze$
2	Stench		$Breeze$	
1	START	$Breeze$	PIT	$Breeze$
y	1	2	3	4
x	1	2	3	4

Appliquons l'algorithme de résolution pour répondre à cette question

Le monde de Wumpus

Faits: $\neg P_{1,1}, \neg W_{1,1}, \neg B_{1,1}, \neg S_{1,1}, \neg L_{1,1}$

Faits: $\neg P_{2,1}, \neg W_{2,1}, B_{2,1}, \neg S_{2,1}, \neg L_{2,1}$

Faits: $\neg P_{1,2}, \neg W_{1,2}, \neg B_{1,2}, S_{1,2}, \neg L_{1,2}$

R1-C1: $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$

R1-C2: $\neg P_{1,2} \vee B_{1,1}$

R1-C3: $\neg P_{2,1} \vee B_{1,1}$

R2-C1: $\neg S_{1,1} \vee W_{1,2} \vee W_{2,1}$

R2-C2: $\neg W_{1,2} \vee S_{1,1}$

R2-C3: $\neg W_{2,1} \vee S_{1,1}$

O: $W_{2,2}$

N-C1: $\neg W_{2,2}$

R3-C1: $\neg B_{2,1} \vee P_{1,1} \vee P_{3,1} \vee P_{2,2}$

R3-C2: $\neg P_{1,1} \vee B_{2,1}$

R3-C3: $\neg P_{3,1} \vee B_{2,1}$

R3-C4: $\neg P_{2,2} \vee B_{2,1}$

R4-C1: $\neg S_{2,1} \vee W_{1,1} \vee W_{3,1} \vee W_{2,2}$

R4-C2: $\neg W_{1,1} \vee S_{2,1}$

R4-C3: $\neg W_{3,1} \vee S_{2,1}$

R4-C4: $\neg W_{2,2} \vee S_{2,1}$



Base de connaissances

R5-C1: $\neg B_{1,2} \vee P_{1,1} \vee P_{1,3} \vee P_{2,2}$

R5-C2: $\neg P_{1,1} \vee B_{1,2}$

R5-C3: $\neg P_{1,3} \vee B_{1,2}$

R5-C4: $\neg P_{2,2} \vee B_{1,2}$

R6-C1: $\neg S_{1,2} \vee W_{1,1} \vee W_{1,3} \vee W_{2,2}$

R6-C2: $\neg W_{1,1} \vee S_{1,2}$

R6-C3: $\neg W_{1,3} \vee S_{1,2}$

R6-C4: $\neg W_{2,2} \vee S_{1,2}$

Requête ASK: pas de Wumpus en (2,2)

Formule à prouver: $f : \neg W_{2,2}$ (**observez ce qui est mis dans KB**)

Inférence 1 (R4-C4 avec un fait): $\frac{\neg W_{2,2} \vee S_{2,1} \quad \neg S_{2,1}}{\neg W_{2,2}}$

Inférence 2 (N-C1 avec O): $\frac{\neg W_{2,2} \quad W_{2,2}}{\emptyset}$

Conclusion: il n'y a pas de Wumpus en (2,2)

4				
3		 		
2				
1				
	1	2	X	3
				4

Le monde de Wumpus

Faits: $\neg P_{1,1}, \neg W_{1,1}, \neg B_{1,1}, \neg S_{1,1}, \neg L_{1,1}$

Faits: $\neg P_{2,1}, \neg W_{2,1}, B_{2,1}, \neg S_{2,1}, \neg L_{2,1}$

Faits: $\neg P_{1,2}, \neg W_{1,2}, \neg B_{1,2}, S_{1,2}, \neg L_{1,2}$

R1-C1: $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$

R1-C2: $\neg P_{1,2} \vee B_{1,1}$

R1-C3: $\neg P_{2,1} \vee B_{1,1}$

R2-C1: $\neg S_{1,1} \vee W_{1,2} \vee W_{2,1}$

R2-C2: $\neg W_{1,2} \vee S_{1,1}$

R2-C3: $\neg W_{2,1} \vee S_{1,1}$

O: $P_{2,2}$

N-C1: $\neg P_{2,2}$

R3-C1: $\neg B_{2,1} \vee P_{1,1} \vee P_{3,1} \vee P_{2,2}$

R3-C2: $\neg P_{1,1} \vee B_{2,1}$

R3-C3: $\neg P_{3,1} \vee B_{2,1}$

R3-C4: $\neg P_{2,2} \vee B_{2,1}$

R4-C1: $\neg S_{2,1} \vee W_{1,1} \vee W_{3,1} \vee W_{2,2}$

R4-C2: $\neg W_{1,1} \vee S_{2,1}$

R4-C3: $\neg W_{3,1} \vee S_{2,1}$

R4-C4: $\neg W_{2,2} \vee S_{2,1}$



Base de connaissances

R5-C1: $\neg B_{1,2} \vee P_{1,1} \vee P_{1,3} \vee P_{2,2}$

R5-C2: $\neg P_{1,1} \vee B_{1,2}$

R5-C3: $\neg P_{1,3} \vee B_{1,2}$

R5-C4: $\neg P_{2,2} \vee B_{1,2}$

R6-C1: $\neg S_{1,2} \vee W_{1,1} \vee W_{1,3} \vee W_{2,2}$

R6-C2: $\neg W_{1,1} \vee S_{1,2}$

R6-C3: $\neg W_{1,3} \vee S_{1,2}$

R6-C4: $\neg W_{2,2} \vee S_{1,2}$

Requête ASK: pas de trous en (2,2)

Formule à prouver: $f : \neg P_{2,2}$ (**observez ce qui est mis dans KB**)

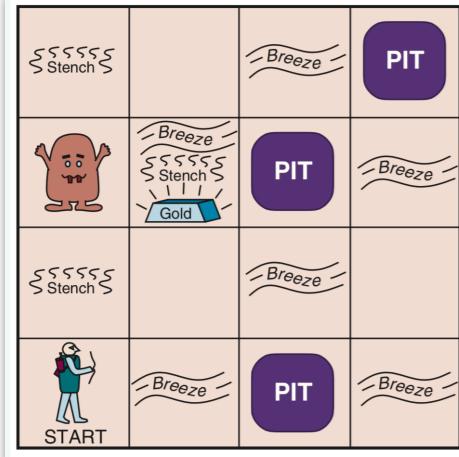
Inférence 1 (R5-C4 avec un fait):
$$\frac{\neg P_{2,2} \vee B_{1,2} \quad \neg B_{1,2}}{\neg P_{2,2}}$$

Inférence 2 (N-C1 avec O):
$$\frac{\neg P_{2,2} \quad P_{2,2}}{\emptyset}$$

Conclusion: il n'y a pas de trous en (2,2)

4				
3		 		
2				
1				
	1	2	3	4
	x			

Le monde de Wumpus



Certitude: on peut se déplacer de façon sécuritaire sur (2,2)

Ensuite: on obtient de la nouvelle information, et on recommence...

Au final: on sera capable d'atteindre l'or qu'avec des raisonnements logiques

? Quelles sont les difficultés de notre approche ?

Difficulté 1: manque de concision

Taille du problème: on a énormément de symboles et de formules dans notre base de connaissances

Mise à l'échelle: le nombre de formules grandit très vite avec l'acquisition de nouvelles connaissances

Observation: un grand nombre de formules ont une structure très similaire

$$\text{R1: } B_{1,1} \iff (P_{1,2} \vee P_{2,1})$$

$$\text{R2: } B_{2,1} \iff (P_{1,1} \vee P_{3,1} \vee P_{2,2})$$

$$\text{R3: } B_{1,2} \iff (P_{1,1} \vee P_{1,3} \vee P_{2,2})$$

$$\text{R4: } S_{1,1} \iff (W_{1,2} \vee W_{2,1})$$

$$\text{R5: } S_{2,1} \iff (W_{1,1} \vee W_{3,1} \vee W_{2,2})$$

$$\text{R6: } S_{1,2} \iff (W_{1,1} \vee W_{1,3} \vee W_{2,2})$$

Mitigation possible: utiliser un système logique plus expressif (mais plus complexe/coûteux)

Difficulté 2: complexité exponentielle de l'algorithme

Conséquence: les preuves de conséquence logique peuvent être très coûteuses à réaliser

Mitigation possible: utiliser un système logique moins complexe (mais moins expressif)

Table des matières

Agents logiques

✓ 1. Motivation et définition d'un agent logique

✓ 2. Représentation des connaissances

✓ 3. Système logique (syntaxe, sémantique, et règles d'inférences)

✓ 4. Formalisation de la logique des propositions

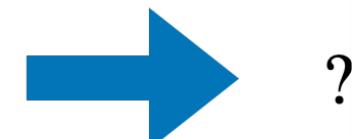
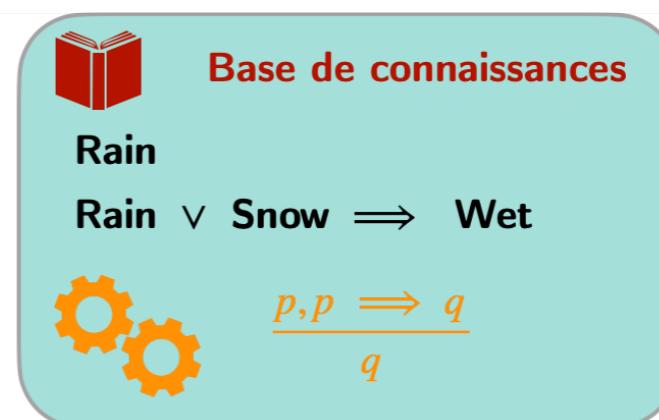
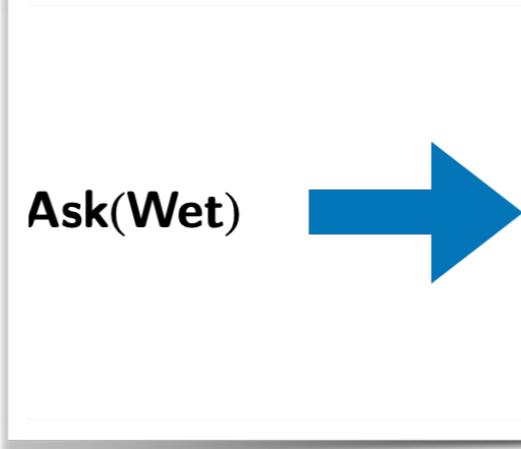
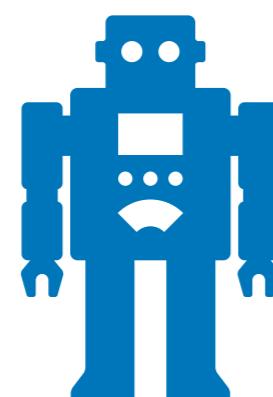
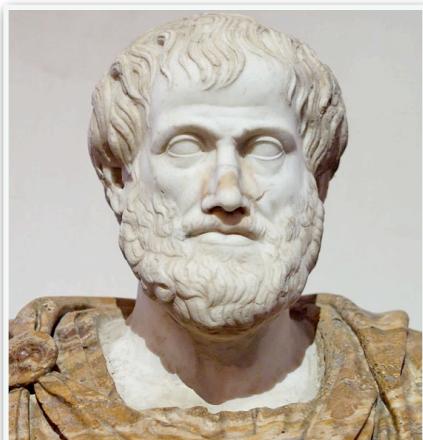
✓ 5. Raisonnement par model-checking

✓ 6. Inférences logiques et propriétés

✓ 7. Algorithme de résolution

8. Clauses de Horn

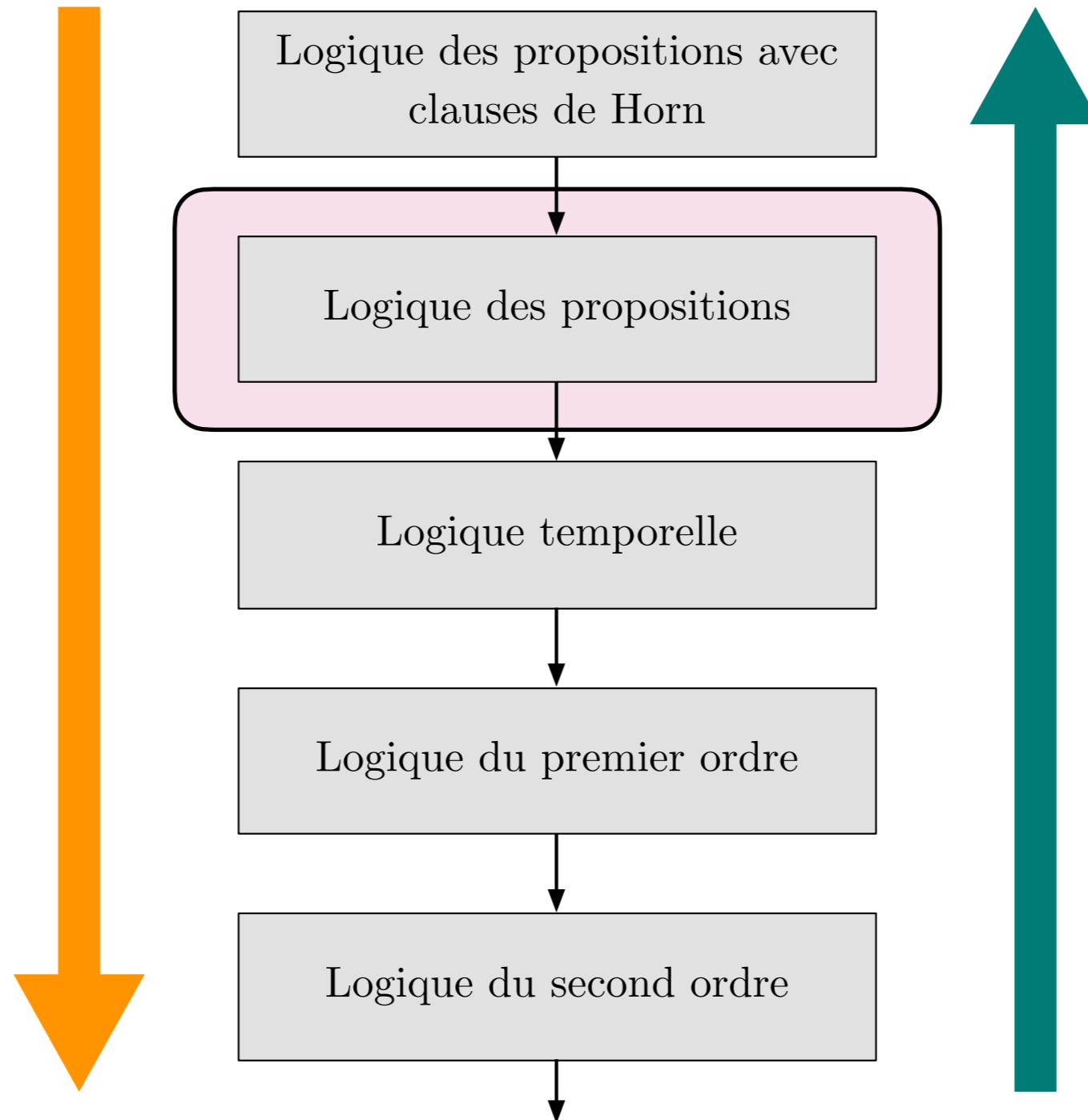
9. Extension à d'autres systèmes logiques



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

Logique des propositions avec l'algorithme de résolution

Toutes les formules propositionnelles



Gain en expressivité

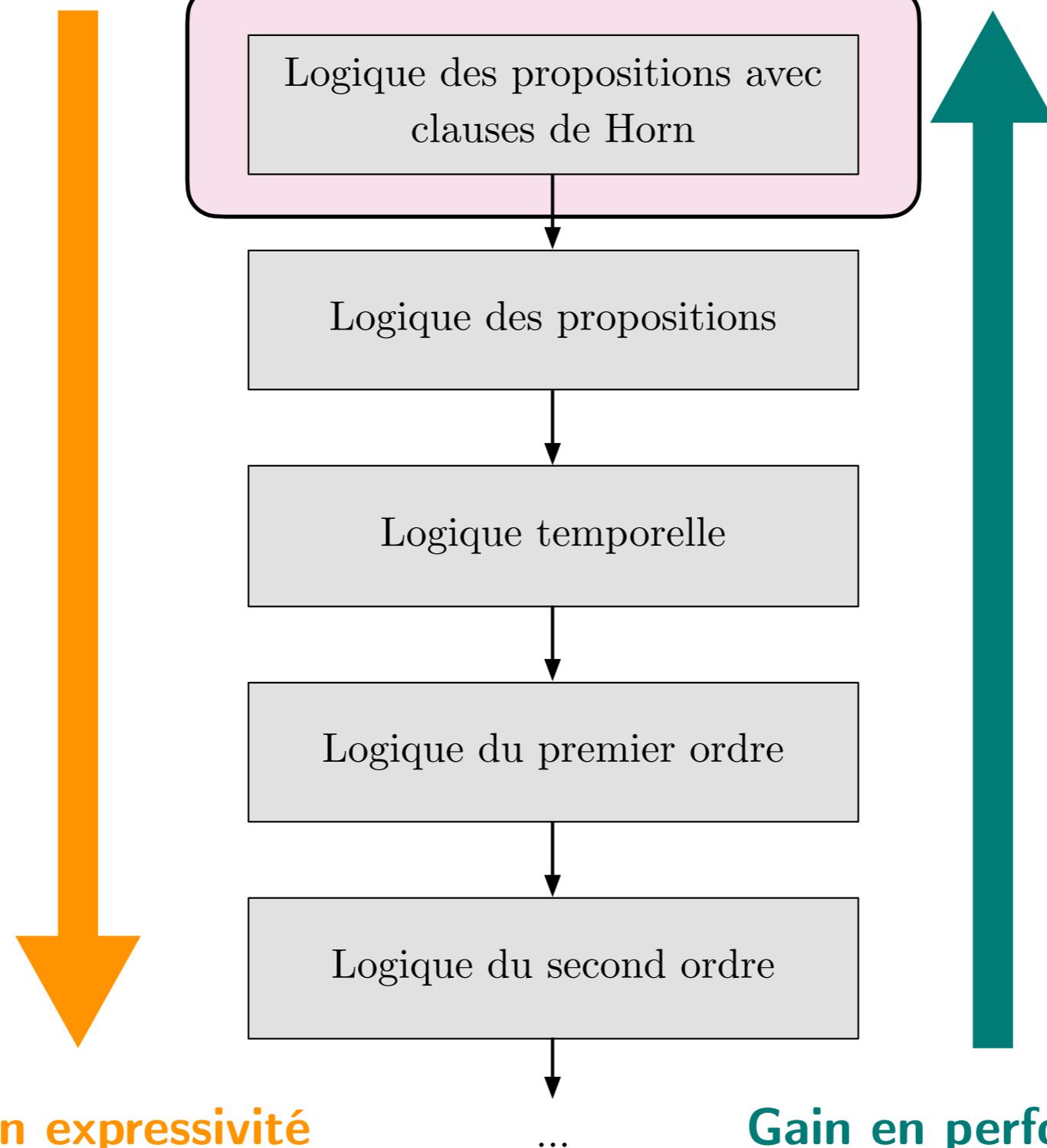
...

Gain en performance

Algorithme de résolution
Cohérent
Complet
Complexité exponentielle

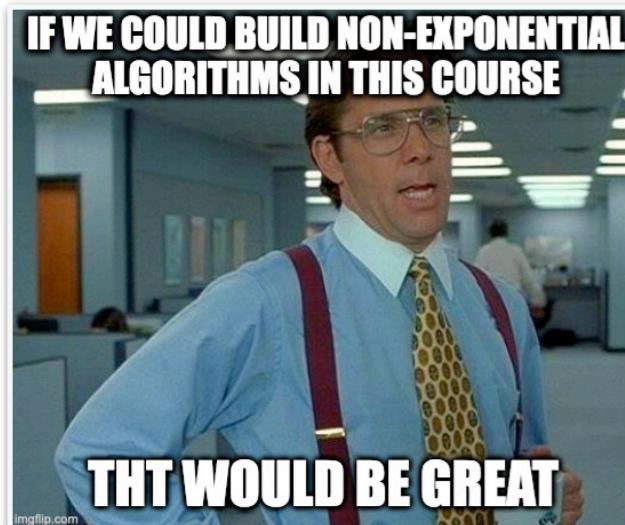
Logique des propositions avec clauses de Horn

Seulement les clauses de Horn



Inférence logique (MP)
Cohérent
Complet
Complexité linéaire

Logique des propositions avec clauses de Horn



Algorithme de résolution: génération d'un nombre exponentiel de clauses

En théorie: c'est le cas de tout algorithme cohérent et complet (si $P \neq NP$)

Astuce: c'est vrai tant qu'on reste dans la logique des propositions !



Peut-on considérer une autre logique avec de meilleures propriétés ?

Cas 1: une logique plus expressive sera également plus coûteuse au niveau des inférences

Cas 2: à l'inverse, une logique moins expressive peut être moins coûteuse à raisonner



Logique des propositions avec clauses de Horn

Forme simplifiée de la logique des propositions qui contient que des clauses de Horn



Clause de Horn: type particulier de clauses (définie aux diapositives suivantes)

Intérêt: on peut concevoir des algorithmes d'inférence à complexité temporelle linéaire !

Intérêt: ces algorithmes peuvent à la fois être cohérents et complets !

Prix à payer: on est limité aux clauses de Horn pour représenter nos connaissances

A. Horn (1918-2001)

Cette logique sacrifie un peu d'expressivité pour améliorer le processus d'inférence

Definite clauses et goal clauses



Definite clause (Clause définie)

Clause (disjonction de littéraux) ayant **exactement** un littéral qui est positif

$$f: P \vee \neg Q_1 \vee \neg Q_2 \vee \dots \vee \neg Q_n$$

$$\frac{\neg f_1 \vee \dots \vee \neg f_n}{\neg(f_1 \wedge \dots \wedge f_n)}$$

S'exprime également sous la forme suivante

$$f: (Q_1 \wedge Q_2 \wedge \dots \wedge Q_n) \Rightarrow P$$

$$\frac{(\neg f \vee g)}{(f \Rightarrow g)}$$

Une clause constituée d'un seul littéral est appelé un **fait (fact)**

$$f: R$$

Note: la deuxième expression est obtenue via l'application de deux règles

Objectif de la deuxième expression: mettre en évidence une implication entre deux formules

Interprétation: si Q_1, \dots, Q_n sont tous vraies, alors P l'est également



Goal clause (Clause de but)

Clause (disjonction de littéraux) n'ayant **aucun** littéral positif

$$f: \neg Q_1 \vee \neg Q_2 \vee \dots \vee \neg Q_n$$

S'exprime également sous la forme suivante

$$f: (Q_1 \wedge Q_2 \wedge \dots \wedge Q_n) \Rightarrow \text{false}$$

Objectif: mettre en évidence que la formule est vraie si au moins un symbole est faux

Clauses de Horn



Clauses de Horn

Clauses (disjonction de littéraux) ayant au plus un littéral qui est positif:

(1) Soit une *definite clause* $f: (Q_1 \wedge Q_2 \wedge \dots \wedge Q_n) \Rightarrow P$

(2) Soit une *goal clause* $f: (Q_1 \wedge Q_2 \wedge \dots \wedge Q_n) \Rightarrow \text{false}$

Logique des propositions avec clauses de Horn: système logique qui ne contient que des clauses de Horn

Conséquence: on a un système plus restrictif que la logique des propositions standard



Quel est l'avantage de cette restriction ?

Intérêt: le Modus Ponens est suffisant pour construire des algorithmes d'inférence cohérent et complet

$$\text{MP généralisé : } \frac{p_1, \dots, p_n \quad (p_1 \wedge \dots \wedge p_n) \Rightarrow q}{q}$$

$$\begin{array}{ccc} \text{Rain} & \text{Snow} & (\text{Rain} \wedge \text{Snow}) \Rightarrow \text{Dangerous} \\ & & \hline & & \text{Dangerous} \end{array}$$

Si il pleut et neige actuellement, et si je sais que la pluie avec de la neige sont dangereux, alors la situation actuelle est dangereuse

Intuition: on observe que les clauses de Horn correspondent au type de formules attendues dans le MP

Complexité de ces algorithmes: complexité linéaire en fonction de la taille de la base de connaissances

Exemples: algorithme du chaînage avant et du chaînage arrière (lectures complémentaires)

Table des matières

Agents logiques

✓ 1. Motivation et définition d'un agent logique

✓ 2. Représentation des connaissances

✓ 3. Système logique (syntaxe, sémantique, et règles d'inférences)

✓ 4. Formalisation de la logique des propositions

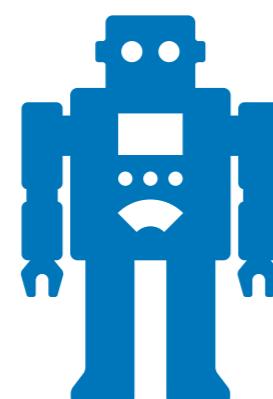
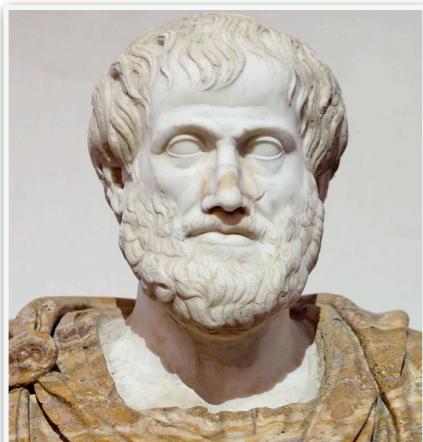
✓ 5. Raisonnement par model-checking

✓ 6. Inférences logiques et propriétés

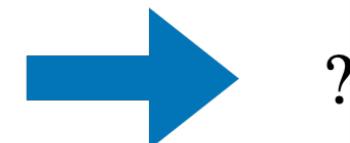
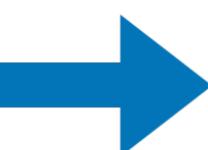
✓ 7. Algorithme de résolution

✓ 8. Clauses de Horn

9. Extension à d'autres systèmes logiques



Ask(Wet)



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

Etat actuel de nos logiques

Seulement les clauses de Horn

Toutes les formules propositionnelles



Logique des propositions avec clauses de Horn

Logique des propositions

Logique temporelle

Logique du premier ordre

Logique du second ordre

...

Gain en expressivité

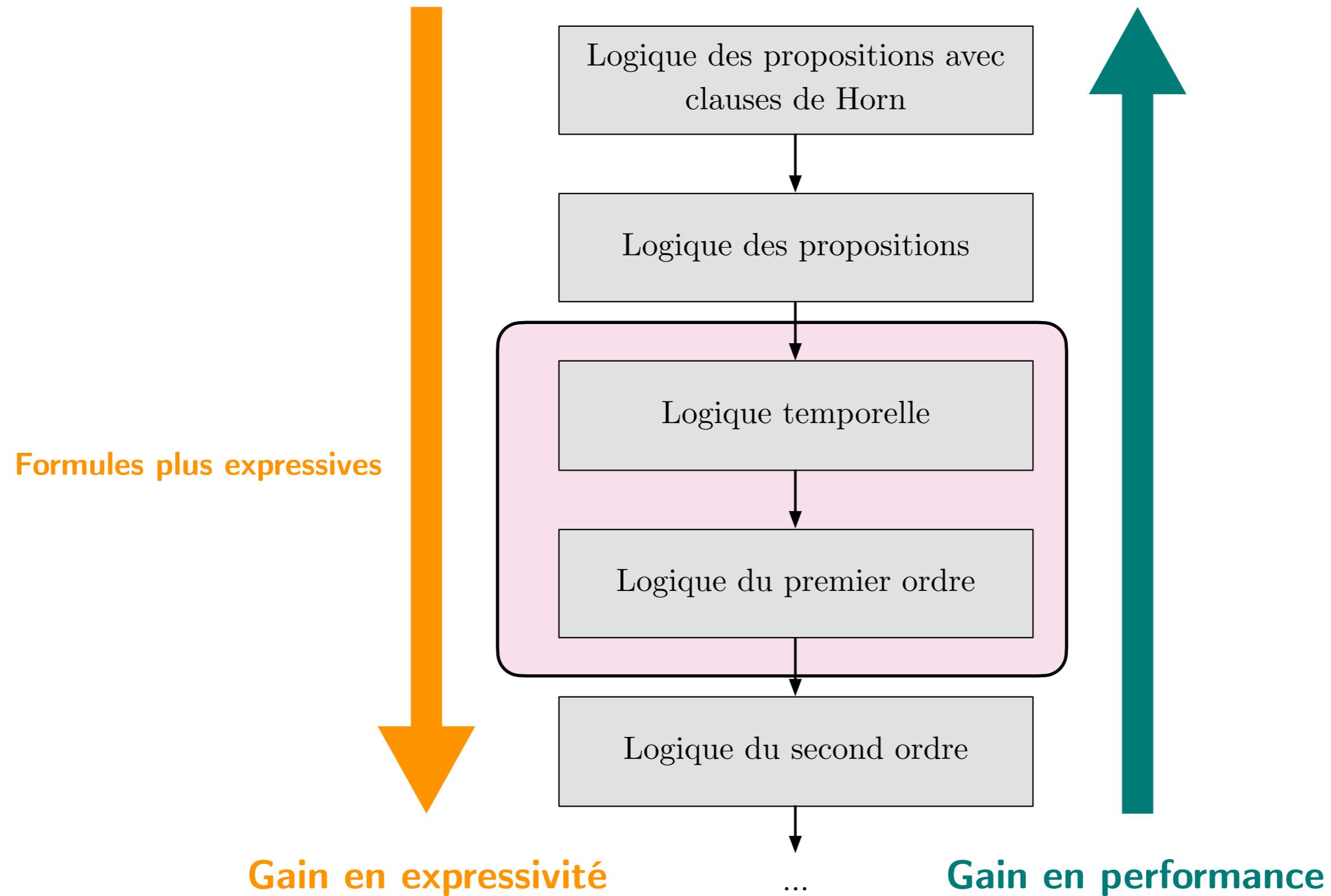
Gain en performance



Inférence logique (MP)
Cohérent et complet
Complexité linéaire

Algorithme de résolution
Cohérent et complet
Complexité exponentielle

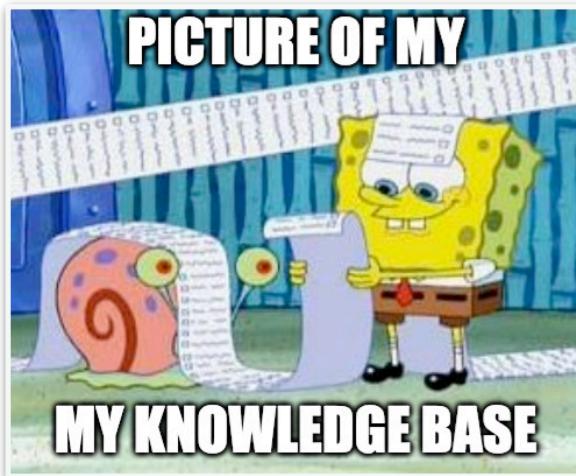
Logiques plus expressives



Limitation de l'expressivité de la logique des propositions



Quelles sont les limitations de la logique des propositions ?



Difficulté 1: manque de concision

Observation: certaines connaissances peuvent être très verbeuses

Exemple 1: notre situation du monde de Wumpus

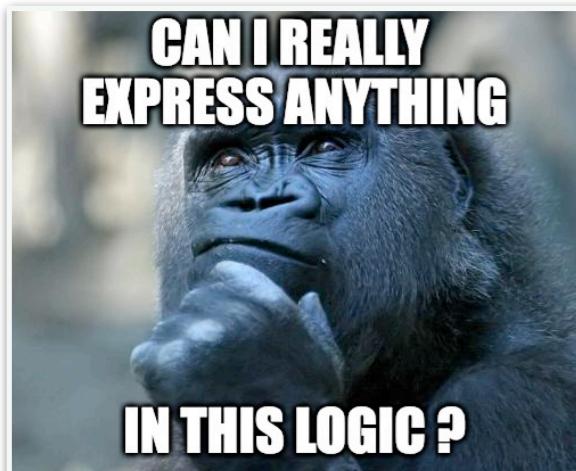
Exemple 2: tous les étudiants qui suivent INF8175 connaissent Python

$(\text{QuentinIsStudent} \wedge \text{QuentinFollowsINF8175}) \Rightarrow \text{QuentinKnowsPython}$

$(\text{AliceIsStudent} \wedge \text{AliceFollowsINF8175}) \Rightarrow \text{AliceKnowsPython}$

$(\text{SimonIsStudent} \wedge \text{SimonFollowsINF8175}) \Rightarrow \text{SimonKnowsPython}$

... (pour tous les étudiants)



Difficulté 2: des connaissances sont inexprimables par la logique des propositions

Ensembles non-fini: "*Tous les nombres pairs sont divisibles par deux*"

Connaissances temporelles: "*A 17h, il y a toujours du trafic à Montréal*"

Connaissances floues: "*La ville de Montréal est grande*" (*grande* est un terme vague)

Connaissances incertaines: "*Il y a 50% de chance qu'il pleuve demain* »

Logique du premier ordre (ou logique des prédictats - *first-order logic*)

Extension de la logique des prop. avec **des objets**, **des relations**, **des fonctions**, et **2 quantificateurs**

Objets: *Quentin, maison, nombres, étudiants, Montréal, jaune, voiture, etc.*

Relations entre objets: *est le frère de, se produit après, a la couleur, est inscrit à, etc.*

Fonctions (relation avec une seule valeur par input): *est le père de, est la capitale de, etc.*

Quantificateur universel: "pour tout"



Quantificateur existentiel: "il existe"

L'entièreté des éléments de la logique des propositions est également disponible

Exemple 1: *tous les étudiants qui suivent INF8175 connaissent Python*

$$f : \forall x \left((\text{Student}(x) \wedge \text{FollowsINF8175}(x)) \implies \text{KnowsPython}(x) \right)$$

Exemple 2: *il existe au moins un étudiant qui comprend la physique quantique*

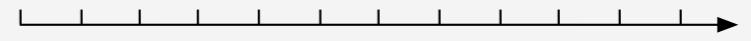
$$f : \exists x \left(\text{Student}(x) \implies \text{UnderstandsQuantumPhysics}(x) \right)$$

Exemple 3: *tous les nombres pairs sont divisibles par 2*

$$f : \forall x \left((\text{Number}(x) \wedge \text{Even}(x)) \implies \text{DivisibleBy2}(x) \right)$$

Logique temporelle (*temporal logic*)

Extension de la logique des prop. avec des **opérateurs temporels**



Suivant (next - $\bigcirc f$): la formule f doit tenir à l'instant suivant

Globallement (globally - $\Box f$): la formule f doit toujours tenir dans le futur



Finalement (finally - $\lozenge f$): la formule f doit devenir vraie au moins une fois dans le futur

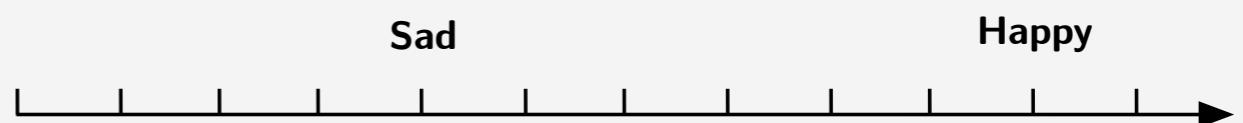
Jusqu'à ce que (until - $f \mathcal{U} g$): la formule f doit être vraie jusqu'à ce que g soit vraie

Représentation du temps: discréétisation en une séquence d'instant

Particularité: la valeur de vérité d'une formule peut changer avec le temps

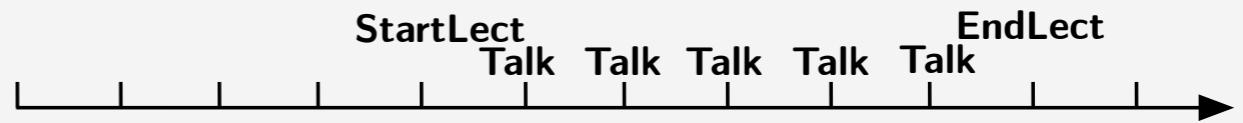
Exemple 1: quand on est triste, on devient heureux plus tard

$$f : \text{Sad} \implies \lozenge \text{Happy}$$



Exemple 2: quand je commence un cours, je parle jusqu'à ce que le cours soit fini

$$f : \text{StartLecture} \implies (\text{Talking} \mathcal{U} \text{EndLecture})$$



Exemple 3: en tout temps, si je n'ai pas de passeport ou de ticket, je ne peux pas monter dans un avion

$$f : \Box ((\neg \text{Passport} \vee \neg \text{BoardingPass}) \implies \bigcirc \neg \text{Boarding})$$

Logique floue (*fuzzy logic*)

Extension de la logique des propositions en rajoutant la notion de vérité partielle

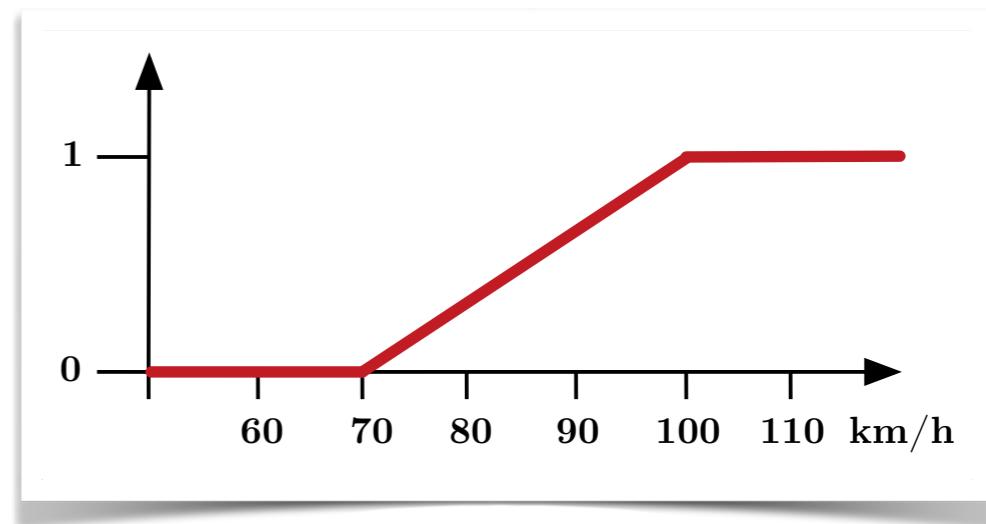
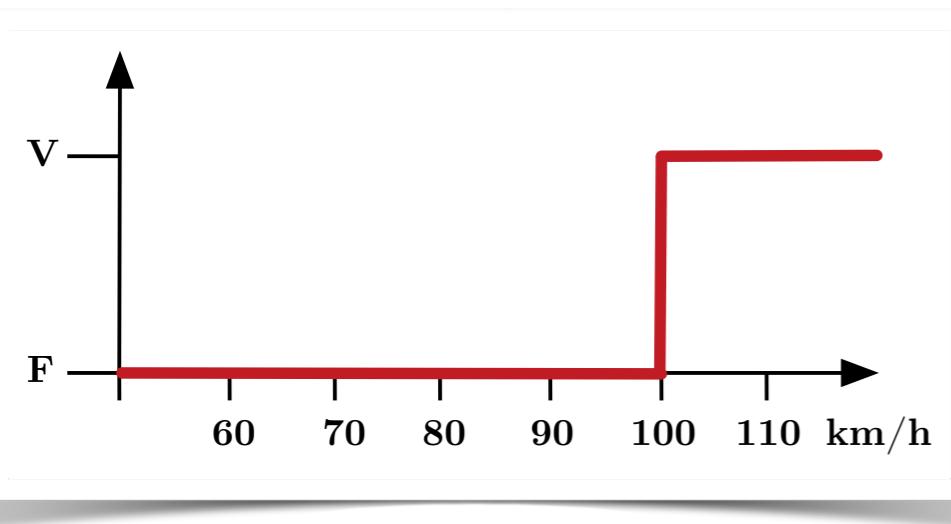
Idée: au lieu d'avoir une valeur sémantique vrai/faux, on considère un degré de confiance entre 0 et 1

Intérêt: permet d'exprimer des connaissances qui ne sont pas absolues mais relatives

Mise en situation: comment exprimer qu'une vitesse est *rapide* ?

Logique des propositions: fixer un seuil de séparation arbitraire (p.e., 100 km/h)

Logique floue: autoriser une transition progressive du faux vers le vrai



Exemple 1: Montréal est une grande ville

MontrealBig = 0.7 : indique que Montréal est grande avec une vérité relative de 0.7

Exemple 2: New York est une grande ville

NewYorkBig = 0.9 : indique que New-York est également grande, mais plus que Montréal

Logique probabiliste (*probabilistic reasoning*)

Extension de la logique des prop. avec des probabilités pour tenir compte de situations incertaines

Idée: au lieu d'avoir une valeur binaire pour chaque modèle, on a une probabilité d'occurrence

Difficulté: demande de connaître les probabilités d'occurrence d'évènements (ou de savoir les estimer)

Logique des propositions: $f = \text{Raining} \vee \text{Wet}$

		Wet	
		0	1
Raining	0		
	1		

		Wet	
		0	1
Raining	0	.1	.2
	1	.2	.5

		Wet	
		0	1
Raining	0	.1	.2
	1	.2	.5

		Wet	
		0	1
Raining	0	.1	.2
	1	.2	.5

		Wet	
		0	1
Raining	0	.1	.2
	1	.2	.5

Logique probabiliste: intégration de probabilités

Attention: notez la différence avec la logique floue où les connaissances sont connues mais non absolues

Exemple 1: quelle est la probabilité qu'il pleuve ?

$$P(\text{Raining}) = 0.2 + 0.5 = 0.7$$

Exemple 2: quelle est la probabilité qu'il pleuve et que le sol soit mouillé ?

$$P(\text{Raining} \wedge \text{Wet}) = 0.5$$

Exemple 3: quelle est la probabilité qu'il pleuve sans que le sol soit mouillé ?

$$P(\text{Raining} \wedge \neg \text{Wet}) = 0.2$$

Table des matières

Agents logiques

✓ 1. Motivation et définition d'un agent logique

✓ 2. Représentation des connaissances

✓ 3. Système logique (syntaxe, sémantique, et règles d'inférences)

✓ 4. Formalisation de la logique des propositions

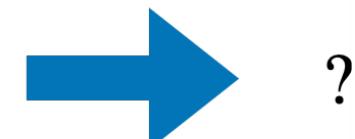
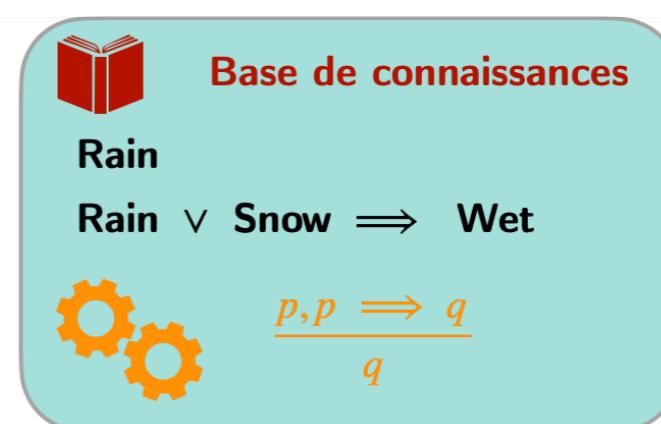
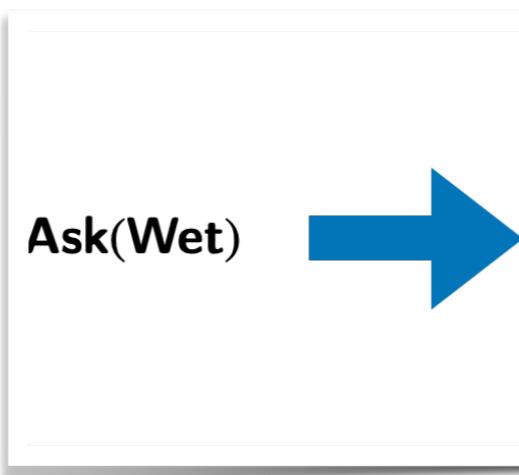
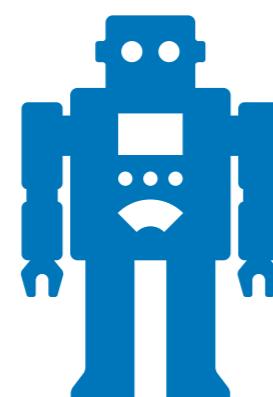
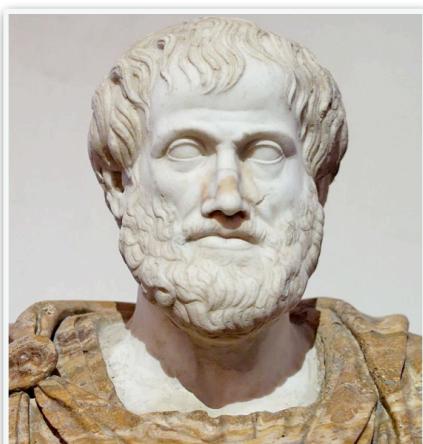
✓ 5. Raisonnement par model-checking

✓ 6. Inférences logiques et propriétés

✓ 7. Algorithme de résolution

✓ 8. Clauses de Horn

9. Extension à d'autres systèmes logiques



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

Synthèse des notions vues

Langage logique

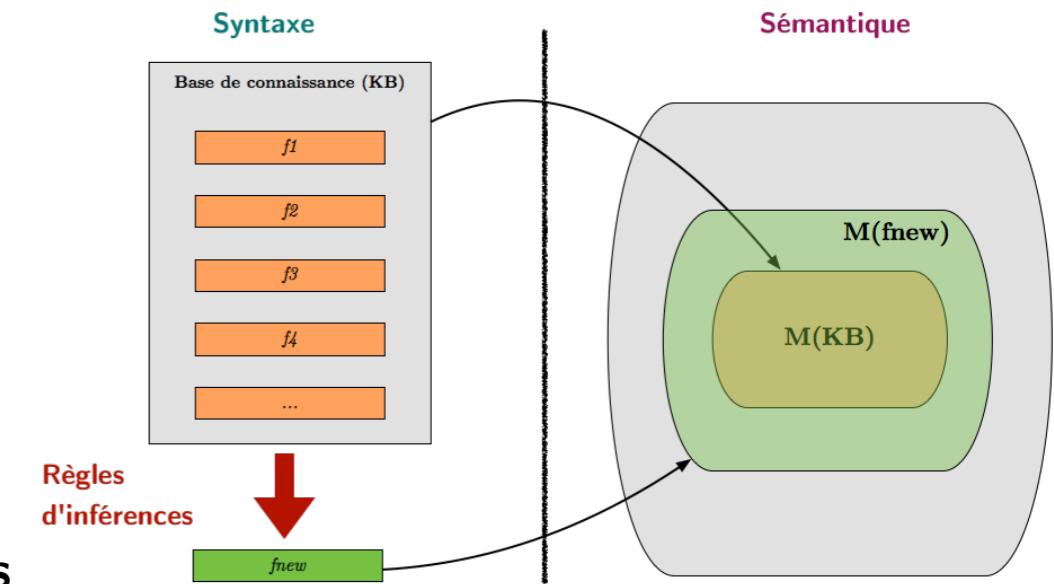
Objectif: raisonner sur base d'un ensemble de connaissances

Composé de 3 éléments fondamentaux

Syntaxe: définit un ensemble de formules valides

Sémantique: définit la signification d'une formule

Règles d'inférence: façon d'obtenir de nouvelles connaissances

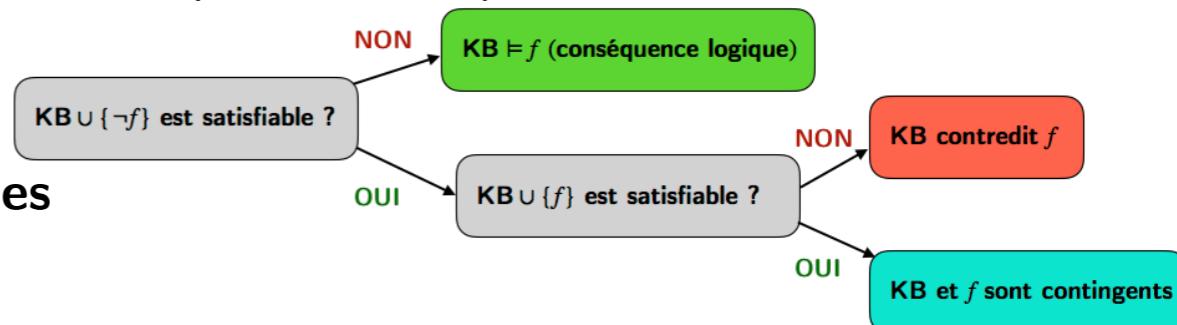


Vérification par model checking

Vérification de la satisfiabilité d'une formule basée sur les modèles (sémantique)

Résolution: via des méthodes de recherche

Limitation: ne tient pas compte qu'on a des formules logiques



Algorithmes d'inférences

Algorithme utilisé pour inférer des nouvelles connaissances en conséquence logique de KB

Cohérence: assure que l'algorithme ne génère que des formules vraies

Complétude: assure que l'algorithme génère toutes les formules vraies existantes

Algorithme de résolution: cohérent et complet dans la logique des propositions



Aller plus loin en programmation logique



Bonne nouvelle: les notions vues sont valides pour d'autres systèmes logiques

Différence: au niveau de la syntaxe, sémantique, et règles d'inférence

Autres logiques: premier ordre, temporelle, floue, probabiliste, etc.

En fonction de vos besoins, vous pouvez sélectionner la logique la plus appropriée



Implémentation: langages déclaratifs permettant d'utiliser des langages logiques

Prolog: le plus populaire, réalisant des inférences logiques sur des clauses de Horn

NuSMV: langage permettant une résolution basée sur du *model checking*

Minizinc: propose une interface avec le solveur Chuffed (apprentissage des clauses)



INF8225: I.A.: techniques probabilistes et d'apprentissage (Christopher Pal)

Cours consacré en partie au raisonnement probabiliste

Recherche scientifique: paradigme de l'intelligence *neuro-symbolique*

Principe: améliorer l'apprentissage profond avec des raisonnements logiques

Exemples de questions d'examen

Théorie

1. Expliquer ce qu'est un langage logique et ses différentes composantes
2. Comprendre le fonctionnement de la logique des propositions
3. Comprendre les notions de cohérence et de complétude pour un algorithme d'inférence

Pratique

1. Savoir comparer différents algorithmes dédiés à la logique (model checking, résolution, etc.)
2. Savoir appliquer l'algorithme de résolution
3. Savoir indiquer si un algorithme d'inférence est cohérent et/ou complet





DALLE: *The world of wumpus, painted by Vermeer*