

Bases de l'IA

**Programmation logique et systèmes de
règles**

Elena CABRIO

elena.cabrio@univ-cotedazur.fr

Plan pour cette séance

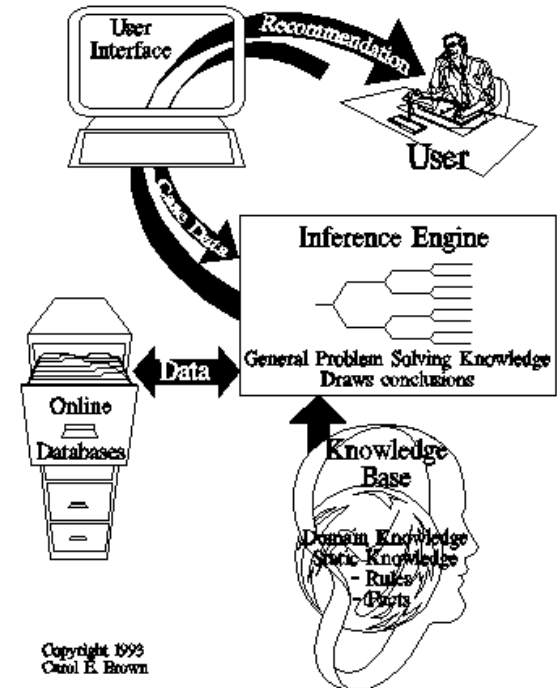
- Systèmes experts
- Quelques notions de logique
- Chaînage avant
- Chaînage arrière
- Programmation logique

Système expert

- Un système qui peut aider à résoudre
 - Des problèmes complexes du monde réel
 - Dans des domaines spécifiques
 - Science
 - Ingénierie
 - Médecine
- Utilisation de connaissances du domaine
 - Faits et procédures glanés d'experts humains
 - Utiles pour résoudre des problèmes typiques d'un domaine
- Très à la mode dans les années 1970 et 1980
- Aujourd'hui on parle plutôt de **systèmes d'aide à la décision**

Architecture d'un système expert

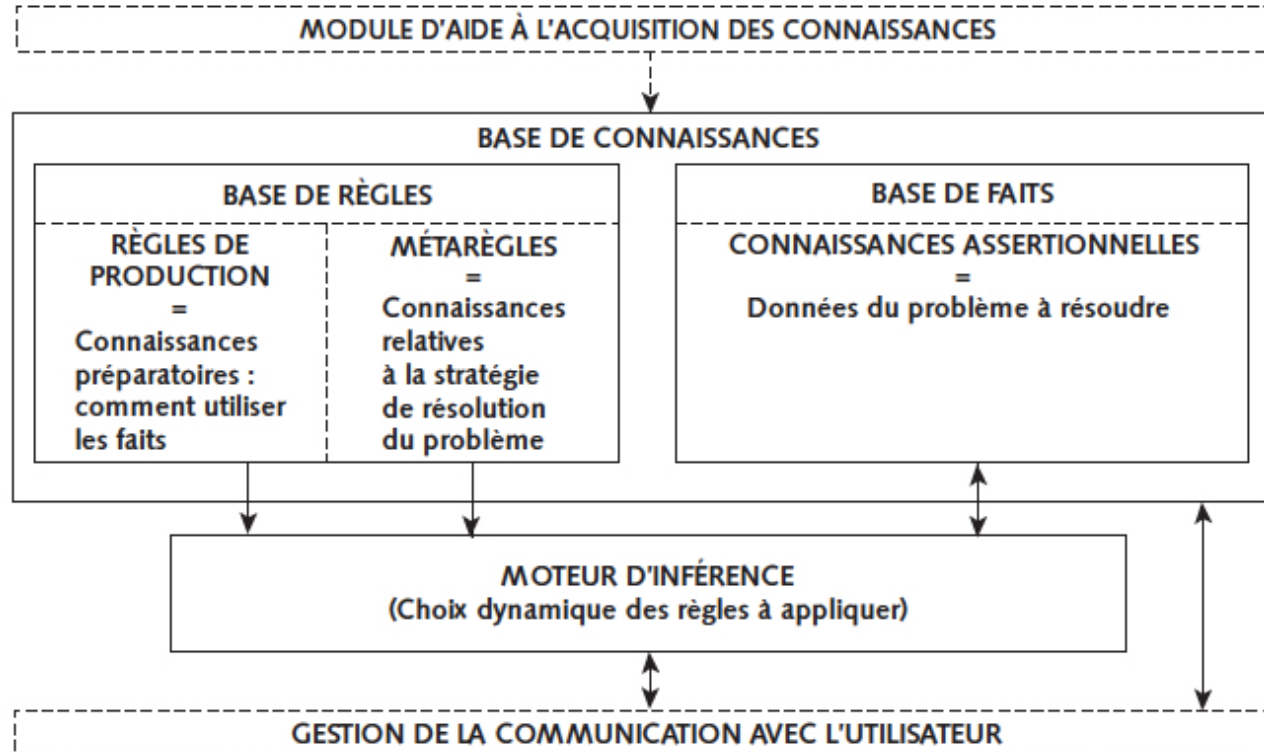
- Un système expert est une application qui simule l'intelligence et le comportement humain dans un domaine spécifique et limité
- Il se compose de trois modules principaux :
 - Une base de connaissances
 - Un moteur d'inférence
 - Une interface utilisateur



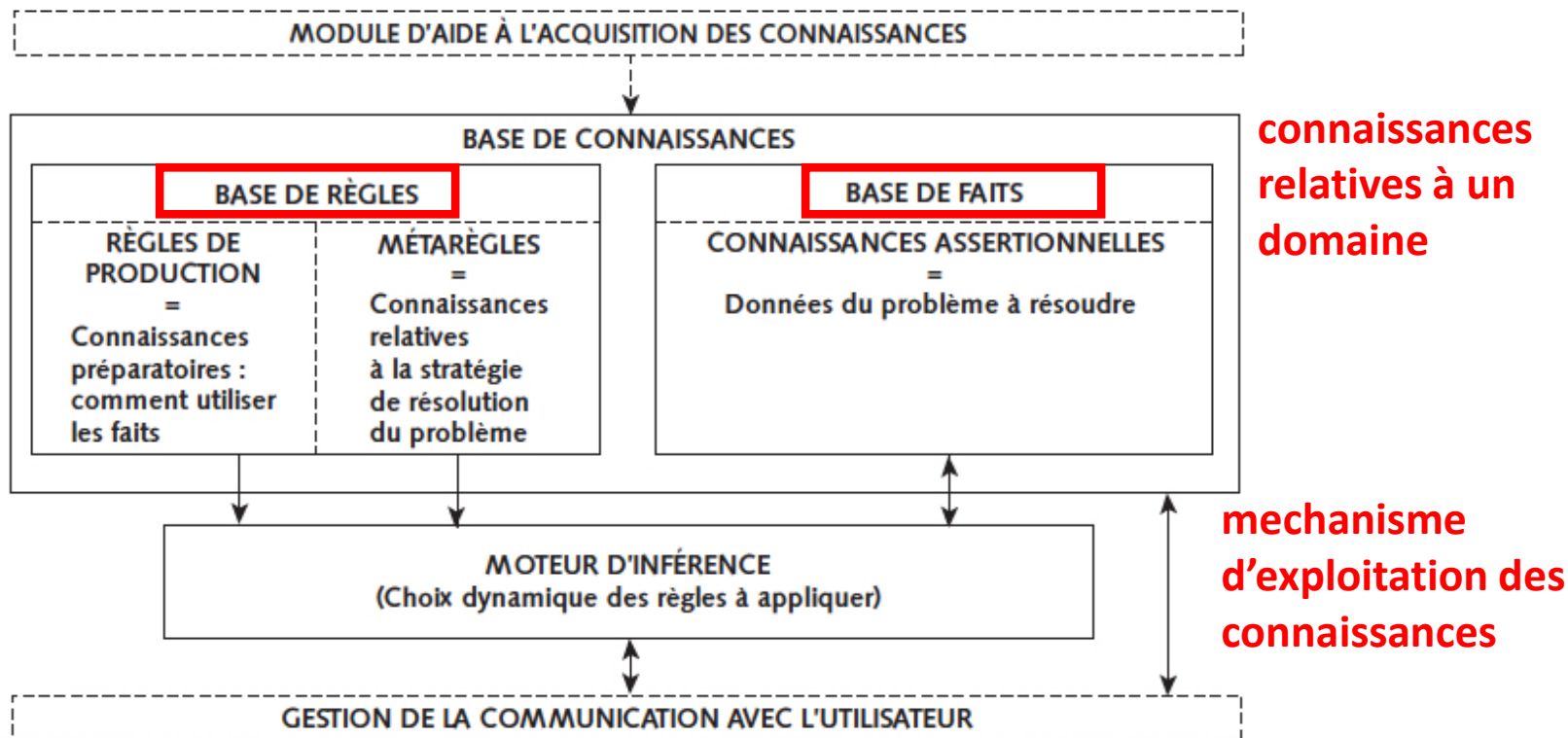
Exemples d'applications

- Chimie : détermination de structure, synthèse
- Maths : résolution de problèmes
- Géologie : prospection
- Médecine :
 - Consultation clinique / aide à la décision
 - Diagnose à partir d'imagerie médicale
- Finance : planification stratégique, trading
- Commerce : marketing, recommandation

Structure d'un système expert

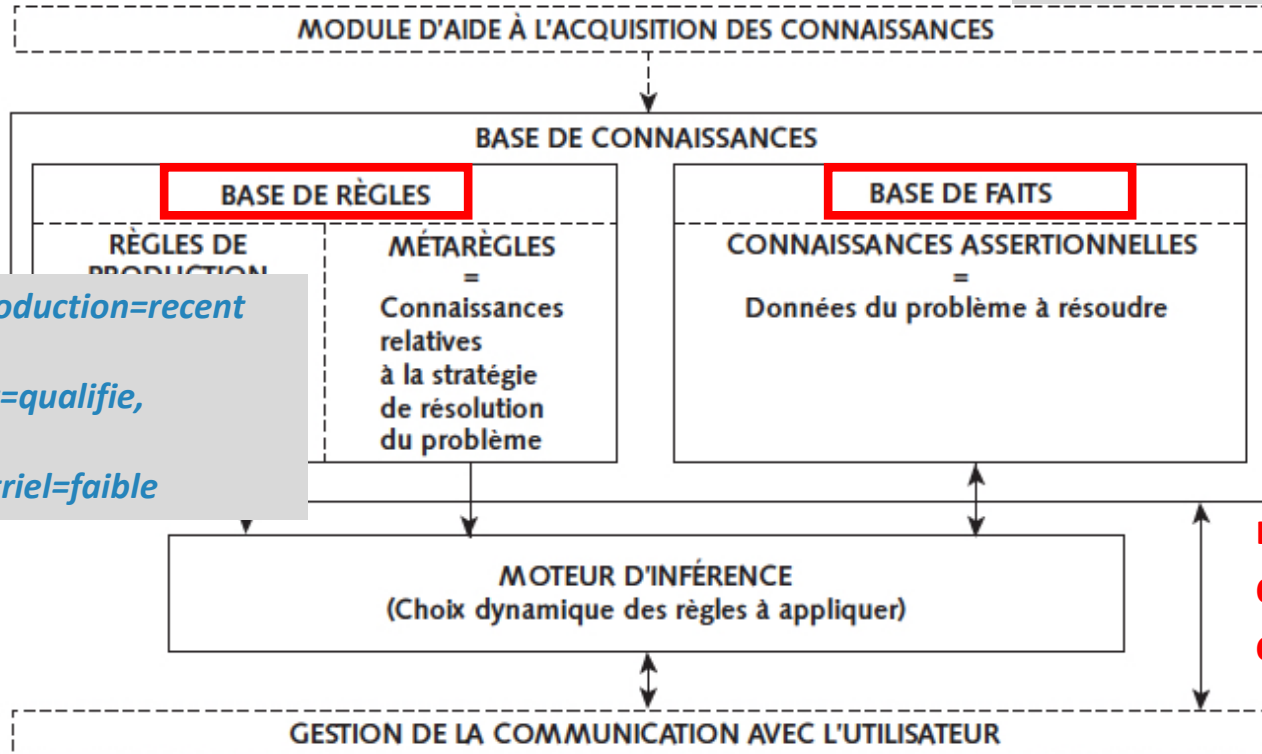


Structure d'un système expert



Structure d'un système expert

Problème de
diagnostic
d'entreprise



connaissances
relatives à un
domaine

- Statut juridique:
SARL
- Chiffre d'affaires
annuels:...

mechanisme
d'exploitation des
connaissances

Si outil de production=recent
ET
sous-traitant=qualifié,
alors
risque-industriel=faible

Points forts et faibles

- **Le systèmes experts sont bons pour :**
 - Des domaines limités où des connaissances expert sont disponibles
 - Fournir un avis expert dans des sites distants
 - Améliorer les performances en appliquant des heuristiques suggérées per des experts
 - Planification, diagnostic, manipulation robotisée, exploration, commande
- **Ils ne sont pas adaptés pour :**
 - Représenter des connaissances temporelles
 - Représenter des connaissances spatiales
 - Faire du raisonnement « de bon sens »
 - Reconnaître leurs limites
 - Exploiter des connaissances incohérentes (c-à-d contradictoires)

Points forts et faibles

- **Le systèmes experts sont bons pour :**
 - Des domaines limités où des connaissances expert sont disponibles
 - Fournir un avis expert dans des sites distants
 - Améliorer les performances en appliquant des heuristiques suggérées per des experts
 - Planification, diagnostic, manipulation robotisée, exploration, commande
- **Ils ne sont pas adaptés pour :**
 - Représenter des connaissances temporelles
 - Représenter des connaissances spatiales
 - Faire du raisonnement « de bon sens »
 - Reconnaître leurs limites
 - Exploiter des connaissances incohérentes (c-à-d contradictoires)

“Il est plus facile de simuler un geologue qu’un enfant de 5 ans...”

- **Élément critique d'un système expert**
- Nature (déclarative, procédurale, implicite, ...)
- Représentation (voir aussi prochaine séance)
- Phénomène essentiellement social
 - Partage
 - Transfert humain → machine : *extraction*
 - Transfert machine → humain : *explication*
- Ressource, avec une valeur économique

- Si on veut formaliser des connaissances et les utiliser pour faire des raisonnements, la **logique** semble être l'outil parfait
- La logique est l'étude
 - Du sens de ce que nous affirmons (sémantique)
 - De la façon dont nous raisonnons (ou devrions raisonner)
- Or, **l'une des caractéristiques de l'intelligence semble être la capacité de raisonner**
- Donc, si nous voulons construire des machines intelligentes, nous devons être capables d'automatiser le raisonnement

Qu'est-ce que la Logique ?

- La logique est l'étude des informations encodées sous forme de phrases (ou formules) logiques.
- Chaque phrase S divise l'ensemble des mondes possibles en l'ensemble des mondes dans lesquels S est vrai (modèles de S) et l'ensemble des mondes dans lesquels S est faux (contre-modèles de S)
- **Un ensemble de prémisses implique logiquement une conclusion \Leftrightarrow la conclusion est vraie dans tous les mondes où toutes les prémisses sont vraies**
- Une logique consiste en :
 - Un langage avec une syntaxe formelle et une sémantique précise
 - Une notion d'implication logique
 - Des règles de manipulation des expressions du langage

Logique Propositionnelle

- Une signature propositionnelle est un ensemble de symboles primitifs, appelés **constantes propositionnelles**.
- Une constante propositionnelle symbolise une phrase simple, comme
 - “il pleut” $\rightarrow p$
 - “le réservoir est vide” $\rightarrow v$
- Étant donnée une signature propositionnelle, une phrase propositionnelle est :
 - Soit un élément de sa signature
 - Soit une expression composée à partir d'éléments de sa signature. (voir prochaine diapo pour plus de détails)
- Un langage propositionnel est l'ensemble des phrases propositionnelles qui peuvent être formées à partir de sa signature.

Phrases Composées

- Négations : $\neg \textit{pluvieux}$
- Conjonction : $(\textit{pluvieux} \wedge \textit{neigeux})$
- Disjonction : $(\textit{pluvieux} \vee \textit{neigeux})$
- Implication : $(\textit{pluvieux} \Rightarrow \textit{nuageux})$
- L'argument de gauche d'une implication s'appelle l'*antécédent*.
- L'argument de droite s'appelle le *conséquent*.
- Équivalence : $(\textit{nuageux} \Leftrightarrow \textit{couvert})$

Interprétation propositionnelle

- Une interprétation propositionnelle est une fonction qui met en correspondance chaque constante propositionnelle dans un langage propositionnel avec les valeurs de vérité V ou F

$$\mathcal{I} : \text{Constantes} \rightarrow \{F, V\}$$

$$p \xrightarrow{\mathcal{I}} V$$

$$q \xrightarrow{\mathcal{I}} F$$

$$r \xrightarrow{\mathcal{I}} V$$

$$p^{\mathcal{I}} = V$$

$$q^{\mathcal{I}} = F$$

$$r^{\mathcal{I}} = V$$

- Nous considérons parfois une interprétation comme un vecteur booléen de valeurs pour les éléments de la signature de la langue (lorsque la signature est ordonnée) : VFV

Interprétation d'une phrase

- Une interprétation de phrase est une fonction qui met en correspondance chaque phrase propositionnelle avec les valeurs de vérité V or F.

$$p^{\mathcal{I}} = V$$

$$q^{\mathcal{I}} = F$$

$$r^{\mathcal{I}} = V$$

$$(p \vee q)^{\mathcal{I}} = V$$

$$(\neg q \vee r)^{\mathcal{I}} = V$$

$$((p \vee q) \wedge (\neg p \vee r))^{\mathcal{I}} = V$$

- Une interprétation propositionnelle définit une interprétation de phrase par application de la sémantique des opérateurs.

Sémantique des opérateurs

ϕ	$\neg\phi$
F	V
V	F

ϕ	ψ	$\phi \wedge \psi$
F	F	F
F	V	F
V	F	F
V	V	V

ϕ	ψ	$\phi \vee \psi$
F	F	F
F	V	V
V	F	V
V	V	V

ϕ	ψ	$\phi \Rightarrow \psi$
F	F	V
F	V	V
V	F	F
V	V	V

ϕ	ψ	$\phi \Leftrightarrow \psi$
F	F	V
F	V	F
V	F	F
V	V	V

Interprétations multiples

- La logique ne prescrit pas quelle interprétation est "correcte". En l'absence d'informations supplémentaires, une interprétation est aussi bonne qu'une autre.
- Exemples:
 - Des jours de la semaine différents
 - Des lieux différents
 - Opinions de gens différents
- Nous pouvons penser à une interprétation comme un *monde possible*
- L'ensemble de toute les interprétations (mondes possibles) est

$$\Omega = \{F, V\}^{\text{Constantes}}$$

Tables de vérité

- Une table de vérité est une table de toutes les interprétations possibles des constantes propositionnelles dans un langage (c'est-à-dire une représentation de Ω).

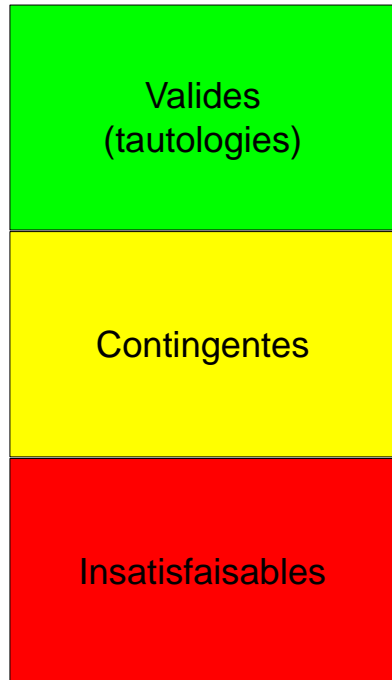
p	q	r
F	F	F
F	F	V
F	V	F
F	V	V
V	F	F
V	F	V
V	V	F
V	V	V

Une ligne par interprétation

Une colonne par constante

Pour un langage avec n constantes,
il y a 2^n interprétations

Propriétés des phrases

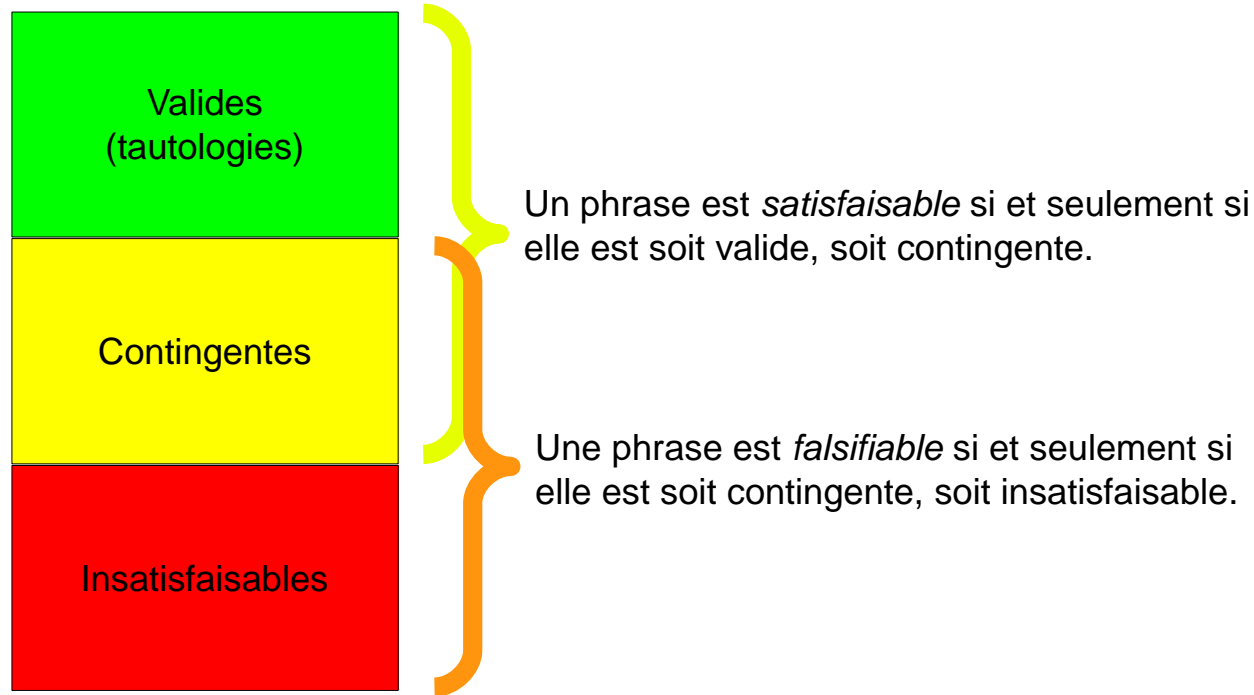


Une phrase est *valide* si et seulement si *toute* interprétation la satisfait.

Une phrase est *contingente* si et seulement si *au moins une* interprétation la satisfait et *au moins une* interprétation la falsifie.

Une phrase est *insatisfaisable* si et seulement si *aucune* interprétation ne la satisfait.

Propriétés des phrases



Exemple d'une tautologie

p	q	r	$p \Rightarrow q$	$q \Rightarrow r$	$(p \Rightarrow q) \vee (q \Rightarrow r)$
F	F	F			
F	F	V			
F	V	F			
F	V	V			
V	F	F			
V	F	V			
V	V	F			
V	V	V			

Exemple d'une tautologie

une proposition (ou énoncé)
 toujours vraie, c'est-à-dire
 vraie quelle que soit la valeur
 de vérité, vraie ou fausse, de
 ses constituants
 élémentaires

$\frac{p}{F}$	$\frac{p}{F}$	$\frac{p}{F}$			
F	F	F			
F	F	F			
F	F	F			
V	V	V			
V	V	V			
V	V	F			
V	V	V			



Exemple d'une tautologie

p	q	r	$p \Rightarrow q$	$q \Rightarrow r$	$(p \Rightarrow q) \vee (q \Rightarrow r)$
F	F	F	V	V	
F	F	V	V	V	
F	V	F	V	F	
F	V	V	V	V	
V	F	F	F	V	
V	F	V	F	V	
V	V	F	V	F	
V	V	V	V	V	

Exemple d'une tautologie

p	q	r	$p \Rightarrow q$	$q \Rightarrow r$	$(p \Rightarrow q) \vee (q \Rightarrow r)$
F	F	F	V	V	V
F	F	V	V	V	V
F	V	F	V	F	V
F	V	V	V	V	V
V	F	F	F	V	V
V	F	V	F	V	V
V	V	F	V	F	V
V	V	V	V	V	V

Quelques autres tautologies

Double Négation : $p \Leftrightarrow \neg\neg p$

Lois de Morgan : $\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$
 $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$

Introduction de l'implication : $p \Rightarrow (q \Rightarrow p)$

Distribution de l'implication :

$$(p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))$$

Implication logique

- Un ensemble de prémisses Δ implique logiquement une conclusion ϕ , écrit

$$\Delta \models \phi$$

- si et seulement si toute interprétation qui satisfait les prémisses satisfait aussi la conclusion.

- Exemples:

$$\{p\} \models p \vee q$$

$$\{p\} \not\models p \wedge q$$

$$\{p, q\} \models p \wedge q$$



Implication logique \neq Équivalence !

Méthode de la table de vérité

- Pour calculer **si un ensemble de prémisses entraîne logiquement une conclusion** :
 - 1) Former un tableau de vérité pour les constantes propositionnelles se produisant dans les prémisses et la conclusion ; ajouter une colonne pour les prémisses et une colonne pour la conclusion
 - 2) Évaluer les prémisses pour chaque ligne de la table
 - 3) Évaluer la conclusion pour chaque ligne de la table
 - 4) Si toute ligne qui satisfait les prémisses satisfait aussi la conclusion, alors les prémisses entraînent logiquement la conclusion

Implication logique et satisfaisabilité

- **Théorème :** $\Delta \models \phi$ si et seulement si $\Delta \cup \{\neg\phi\}$ est insatisfaisable.
- **Corollaire:** nous pouvons déterminer l'implication logique en déterminant la satisfaisabilité (preuve par réfutation).

De la logique propositionnelle à la logique des prédicats

- En logique propositionnelle nous pouvons exprimer cela :
 - Prémisses:
 - Si Jaques connaît Gilles, alors Gilles connaît Jacques ;
 - Jacques connaît Gilles.
 - Conclusion:
 - Gilles connaît Jacques
- Mais qu'en est-il de cela ?
 - Premises:
 - Si une personne en connaît une autre, alors cette autre personne connaît la première ;
 - Jacques connaît Gilles.
 - Conclusion: ...

Logique relationnelle

- Il nous faut des nouveaux ingrédients :
 - Variables
 - Quantificateurs
- Par exemple,

$$\forall x \forall y (\text{connaît}(x, y) \Rightarrow \text{connaît}(y, x))$$

Logique relationnelle : Syntaxe

- Objets constants (individus): Jacques, Nice, France, 0, 2345, 3.1415
- Relation constantes (prédicats): connaît, aime, identique
- Les prédicats ont une « arité »:
 - Unaire – 1 argument
 - Binaire – 2 arguments
 - Ternaire – 3 arguments
 - n -aire – n arguments
- Signature:
 - Ensemble d'objets constants
 - Ensemble de prédicats avec leur arité
- Variables: x , y , z , etc.

Termes et phrases

- Un terme est une variable ou un objet constant
- Les termes représentent des objets
- Ils sont l'équivalent des syntagmes nominaux en français
- Phrases
 - Relationnelles (atomes, \approx propositions simples):
 - Un prédicat d'arité n appliqué à n termes
 - Logiques (\approx proposition complexes):
 - Combinaisons de phrases liées par des opérateurs
 - Quantifiées :
 - Phrases avec variables quantifiées

Phrases

Phrase ::=

- Une constante relationnelle d'arité n appliquée à n termes.
- $(\neg \phi)$ où ϕ est une phrase.
- $(\phi \vee \psi)$, où ϕ et ψ sont des phrases.
- $(\phi \wedge \psi)$, où ϕ et ψ sont des phrases.
- $(\phi \Rightarrow \psi)$, où ϕ et ψ sont des phrases.
- $(\phi \Leftrightarrow \psi)$, où ϕ et ψ sont des phrases.
- $(\forall x \phi)$, où ϕ est une phrase.
- $(\exists x \phi)$, où ϕ est une phrase.

Seules les expressions produites par les règles ci-dessus sont des phrases.

Logique des prédicats (avec fonctions)

- La syntaxe est une extension de la syntaxe de la logique relationnelle
- Constantes fonction avec leur arité : $f(.)$, $g(., .)$, etc.
- La définition d'un terme devient :
 - Une variable
 - Une constante objet
 - Une constante fonction d'arité n appliquée à n termes.
- Seules les expressions produites par les règles ci-dessus sont des termes.
- Par conséquent, l'ensemble des termes sera infini, même si le vocabulaire du langage est fini.

Règles d'inférence

- Un *schéma* est une expression qui suit la grammaire de notre langage, mais qui en outre utilise des *méta-variables* (par exemple écrites comme des lettres grèques) à la place de certaines parties d'une expression.

- Exemple:

$$\phi \Rightarrow \psi$$

- Une règle d'inférence s'écrit :

Prémisses
<hr/>
Conclusions

Modus Ponens

$$\frac{\phi \Rightarrow \psi \quad \phi}{\psi}$$

Décidabilité et complexité

- La logique des prédicats est trop puissante
- Une théorie en logique des prédicats peut être indécidable (Théorème de Gödel)
- **Même si la logique propositionnelle est décidable, le problème de décider si un ensemble de phrases est satisfaisable est NP-complet.**
- Il nous faut des fragments plus maniables

Clauses de Horn

$$\neg b_1 \vee \neg b_2 \vee \dots \vee \neg b_n \vee h \quad \text{c-à-d} \quad (b_1 \wedge b_2 \wedge \dots \wedge b_n) \Rightarrow h$$

(stricte)

h (positive)

$$\neg b_1 \vee \neg b_2 \vee \dots \vee \neg b_n \quad \text{(négative)}$$

Intuitivement, elles représentent des règles « si ... alors ... »
et permettent de déduire de nouveaux faits à partir de faits existants



Le problème de la satisfaisabilité d'un ensemble
de clauses de Horn est dans la classe P

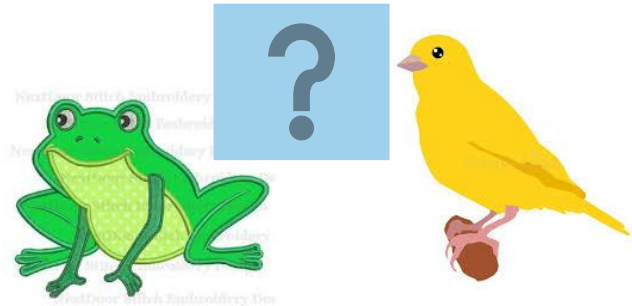
- Étant donnée une base de connaissances qui organise les connaissances et l'expertise liées au domaine par des règles et des faits, il y a trois possibilités pour le moteur d'inférence :
 - Chaînage avant (c-a-d guidé par les données – essentiellement *modus ponens*)
 - Chaînage arrière (c-a-d guidé par les hypothèses)
 - Mixte (c-a-d, combinaison de chaînage avant et arrière)
- La plupart des systèmes experts supposent que le fonctionnement du moteur d'inférences est monotone
- Quelques systèmes permettent de raisonner sous incertitude
 - Probabiliste (MYCIN, Bayes, Demster-Shafer)
 - Possibiliste (logique floue)
 - Non-monotone (systèmes de maintien de la cohérence)

Chaînage avant

- Méthode de déduction qui applique des règles en partant des prémisses pour en déduire de nouvelles conclusions.
- Ces conclusions enrichissent la mémoire de travail et peuvent devenir les prémisses d'autres règles.
- Le chaînage avant est utilisé dans un système expert à base de règles
- Il est complet (toutes les conclusions qui suivent logiquement des prémisses sont générées)

Exemple

- 1) Si X coasse et mange des mouches, alors X est une grenouille.
 - 2) Si X piaule et chante, alors X est un canari.
 - 3) Si X est une grenouille, alors X est vert.
 - 4) Si X est un canari, alors X est jaune.
 - 5) Fritz coasse et mange des mouches
- Quelle est la couleur de Fritz ?

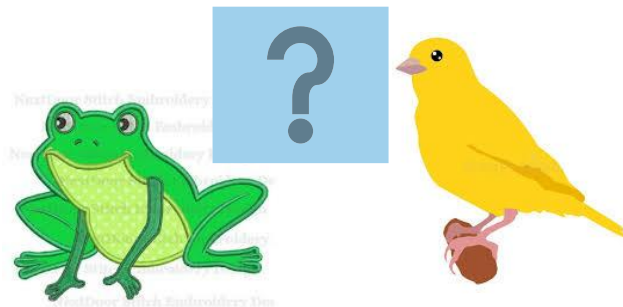


Chaînage Arrière

- Dans le chaînage arrière, on part des conclusions et on cherche de voir si leurs prémisses sont satisfaites
- Typiquement, on fait une recherche en profondeur d'abord
- Plus rapide que le chaînage avant
- Le chaînage arrière est mis en œuvre dans la programmation logique

Exemple

- 1) Si X coasse et mange des mouches, alors X est une grenouille.
 - 2) Si X piaule et chante, alors X est un canari.
 - 3) Si X est une grenouille, alors **X est vert.**
 - 4) Si X est un canari, alors **X est jaune.**
 - 5) Fritz coasse et mange des mouches
- Quelle est la couleur de Fritz ?



- Une forme de programmation qui définit les applications à l'aide
 - d'un ensemble de faits élémentaires les concernant
 - de règles leur associant des conséquences plus ou moins directes
- Ces faits et ces règles sont exploités par un moteur d'inférence, en réaction à une question ou requête
- Cette approche se révèle beaucoup plus souple que la définition d'une succession d'instructions que l'ordinateur exécuterait
- La programmation logique est considérée comme une **programmation déclarative** plutôt qu'impérative, car elle s'attache davantage au quoi qu'au comment, le moteur assumant une large part des enchaînements
- Elle est particulièrement adaptée aux besoins de l'intelligence artificielle

Prolog

- Langage de programmation logique
- Inventé par Alain Colmerauer (U Marseille) en 1971
- Interprète Prolog d'Édimbourg (1977)
- Prolog II, III, et IV (programmation par contraintes)
- Basé sur les clauses de Horn, l'unification, la résolution et le chaînage avant

Syntaxe du Prolog (1)

- Un programme Prolog se présente comme une suite de règles ou clauses de la forme
 $P \text{ :- } Q_1, Q_2, \dots, Q_n$
- qu'on interprète comme :
 P est vrai si Q_1, Q_2, \dots, Q_n sont vrais.
- Une règle de la forme
 P.
- est appelée un fait car il n'y a aucune condition pour sa véracité.
- P est appelé la tête et Q_1, Q_2, \dots, Q_n le corps de la règle.
- Chacun de ces éléments est appelé un littéral, composé d'un symbole de prédicat et d'arguments entre parenthèses séparés par des virgules

Syntaxe du Prolog (2)

- Les variables sont notées par des identifiants débutant par une majuscule ou un souligné :
`X, Couleur, _animal`
- Les constantes objet sont notées soit par des nombres ou par des identifiants débutant par une minuscule :
`jaune, vert, mouches, fritz, 2, 3.14`
- Les termes composés sont notés par un foncteur et des arguments qui sont eux-mêmes des termes :
`parent(X), parent(parent(fritz)), factoriel(3), max(0, X).`
- Les prédicats sont notés (comme les constantes et les foncteurs), par des identifiants débutant par une minuscule
`mange(X, Y), coasse(fritz)`

Exemple

grenouille(X) :- coasse(X), mange(X, mouches).

canari(X) :- piaule(X), chante(X).

couleur(X, vert) :- grenouille(X).

couleur(X, jaune) :- canari(X).

coasse(fritz).

mange(fritz, mouches).

?- couleur(fritz, Couleur)

Couleur = vert