Systèmes à Base de Connaissances

Damien.Olivier@univ-lehavre.fr

Université du Havre

2024

Systèmes à Base de Connaissances

Damien.Olivier@univ-lehavre.fr

Université du Havre

2024

Systèmes à Bases de Connaissances

Objectif de la présentation

- But ;

Modes de représentation de connaissances

Systèmes à Bases de Connaissances

Objectif de la présentation

- But ;
- Structure ;

Modes de représentation de connaissances

Systèmes à Bases de Connaissances

Objectif de la présentation

- But ;
- Structure ;
- Principes fondamentaux ;

Modes de représentation de connaissances

Systèmes à Bases de Connaissances

Objectif de la présentation

- But ;
- Structure ;
- Principes fondamentaux ;

Modes de représentation de connaissances

- Logique ;

Systèmes à Bases de Connaissances

Objectif de la présentation

- But ;
- Structure ;
- Principes fondamentaux ;

Modes de représentation de connaissances

- Logique ;
- Règles de production ;

Systèmes à Bases de Connaissances

Objectif de la présentation

- But ;
- Structure ;
- Principes fondamentaux ;

Modes de représentation de connaissances

- Logique ;
- Règles de production ;
- Réseaux sémantiques ;

Systèmes à Bases de Connaissances

Objectif de la présentation

- But ;
- Structure ;
- Principes fondamentaux ;

Modes de représentation de connaissances

- Logique ;
- Règles de production ;
- Réseaux sémantiques ;
- Objets.

Séparation des connaissances des traitements

On distingue :

- Les connaissances exprimées de façon déclarative ;

Séparation des connaissances des traitements

On distingue :

- Les connaissances exprimées de façon déclarative ;
- Des mécanismes d'exploitation de ces connaissances qui vont permettent d'inférer de nouvelles connaissances.

Séparation des connaissances des traitements

On distingue :

- Les connaissances exprimées de façon déclarative ;
- Des mécanismes d'exploitation de ces connaissances qui vont permettent d'inférer de nouvelles connaissances.

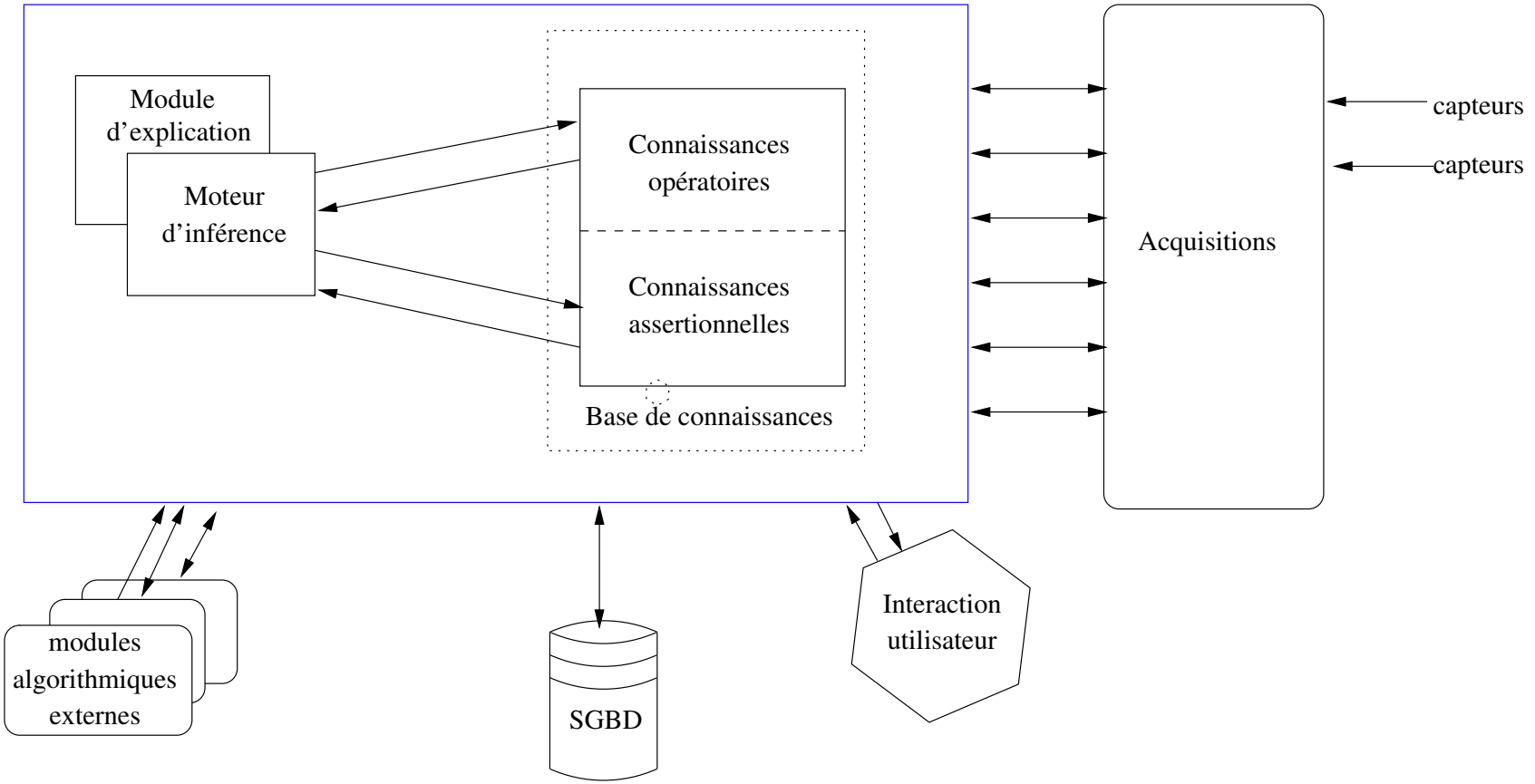
Opposition

Un langage de programmation n'est pas un système à base de connaissances. Les programmes mélangent données (variables) et mécanismes (instructions) chargés de les exploiter.

Caractéristiques

- Énoncé des connaissances indépendant de leurs modes d'exploitation ;
- Facilité de compréhension et de modification des connaissances ;
- Contrôle des mécanismes ;
- Génération d'explications ;
- Possibilité d'utiliser des connaissances :
  - incrémentales ;
  - évolutives ;
  - non consensuelles ;
  - multiples ;
  - contradictoires.

Architecture



Objectifs

- Modélisation des connaissances ;
- Modélisation du raisonnement ;
- Génération d'explications
- Acquisition et apprentissage
- Interfaces homme / machine
- Interaction avec les bases de données

Procédural vs. Déclaratif

- Opposition procédural / déclaratif ;

Procédural vs. Déclaratif

- Opposition procédural / déclaratif ;
- Savoir comment / savoir quoi.

Procédural vs. Déclaratif

- Opposition procédural / déclaratif ;
- Savoir comment / savoir quoi.

Prenons un cercle

Procédural vs. Déclaratif

- Opposition procédural / déclaratif ;
- Savoir comment / savoir quoi.

Prenons un cercle

- Faire tourner un compas dont l'une des deux branches est fixe jusqu'à ce que l'autre branche soir revenue à son point de départ

Procédural vs. Déclaratif

- Opposition procédural / déclaratif ;
- Savoir comment / savoir quoi.

Prenons un cercle

- Faire tourner un compas dont l'une des deux branches est fixe jusqu'à ce que l'autre branche soir revenue à son point de départ $\Rightarrow$ Comment ;

Procédural vs. Déclaratif

- Opposition procédural / déclaratif ;
- Savoir comment / savoir quoi.

Prenons un cercle

- Faire tourner un compas dont l'une des deux branches est fixe jusqu'à ce que l'autre branche soir revenue à son point de départ $\Rightarrow$ Comment ;
- Un cercle est le lieu de tous les points équidistants d'un point donné

Procédural vs. Déclaratif

- Opposition procédural / déclaratif ;
- Savoir comment / savoir quoi.

Prenons un cercle

- Faire tourner un compas dont l'une des deux branches est fixe jusqu'à ce que l'autre branche soir revenue à son point de départ $\Rightarrow$ Comment ;
- Un cercle est le lieu de tous les points équidistants d'un point donné $\Rightarrow$ Quoi ;

Procédural vs. Déclaratif

- Opposition procédural / déclaratif ;
- Savoir comment / savoir quoi.

Prenons un cercle

- Faire tourner un compas dont l'une des deux branches est fixe jusqu'à ce que l'autre branche soir revenue à son point de départ $\Rightarrow$ Comment ;
- Un cercle est le lieu de tous les points équidistants d'un point donné $\Rightarrow$ Quoi ;

Idée

Distinguer l'expression de ces deux savoirs et les combiner.

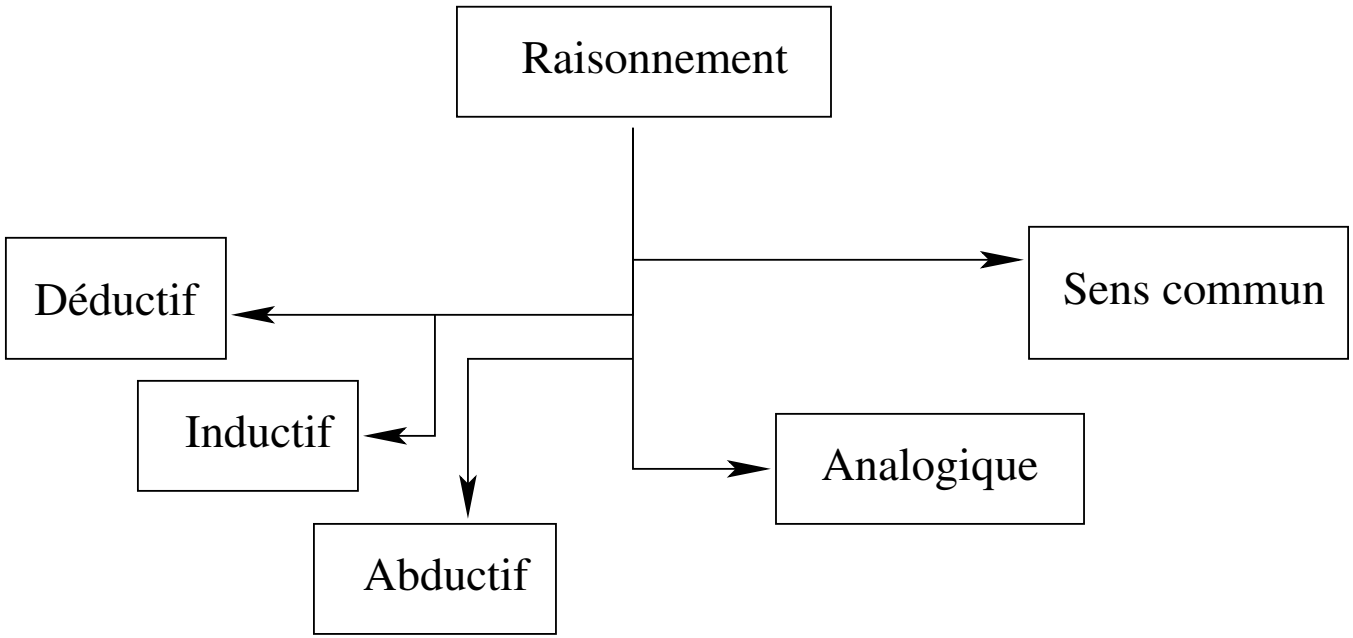
Connaissances déclaratives

- L'énoncé des connaissances ne préjuge pas de leur utilisation ultérieure ;
- Les connaissances sont faciles à lire et à modifier ;
- Formalisme proche de la représentation de l'utilisateur.

Raisonnement

- C'est le processus de faire coopérer connaissances, faits, et stratégies de résolution de problèmes, dans le but d'atteindre des conclusions.
- Comprendre comment un expert humain raisonne lors de la résolution d'un problème.

Types de raisonnement



Types de raisonnement

- Raisonnement par déduction
  A est vrai
  $A \Rightarrow B$ est vrai
  On en déduit B est vrai

Types de raisonnement

- Raisonnement par déduction
  A est vrai
  $A \Rightarrow B$ est vrai
  On en déduit B est vrai

- Raisonnement par abduction
  C'est une inférence plausible
  B est vrai
  $A \Rightarrow B$ est vrai
  On abduit que A est vrai

Types de raisonnement

- Raisonnement par déduction
  A est vrai
  $A \Rightarrow B$ est vrai
  On en déduit B est vrai
- Raisonnement par induction
  Un ensemble d'objets $\{a, b, c, d, \ldots\}$
  Une propriété P vraie pour les objets $a, b, c, \ldots$ de l'ensemble
  On induit que P est vraie pour tout x de l'ensemble
- Raisonnement par abduction
  C'est une inférence plausible
  B est vrai
  $A \Rightarrow B$ est vrai
  On abduit que A est vrai

Types de raisonnement

- Raisonnement par déduction
  A est vrai
  $A \Rightarrow B$ est vrai
  On en déduit B est vrai
- Raisonnement par induction
  Un ensemble d'objets $\{a, b, c, d, \ldots\}$
  Une propriété P vraie pour les objets $a, b, c, \ldots$ de l'ensemble
  On induit que P est vraie pour tout x de l'ensemble
- Raisonnement par abduction
  C'est une inférence plausible
  B est vrai
  $A \Rightarrow B$ est vrai
  On abduit que A est vrai
- Raisonnement par analogie
  Faire une analogie entre 2 situations, rechercher les similarités et différences, etc.
  Exploite par ex. la notion de frame(cf. suite) pour raisonner.

Types de raisonnement

- Raisonnement par déduction
  A est vrai
  $A \Rightarrow B$ est vrai
  On en déduit B est vrai
- Raisonnement par induction
  Un ensemble d'objets $\{a, b, c, d, \ldots\}$
  Une propriété P vraie pour les objets $a, b, c, \ldots$ de l'ensemble
  On induit que P est vraie pour tout x de l'ensemble
- Raisonnement par abduction
  C'est une inférence plausible
  B est vrai
  $A \Rightarrow B$ est vrai
  On abduit que A est vrai
- Raisonnement par analogie
  Faire une analogie entre 2 situations, rechercher les similarités et différences, etc. Exploite par ex. la notion de frame(cf. suite) pour raisonner.
- Sens commun
  S'appuie sur l'expérience de l'expert, sur la notion de «bon» jugement, plus que sur la logique. Notion d'heuristique.

Mode de représentation des connaissances

Au niveau des systèmes à base de connaissances, on trouve :

- La logique ;

Mode de représentation des connaissances

Au niveau des systèmes à base de connaissances, on trouve :

- La logique ;
- Les règles de production ;

Mode de représentation des connaissances

Au niveau des systèmes à base de connaissances, on trouve :

- La logique ;
- Les règles de production ;
- Les réseaux sémantiques ;

Mode de représentation des connaissances

Au niveau des systèmes à base de connaissances, on trouve :

- La logique ;
- Les règles de production ;
- Les réseaux sémantiques ;
- Les objets.

La logique

Langage formel qui permet d'exprimer des connaissances avec rigueur, ainsi que des méthodes de combinaison de ces connaissances qui en engrendrent de nouvelles.

La logique

Langage formel qui permet d'exprimer des connaissances avec rigueur, ainsi que des méthodes de combinaison de ces connaissances qui en engrendrent de nouvelles.

- Logique des propositions (d'ordre 0 - sans variable) ;

La logique

Langage formel qui permet d'exprimer des connaissances avec rigueur, ainsi que des méthodes de combinaison de ces connaissances qui en engrendrent de nouvelles.

- Logique des propositions (d'ordre 0 - sans variable) ;
- Logique des prédicats (d'ordre 1 - utilisation de variables) ;

La logique

Langage formel qui permet d'exprimer des connaissances avec rigueur, ainsi que des méthodes de combinaison de ces connaissances qui en engrendrent de nouvelles.

- Logique des propositions (d'ordre 0 - sans variable) ;
- Logique des prédicats (d'ordre 1 - utilisation de variables) ;
- Mais aussi logiques multivaluées, floues, modales, de défauts, temporelles ...

Syntaxe de la logique des prédicats

- Les termes
  - les objets du domaine : titi, température, ...
  - les variables : x, y, z, ...
  - les fonctions : père, fils ...
- Les formules atomiques
  - prédicats dont les arguments sont des termes : (ex : $estPère(titi, rominet), vole(titi)$)
- Les formules bien formées
  - Les formules atomiques (par définition) ;
  - Des formules bien formées liées par des *connecteurs* $\neg \wedge \vee \Rightarrow \Leftrightarrow$: (ex : $G \vee H$) ;
  - Des formules bien formées utilisant des variables quantifiées $\forall \exists$: (ex : $\forall y \; \exists x \; P(x, y) \vee Q(x, y) \Rightarrow R(x)$) ou encore ($\forall x \; Plume(x) \Rightarrow Oiseau(x)$).

Résolution

- *Syllogisme* : Socrate est un homme, tout homme est mortel donc Socrate est mortel ;

Résolution

- *Syllogisme* : Socrate est un homme, tout homme est mortel donc Socrate est mortel ;
- Règles de dérivation : $E1 \Rightarrow E2$ est équivalent à $\neg E1 \vee E2$

Résolution

- *Syllogisme* : Socrate est un homme, tout homme est mortel donc Socrate est mortel ;
- Règles de dérivation : $E1 \Rightarrow E2$ est équivalent à $\neg E1 \vee E2$
  - modus ponens
    Si $E1 \Rightarrow E2$ et $E1$ alors on peut déduire $E2$

Résolution

- *Syllogisme* : Socrate est un homme, tout homme est mortel donc Socrate est mortel ;
- Règles de dérivation : $E1 \Rightarrow E2$ est équivalent à $\neg E1 \vee E2$
  - modus ponens
    Si $E1 \Rightarrow E2$ et $E1$ alors on peut déduire $E2$
  - modus tollens
    Si $E1 \Rightarrow E2$ et $\neg E2$ alors on peut déduire $\neg E1$

Résolution

- *Syllogisme* : Socrate est un homme, tout homme est mortel donc Socrate est mortel ;
- Règles de dérivation : $E1 \Rightarrow E2$ est équivalent à $\neg E1 \vee E2$
  - modus ponens
    Si $E1 \Rightarrow E2$ et $E1$ alors on peut déduire $E2$
  - modus tollens
    Si $E1 \Rightarrow E2$ et $\neg E2$ alors on peut déduire $\neg E1$
- Principe de résolution :

Résolution

- *Syllogisme* : Socrate est un homme, tout homme est mortel donc Socrate est mortel ;
- Règles de dérivation : $E1 \Rightarrow E2$ est équivalent à $\neg E1 \vee E2$
  - modus ponens
    Si $E1 \Rightarrow E2$ et $E1$ alors on peut déduire $E2$
  - modus tollens
    Si $E1 \Rightarrow E2$ et $\neg E2$ alors on peut déduire $\neg E1$
- Principe de résolution :
  - Supposer que la négation du théorème est vraie ;

Résolution

- *Syllogisme* : Socrate est un homme, tout homme est mortel donc Socrate est mortel ;
- Règles de dérivation : $E1 \Rightarrow E2$ est équivalent à $\neg E1 \vee E2$
  - modus ponens
    Si $E1 \Rightarrow E2$ et $E1$ alors on peut déduire $E2$
  - modus tollens
    Si $E1 \Rightarrow E2$ et $\neg E2$ alors on peut déduire $\neg E1$
- Principe de résolution :
  - Supposer que la négation du théorème est vraie ;
  - Démontrer que les axiomes et la négation du théorème déterminent quelque chose de vrai et qui ne peut l'être ;

Résolution

- *Syllogisme* : Socrate est un homme, tout homme est mortel donc Socrate est mortel ;
- Règles de dérivation : $E1 \Rightarrow E2$ est équivalent à $\neg E1 \vee E2$
  - modus ponens
    Si $E1 \Rightarrow E2$ et $E1$ alors on peut déduire $E2$
  - modus tollens
    Si $E1 \Rightarrow E2$ et $\neg E2$ alors on peut déduire $\neg E1$
- Principe de résolution :
  - Supposer que la négation du théorème est vraie ;
  - Démontrer que les axiomes et la négation du théorème déterminent quelque chose de vrai et qui ne peut l'être ;
  - Conclure que la négation ne peut être vraie car elle conduit à une contradiction ;

Résolution

- *Syllogisme* : Socrate est un homme, tout homme est mortel donc Socrate est mortel ;
- Règles de dérivation : $E1 \Rightarrow E2$ est équivalent à $\neg E1 \vee E2$
  - modus ponens
    Si $E1 \Rightarrow E2$ et $E1$ alors on peut déduire $E2$
  - modus tollens
    Si $E1 \Rightarrow E2$ et $\neg E2$ alors on peut déduire $\neg E1$
- Principe de résolution :
  - Supposer que la négation du théorème est vraie ;
  - Démontrer que les axiomes et la négation du théorème déterminent quelque chose de vrai et qui ne peut l'être ;
  - Conclure que la négation ne peut être vraie car elle conduit à une contradiction ;
  - Conclure que le théorème est vrai.

Exemple

Montrer que le  est un oiseau. Sachant que :

- $\forall x \neg(Plumes(x) \Rightarrow Oiseau(x)), Plumes(BipBip)$

Avantages

- Syntaxe précise, formalisée, répondant à une terminologie standard ;
- Puissance d'expression notamment grâce aux formules quantifiées, à la disjonction et à la négation ;
- Cohérence des structures symboliques déduites ;
- Méthodes de résolution aux propriétés et limites rigoureusement évaluées, aux solutions justifiées ;

Utilisation principale

Beaucoup utilisée pour la démonstration
Un langage de programmation support : Prolog

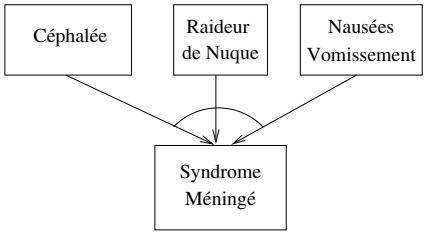
Inconvénients

- Manque de structuration de la connaissance ;
- Difficultés d'expression des connaissances de contrôle ;
- Difficultés d'expression de l'incertitude, du contexte d'interprétation, de la révision d'un énoncé ou d'une déduction.

Les règles

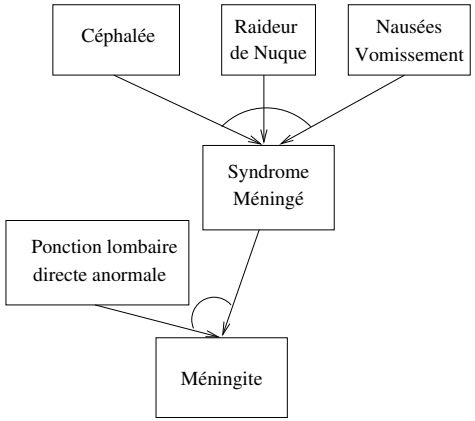
- C'est une quantité de connaissances, déclarative et autonome, de la forme :
  SI condition ALORS conclusion (a)
  - Condition et conclusion sont des expressions logiques ;
  - Condition doit être vérifiée pour que la règle s'applique ;
  - Conclusion peut correspondre à l'ajout ou au retrait d'un fait ou d'une hypothèse, au déclenchement d'une action, etc.
  - Dans certains systèmes à règles de production, a est un coefficient numérique qui traduit que la connaissance exprimée par la règle est entachée d'incertitude (coefficient de vraisemblance, facteur de certitude,...)
    $\Rightarrow$ raisonnement approximatif (logiques multivaluées, floues, ...).

Exemple



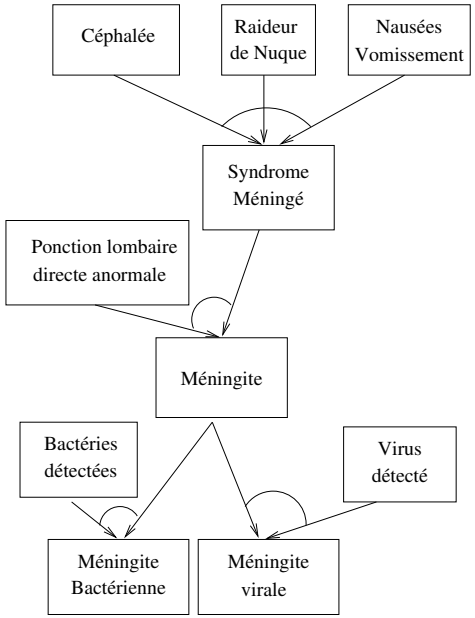
- *SI* Céphalée *et* RaideurdeNuque *et* Nausées *ALORS* SyndromeMéningé

Exemple



- *SI* Céphalée *et* RaideurdeNuque *et* Nausées *ALORS* SyndromeMéningé
- *SI* PonctionAnormale *et* SyndromeMéningé *ALORS* Méningite

Exemple



- *SI* Céphalée *et* RaideurdeNuque *et* Nausées *ALORS* SyndromeMéningé
- *SI* PonctionAnormale *et* SyndromeMéningé *ALORS* Méningite
- *SI* Méningite *et* BactériesDansPonction *ALORS* MéningiteBactérienne
  SI Méningite *et* VirusDansPonction *ALORS* MéningiteVirale

Systèmes de production

Les éléments d'un système de production sont :

- Une base de connaissances composée de :

Systèmes de production

Les éléments d'un système de production sont :

- Une base de connaissances composée de :
  - Une base de données globale ou base de faits ;

Systèmes de production

Les éléments d'un système de production sont :

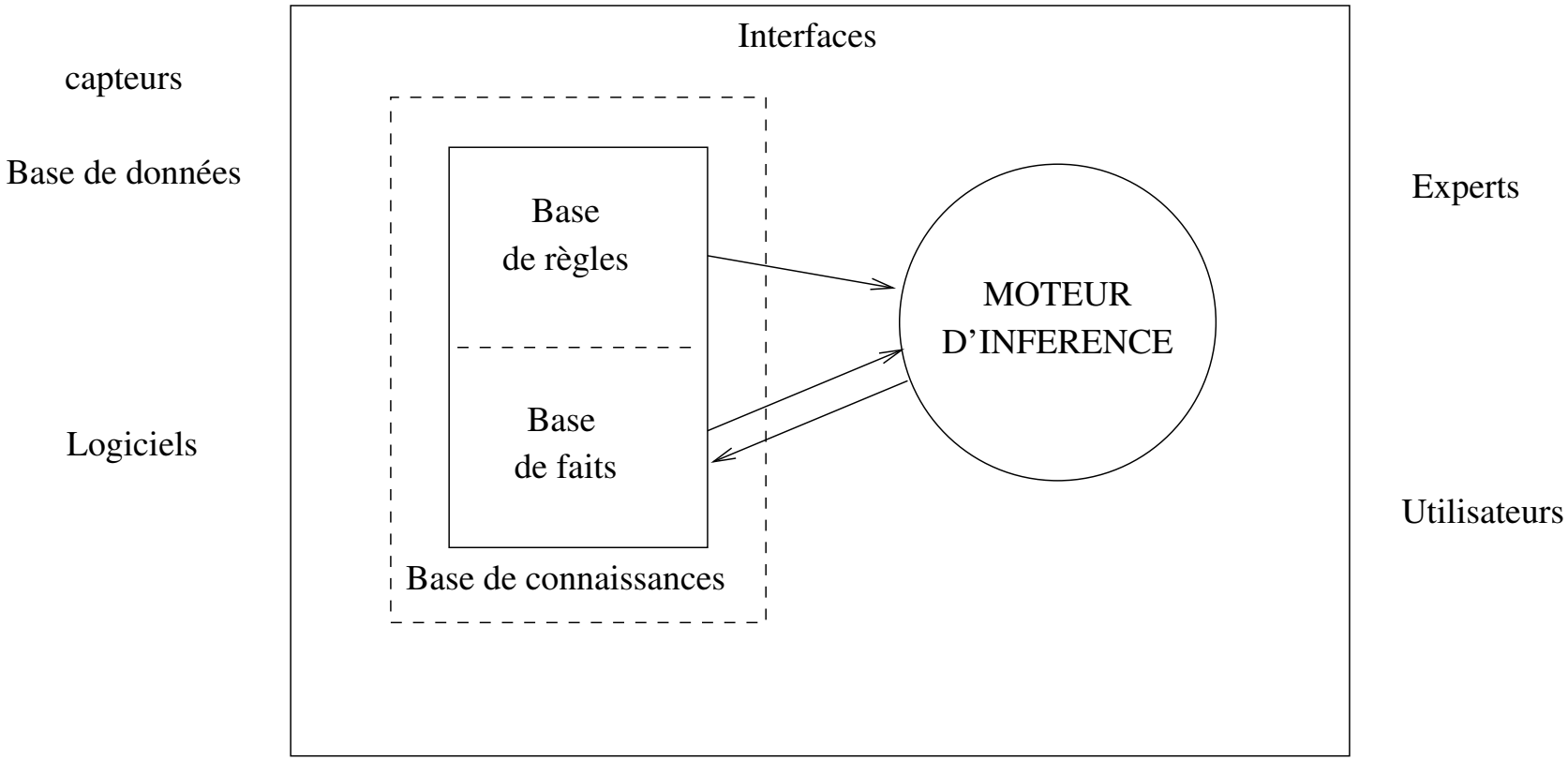
- Une base de connaissances composée de :
  - Une base de données globale ou base de faits ;
  - Un ensemble de règles de production ;

Systèmes de production

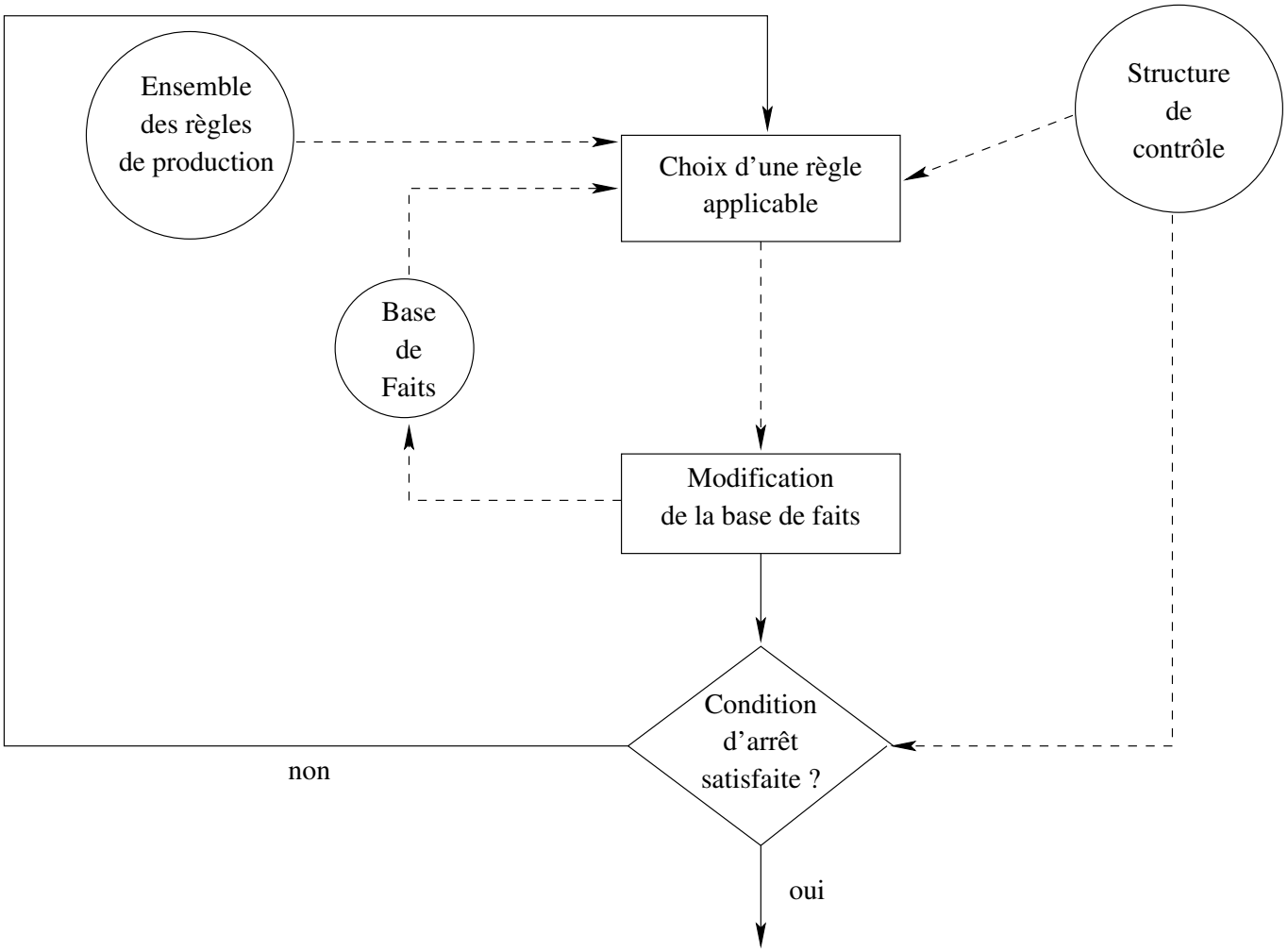
Les éléments d'un système de production sont :

- Une base de connaissances composée de :
  - Une base de données globale ou base de faits ;
  - Un ensemble de règles de production ;
- Une stratégie de contrôle et une condition d'arrêt utilisées par le moteur d'inférence.

Architecture d'un système de production



Fonctionnement d'un système de production



Systèmes de production commutatifs

Un système de production commutatif respecte les propriétés suivantes :

- Soit B la base de données globale ;

Systèmes de production commutatifs

Un système de production commutatif respecte les propriétés suivantes :

- Soit B la base de données globale ;
  - Chaque règle de l'ensemble des règles applicables à B est également applicable à toute BDG résultant de l'application d'une règle à B.

Systèmes de production commutatifs

Un système de production commutatif respecte les propriétés suivantes :

- Soit *B* la base de données globale ;
  - Chaque règle de l'ensemble des règles applicables à *B* est également applicable à toute BDG résultant de l'application d'une règle à *B*.
  - Si la condition d'arrêt est satisfaite par *B* elle le reste par toutes les BDG produites par l'application de toute règle applicable à *B*.

Systèmes de production commutatifs

Un système de production commutatif respecte les propriétés suivantes :

- Soit B la base de données globale ;
  - Chaque règle de l'ensemble des règles applicables à B est également applicable à toute BDG résultant de l'application d'une règle à B.
  - Si la condition d'arrêt est satisfaite par B elle le reste par toutes les BDG produites par l'application de toute règle applicable à B.
  - La base de données obtenue en appliquant à B une séquence de règles applicables ne varie pas $\forall$ les permutations de la séquence.

Systèmes de production commutatifs

Un système de production commutatif respecte les propriétés suivantes :

- Soit B la base de données globale ;
  - Chaque règle de l'ensemble des règles applicables à B est également applicable à toute BDG résultant de l'application d'une règle à B.
  - Si la condition d'arrêt est satisfaite par B elle le reste par toutes les BDG produites par l'application de toute règle applicable à B.
  - La base de données obtenue en appliquant à B une séquence de règles applicables ne varie pas $\forall$ les permutations de la séquence.

Une stratégie de contrôle irrévocable peut être utilisée.

Systèmes de production décomposables

Un système de production est décomposable, si :

- La BDG est décomposable en composantes sur lesquel les régles peuvent être appliquées séparement ;

Systèmes de production décomposables

Un système de production est décomposable, si :

- La BDG est décomposable en composantes sur lesquel les régles peuvent être appliquées séparement ;
- L'application d'une règle applicable quelconque sur une composante produit une BD décomposable ;

Systèmes de production décomposables

Un système de production est décomposable, si :

- La BDG est décomposable en composantes sur lesquel les régles peuvent être appliquées séparement ;
- L'application d'une règle applicable quelconque sur une composante produit une BD décomposable ;
- La condition d'arrêt peut être décomposée pour chaque composante.

Systèmes de production décomposables

Un système de production est décomposable, si :

- La BDG est décomposable en composantes sur lesquel les régles peuvent être appliquées séparement ;
- L'application d'une règle applicable quelconque sur une composante produit une BD décomposable ;
- La condition d'arrêt peut être décomposée pour chaque composante.

Le graphe d'exploration produit est un graphe ET/OU.

Graphes ET/OU

Les graphes ET/OU sont des hypergraphes.

Soit $S = s_1, s_2, \ldots, s_n$ et une *famille* $\mathcal{E} = E_1, E_2, \ldots, E_m$ de partie de S. $\mathcal{E}$ constitue un **hypergraphe** sur S si :

$$\begin{aligned} E_j &\neq \varnothing \text{ pour } j \text{ de 1 à } m \\ \bigcup_j E_j &= S \end{aligned}$$

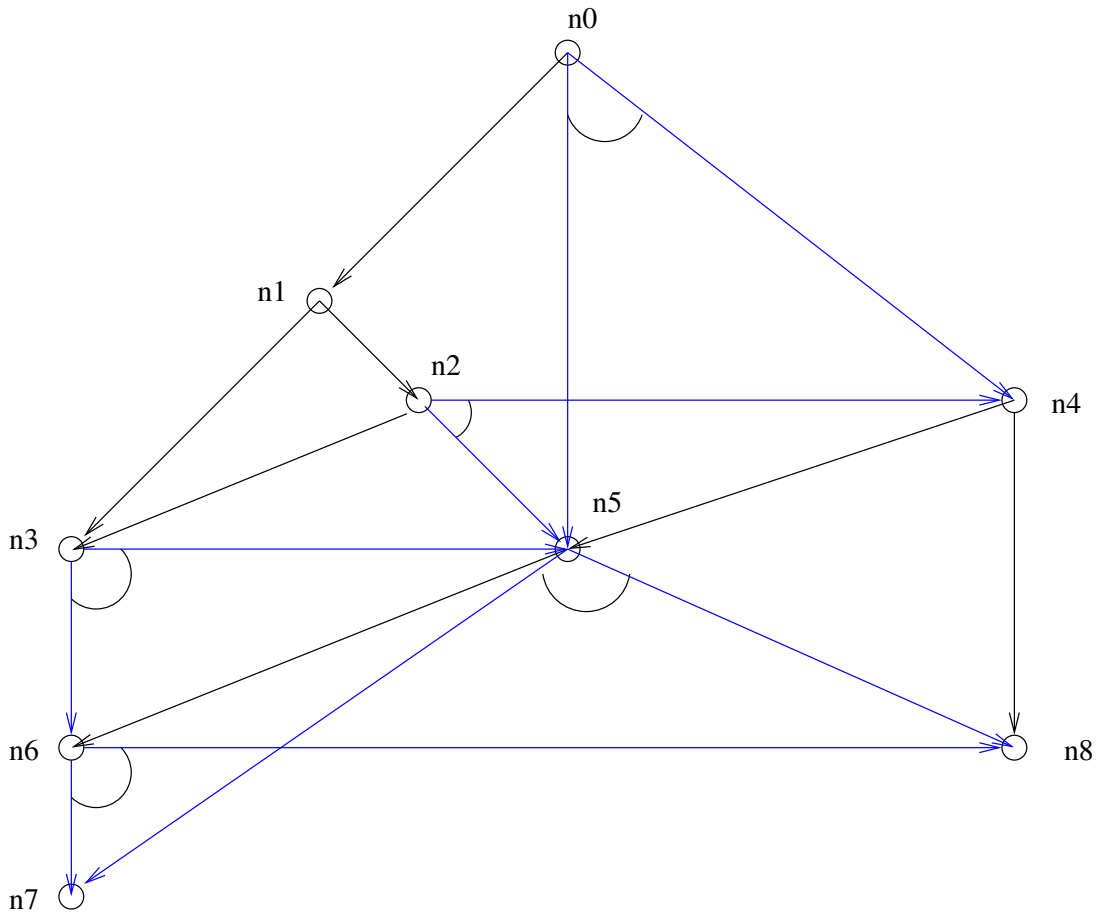
- $H = (S, \mathcal{E})$ est un hypergraphe.
- Les éléments $s_1, s_2, \ldots, s_n$ sont les sommets.
- Les éléments $E_1, \ldots, E_m$ de E sont les arêtes (hyperarcs).

Graphes ET/OU

- Les graphes ET/OU sont des hypergraphes dans lesquels les hyperarcs relient un parent à un ensemble de successeurs.
- Ces hyperarcs sont appelés *k-connecteurs*.
- Chaque k-connecteur est dirigé d'un sommet parent vers un ensemble de *k-successeurs*.

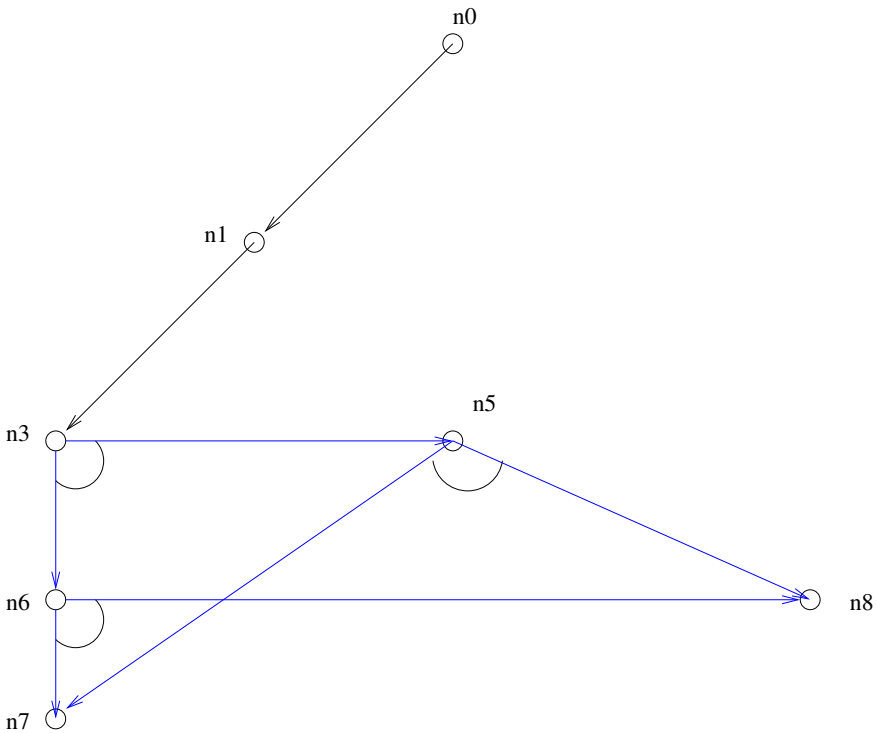
Graphes ET/OU

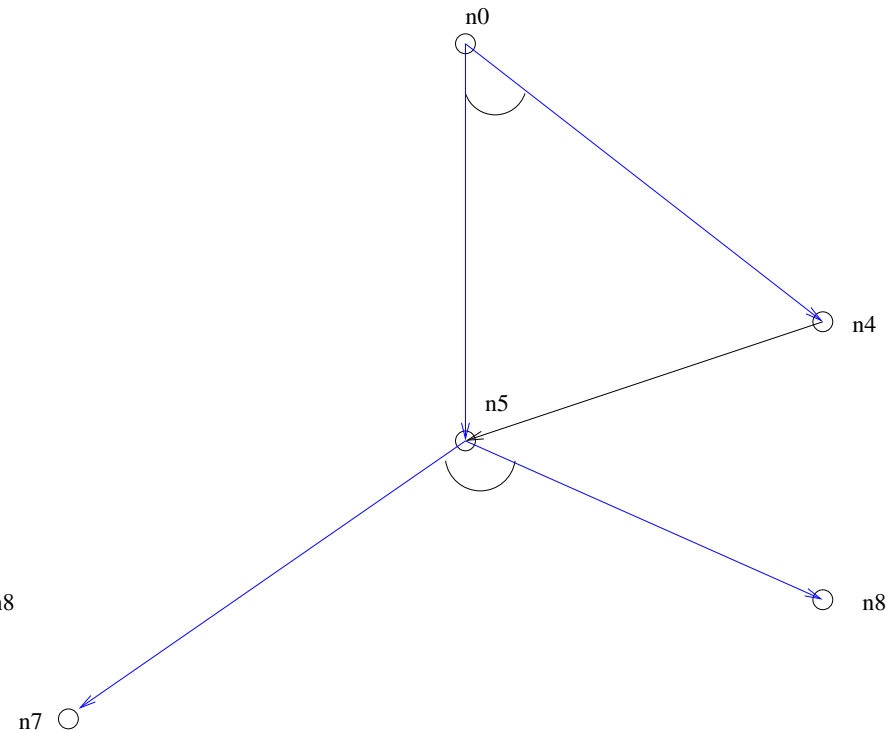
Exemple d'un graphe ET/OU



Graphes ET/OU

Graphes solutions de n_0 à $\{n_7, n_8\}$





Recherche d'un graphe solution dans un graphe ET/OU

- Nœuds OU associés au problème, nœuds ET à la décomposition ;
- Recherche d'une solution = sous graphe contenant :
  - Le nœud racine
  - Pour tout nœud non terminal, un seul k-connecteur et l'ensemble de k-successeurs.

Recherche d'un graphe solution dans un graphe ET/OU

Graphe solution

Soit G' un graphe solution du nœud n à l'ensemble N d'un graphe G ET/OU

1. Si $n \in N$, G' est constitué uniquement de n.

Recherche d'un graphe solution dans un graphe ET/OU

Graphe solution

Soit G' un graphe solution du nœud n à l'ensemble N d'un graphe G ET/OU

1. Si $n \in N$, G' est constitué uniquement de n.
2. Sinon, si n a un k-connecteur dirigé vers les nœuds $n_1, n_2, \ldots, n_k$ de sorte qu'il existe un graphe solution vers N à partir de chacun des n_i avec $i = 1 \ldots k$, alors G' est constitué du nœud n, du k-connecteur, des nœuds $n_1, \ldots, n_k$ et des graphes solutions vers N de chacun des nœuds $n_1, \ldots, n_k$.

Recherche d'un graphe solution dans un graphe ET/OU

Graphe solution

Soit G' un graphe solution du nœud n à l'ensemble N d'un graphe G ET/OU

1. Si $n \in N$, G' est constitué uniquement de n.
2. Sinon, si n a un k-connecteur dirigé vers les nœuds $n_1, n_2, \ldots, n_k$ de sorte qu'il existe un graphe solution vers N à partir de chacun des n_i avec $i = 1 \ldots k$, alors G' est constitué du nœud n, du k-connecteur, des nœuds $n_1, \ldots, n_k$ et des graphes solutions vers N de chacun des nœuds $n_1, \ldots, n_k$.
3. Sinon, il n'y a pas de solution.

Recherche d'un graphe solution dans un graphe ET/OU

Coût d'un graphe solution

$k(n, N)$ coût d'un graphe solution de n à N :

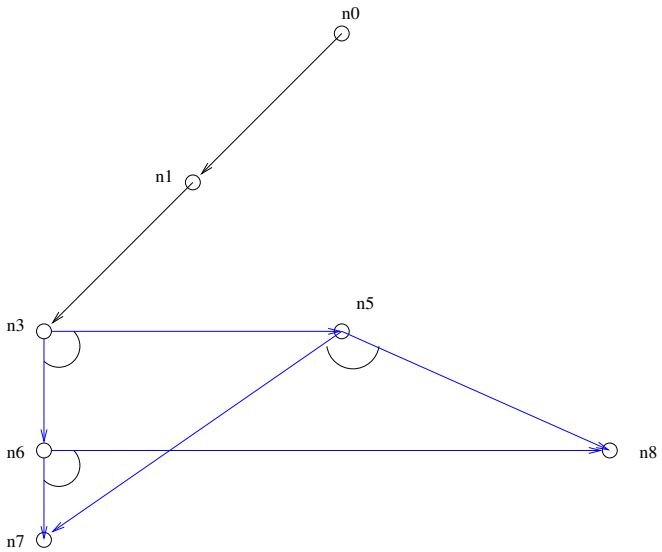
- Si $n \in N$ alors $k(n, N) = 0$
- Sinon n a un k-connecteur vers un ensemble de successeurs $n_1, n_2, \ldots, n_i$ dans le graphe solution. Soit c_n le coût de ce connecteur alors

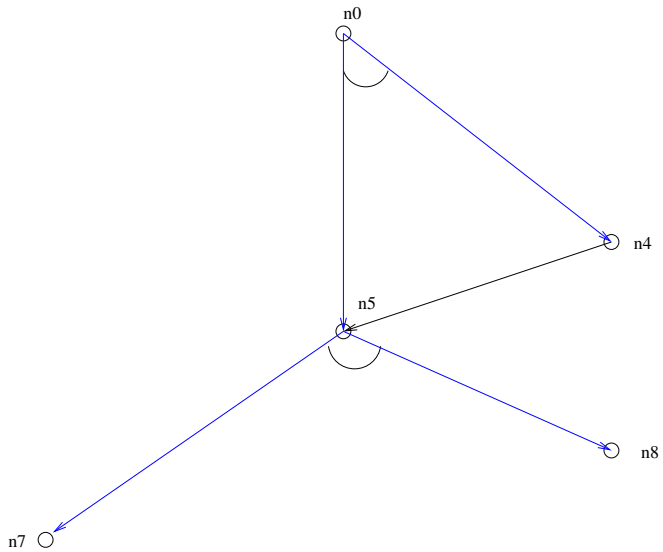
$$k(n, N) = c_n + k(n_1, N) + \ldots + k(n_i, N)$$

Recherche d'un graphe solution dans un graphe ET/OU

Coût d'un graphe solution

Quel est le coût des 2 graphes solutions si le coût d'un k-connecteur est k ?

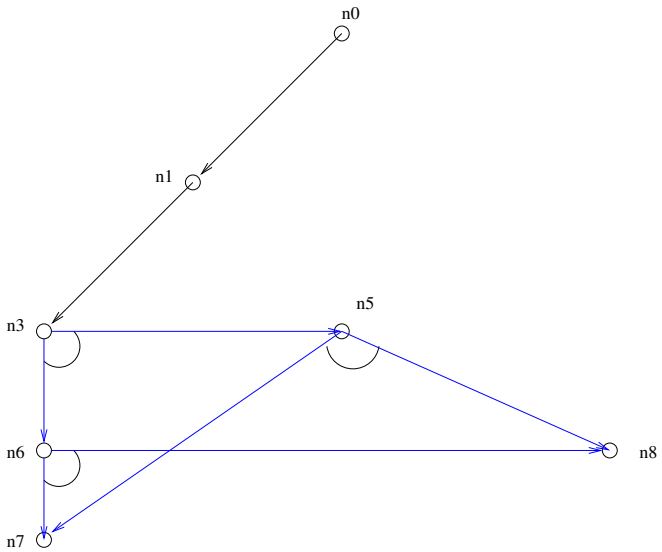




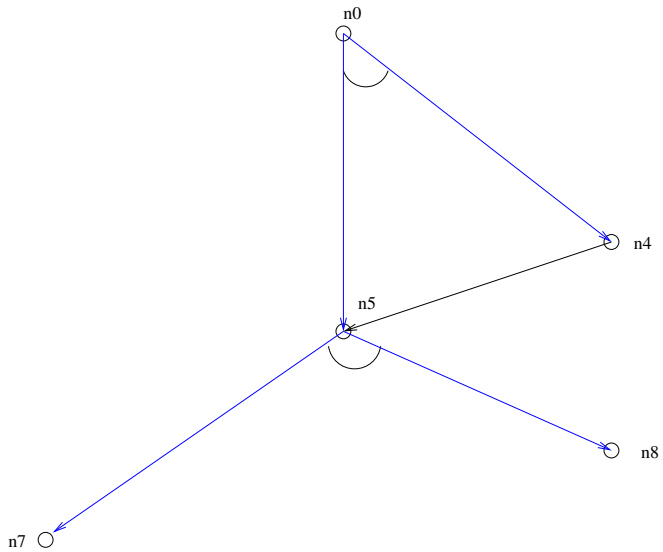
Recherche d'un graphe solution dans un graphe ET/OU

Coût d'un graphe solution

Quel est le coût des 2 graphes solutions si le coût d'un k-connecteur est k ?



coût = 8



coût = 7

Recherche d'un graphe solution dans un graphe ET/OU

- Nœuds résolus

Recherche d'un graphe solution dans un graphe ET/OU

- Nœuds résolus
- Nœuds non résolus (en échec)

Recherche d'un graphe solution dans un graphe ET/OU

- Nœuds résolus
  - Les nœuds terminaux sont des nœuds résolus ;
- Nœuds non résolus (en échec)

Recherche d'un graphe solution dans un graphe ET/OU

- Nœuds résolus
  - Les nœuds terminaux sont des nœuds résolus ;
  - Les nœuds dont un des k-connecteurs a tous ses fils résolus.
- Nœuds non résolus (en échec)

Recherche d'un graphe solution dans un graphe ET/OU

- Nœuds résolus
  - Les nœuds terminaux sont des nœuds résolus ;
  - Les nœuds dont un des k-connecteurs a tous ses fils résolus.
- Nœuds non résolus (en échec)
  - Les nœuds non terminaux sans descendant ;

Recherche d'un graphe solution dans un graphe ET/OU

- Nœuds résolus
  - Les nœuds terminaux sont des nœuds résolus ;
  - Les nœuds dont un des k-connecteurs a tous ses fils résolus.
- Nœuds non résolus (en échec)
  - Les nœuds non terminaux sans descendant ;
  - Les nœuds dont tous les k-connecteurs ont au moins un successeur insoluble.

Algorithmes de recherche dans un graphe ET/OU

Algorithme de recherche à retour arrière dans un graphe ET/OU

Entrée : u nœud de départ Appel : *Recherche*(u)

```
DEBUT
  SI u est terminal ALORS RETOURNER Succès FSI
  SI u n'est pas décomposable OU Evaluation(u) > MAX
    ALORS Retourner ECHEC FSI
  POUR chaque composante i de u FAIRE
    developper ← VRAI
    TQ developper et qu'il existe des successeurs de u pour la composante i FAIRE
      v ← Successeur(u) // Au sens de la composante
      SI v ∉ RESOLUS
        ALORS SI v ∈ INSOLUBLES
                ALORS developper ← FAUX
                SINON
                  Evaluation(v) ← Evaluation(u) + 1
                  SI Recherche(v) = ECHEC
                    ALORS Mettre v dans INSOLUBLES
                          developper ← FAUX
                    SINON Mettre v dans RESOLUS
                  FSI
              FSI
      FSI
    FTQ
    SI developper
      ALORS Mettre u dans RESOLUS
            Retouner SUCCES
    FSI
  FPOUR
  Mettre u dans INSOLUBLES
  Retourner ECHEC
FIN
```

Algorithmes de recherche dans un graphe ET/OU

Algorithme A0*

1 Créer un graphe de recherche G constitué uniquement du nœud de départ s. Attribuer à s un coût $q(s) = h(s)$, $n = s$. Si s est un nœud terminal, $h(s) = 0$, marquer s résolu.

2 TANTQUE s n'est pas marqué résolu FAIRE

- 2.1 Calculer un graphe solution partiel G' en suivant les connecteurs marqués dans G à partir de s
- 2.2 Sélectionner un nœud non terminal n de G'
- 2.3 Developper le nœud n en engendrant tous ses successeurs et mémoriser ceux-ci comme successeurs de n dans G. Pour chaque successeur n_j qui n'est pas dans G calculer $q(n_j) = h(n_j)$ Marquer résolu chacun des successeurs qui sont de nœuds terminaux.
- 2.4 $S \leftarrow \{n\}$
- 2.5 TANTQUE $S \neq \varnothing$ FAIRE
  - 2.5.1 Extraire de S un nœud m, tq m n'a pas de descendant dans G apparaissant dans S.
  - 2.5.2 Réviser le coût $q(m)$ de la façon suivante : Pour chaque k-connecteur i allant de m à un ensemble de nœ uds $\{n_{1i}, \ldots, n_{ki}\}$ calculer $q_i(m) = c_i + q(n_{i1}) + \ldots + q(n_{ki})$. Affecter à $q(m)$ le minimum de tous les connecteurs i en partance et marquer le connecteur pour lequel le minimum est atteint (on efface si la marque précédente est différente). Si tous les nœuds successeurs sont marqués résolus, alors marquer m résolus.
  - 2.5.3 Si m a été marqué résolu $m \neq s$ ou si le coût révisé de m est différent de son coût antérieur, alors ajouter à S tous les parents de m tels que m est l'un de leur successeur via un connecteur marqué.
- 2.6 FTQ

3 FTQ

Algorithmes de recherche dans un graphe ET/OU

Algorithme A0*

- Une première opération descendante qui agrandit le graphe 2 à 2.3. On cherche le meilleur graphe en suivant les connecteurs marqués. Une feuille non termianle est développée 2.2 et 2.3 et un coût est attribué à ses successeurs.

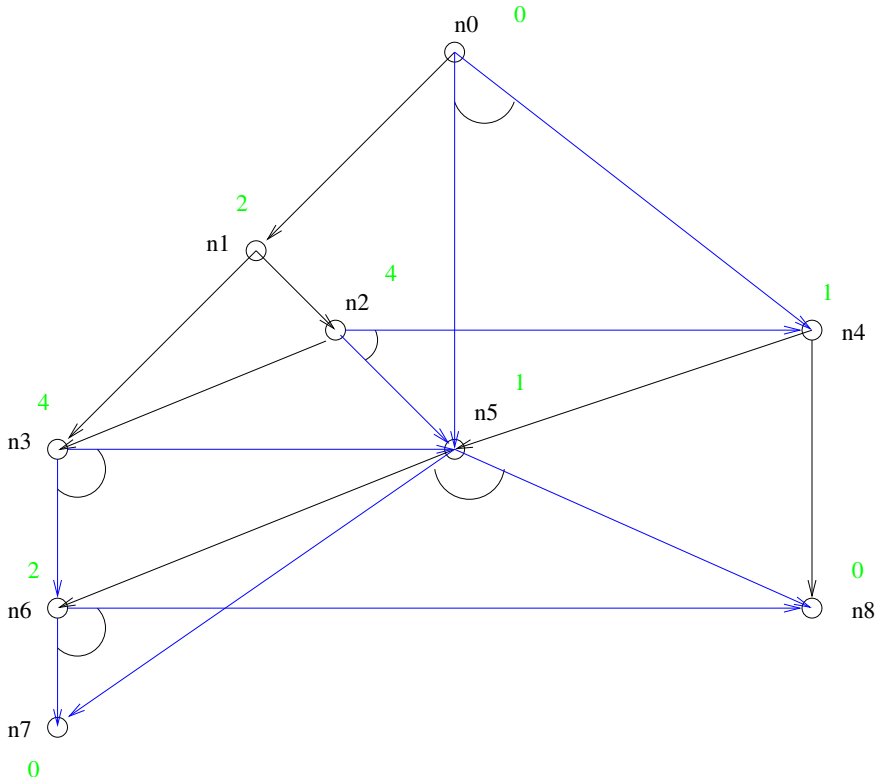
Algorithmes de recherche dans un graphe ET/OU

Algorithme A0*

- Une première opération descendante qui agrandit le graphe 2 à 2.3. On cherche le meilleur graphe en suivant les connecteurs marqués. Une feuille non termianle est développée 2.2 et 2.3 et un coût est attribué à ses successeurs.
- Un deuxième opération ascendante qui révise les coûts et marque les connecteurs résolus 2.4 à 2.5.3. En commençant par le nœud qui vient d'être développé, la procédure révise son coût en utilisant les coûts des successeurs calculés en dernier et marque les connecteurs en partance qui se trouvent sur le chemin estimé être le meilleur et qui mène aux terminaux. Cette nouvelle évaluation doit-être propagée vers le haut. Le coût révisé $q(m)$ est une évaluation mise à jour du coût d'un graphe solution optimal de n à un ensemble de nœuds terminaux.

Algorithmes de recherche dans un graphe ET/OU

Montrer le fonctionnement de l'algorithme AO* sur l'exemple suivant :



Le coût d'un k-connecteur est k et on a $h(n)$.

Systèmes de production décomposables / commutatifs

$$\begin{aligned}
\text{R1} &: T \rightarrow A, B \\
\text{R2} &: T \rightarrow B, C \\
\text{R3} &: A \rightarrow D \\
\text{R4} &: B \rightarrow E, F \\
\text{R5} &: B \rightarrow G \\
\text{R6} &: C \rightarrow G
\end{aligned}$$

Systèmes de production décomposables / commutatifs

$$
\begin{aligned}
\text{R1} &: T \rightarrow A, B \\
\text{R2} &: T \rightarrow B, C \\
\text{R3} &: A \rightarrow D \\
\text{R4} &: B \rightarrow E, F \\
\text{R5} &: B \rightarrow G \\
\text{R6} &: C \rightarrow G
\end{aligned}
$$

- Une règle est applicable quand la partie gauche est présente dans la BDG initiale. Le système de production est ici décomposable.

Systèmes de production décomposables / commutatifs

$$
\begin{aligned}
\text{R1} &: T \rightarrow A, B \\
\text{R2} &: T \rightarrow B, C \\
\text{R3} &: A \rightarrow D \\
\text{R4} &: B \rightarrow E, F \\
\text{R5} &: B \rightarrow G \\
\text{R6} &: C \rightarrow G
\end{aligned}
$$

- Une règle est applicable quand la partie gauche est présente dans la BDG initiale. Le système de production est ici décomposable.
- On peut utiliser les règles en sens inverse. Le système de production est commutatif.

Raisonnement dans les Systèmes de production

Caractérisé par :

- Le mode fonctionnement du moteur (avant, arrière, mixte) ;
- La possibilité d'introduire des variables dans les règles ;
- La monotonie ;
- L'application d'heuristique sous la forme de méta-règles ;
- ...

Chaînage avant

Chainage avant, sans but, irrévocable et monotone

- Guidé par les faits ;
- Pas de but, déclenchement des règles jusqu'à saturation ;
  - Ajout des parties conclusions des règles quand les prémisses sont vérifiées ;
  - En largeur d'abord.
- Irrévocable : le déclenchement d'une règle n'est pas remis en cause ;
- Monotone : les faits produits ne sont pas remis en cause.
- Le système de production doit être commutatif ;

Si on introduit un but, une distance par rapport à ce but est utilisée pour le choix des règles.

Chaînage arrière

Chainage arrière monotone

- Dirigé par le buts ;
- Un but est assigné au système ;
- On sélectionne alors les règles ayant ce but comme conclusion, et vérifie si les prémisses de ces règles font partie de la base des faits. Les prémisses n'appartenant pas à la base de faits deviennent à leur tour des buts à prouver de la même façon.
- Le raisonnement se fait des solutions vers les faits initiaux.
- Construction d'un graphe ET/OU.

Comparaison avant/arrière

	Chaînage avant	Chaînage arrière
Avantages	Beaucoup de fait initiaux Produit beaucoup de faits Planification, contrôle, interprétation	Prouver une hypothèse Questions ciblées Diagnostic
Inconvénients	Ne perçoit pas certaines évidences	Développement de branches improductives à l'évidence

Avantages/Inconvénients

Avantages/Inconvénients

- Avantage
  - Formalisme permettant la représentation d'un comportement humain, fort étudié et utilisé ;
  - Possibilité d'exprimer des stratégies (si... alors...) ;

Avantages/Inconvénients

- Avantage
  - Formalisme permettant la représentation d'un comportement humain, fort étudié et utilisé ;
  - Possibilité d'exprimer des stratégies (si... alors...) ;
- Inconvénients
  - Modifications difficiles ;
  - Manque de structuration de la connaissance ;
  - Difficulté de prendre en compte les exceptions ;
  - Difficulté de raisonner avec des informations incomplètes ;
  - Redondances dans les expressions des règles ;
  - Difficulté de description de systèmes complexes et structurés.

Avantages/Inconvénients

- Avantage
  - Formalisme permettant la représentation d'un comportement humain, fort étudié et utilisé ;
  - Possibilité d'exprimer des stratégies (si... alors...) ;
- Inconvénients
  - Modifications difficiles ;
  - Manque de structuration de la connaissance ;
  - Difficulté de prendre en compte les exceptions ;
  - Difficulté de raisonner avec des informations incomplètes ;
  - Redondances dans les expressions des règles ;
  - Difficulté de description de systèmes complexes et structurés.

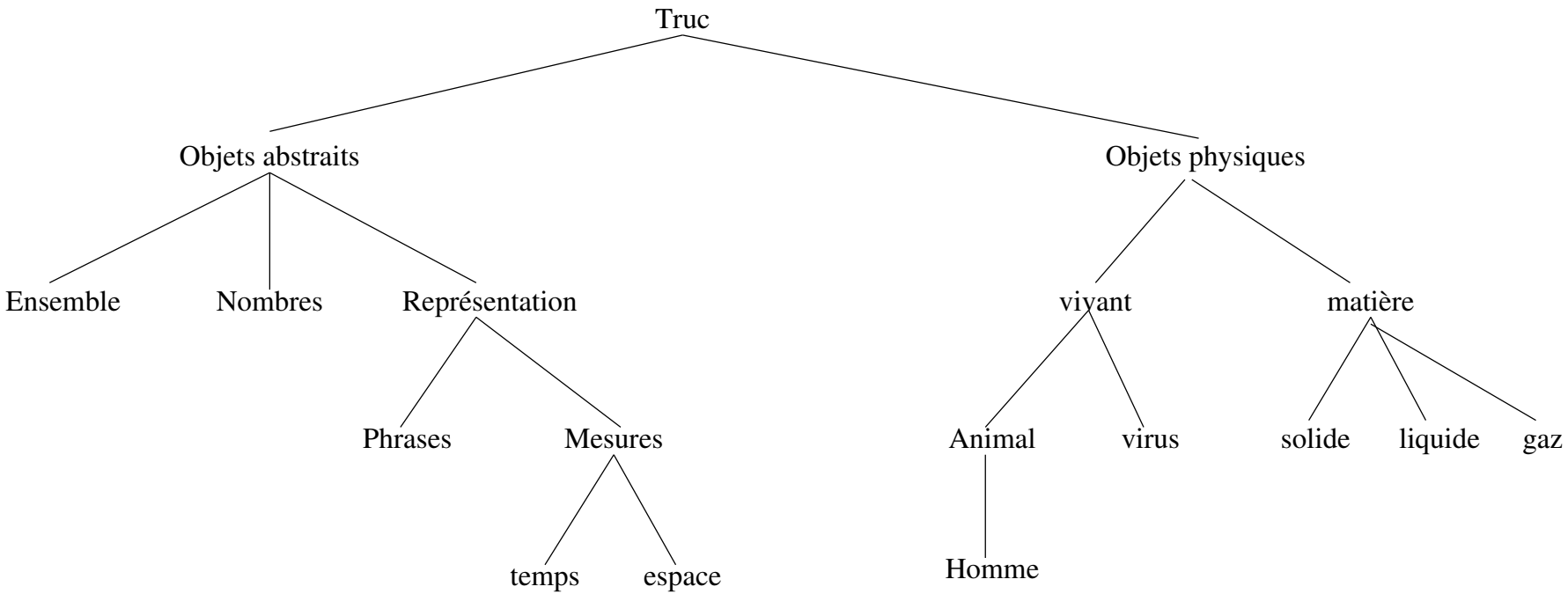
$\Rightarrow$ vers la structuration des connaissances

Représentation structurée

Objectifs :

- Développer un système qui ignore les détails inutiles ;
- Prendre en compte les relations entre objets ;
- Représenter une ontologie.
  - Ensemble des objets reconnus comme existant dans le domaine. Construire une ontologie c'est décider de la manière d'être et d'exister des objets.
  - Définir une ontologie pour la représentation des connaissances c'est définir pour un problème donné, la signature fonctionnelle et relationnelle d'un langage formel de représentation et la sémantique associée [Bachimont]

Exemple d'ontologie



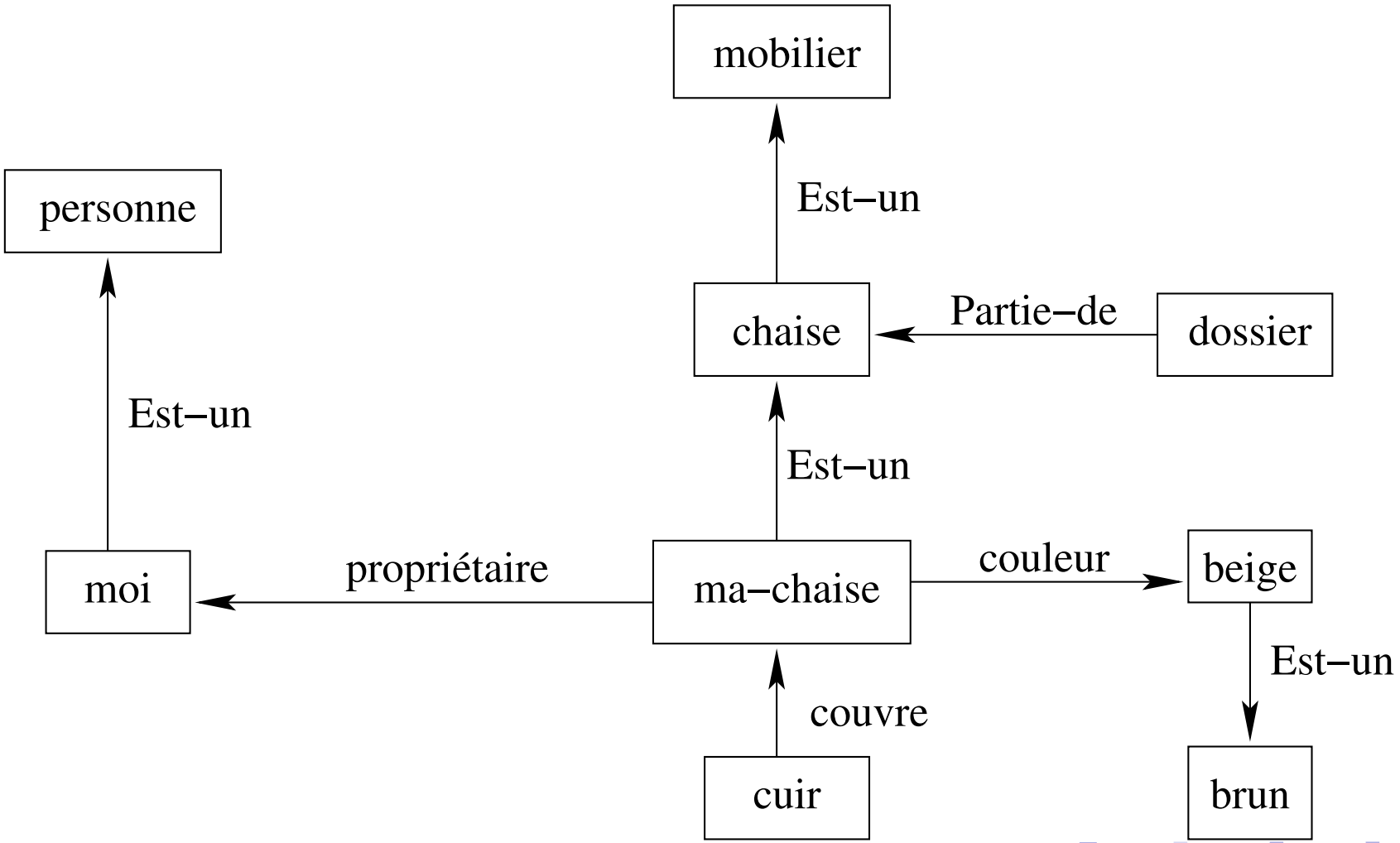
- On a défini une taxonomie hiérarchique ;
- Relation *est-un* ;

$\Rightarrow$ Réseaux sémantiques, frames.

Réseaux sémantiques

- Un réseau sémantique est un graphe orienté sans cycle dont :
  - Les nœuds sont des concepts ;
  - Les arcs sont des relations sémantiques entre les concepts
    - A est une partie de B (Meronymie).
    - B a A comme sous-partie (Holonymie).
    - A est une sorte de B (Hyponymie). Un oiseau est une sorte d'animal, oiseau est un hyponyme d'animal.
    - A est une classe contenant B (Hyperonymie). Animal est hyperonyme de oiseau.
    - A denote la même chose que B (Synonymie).
    - A dénote l'opposé de B (Antonymie).
- Représentation graphique de la connaissance.

Exemple de réseaux sémantiques



Interprétation de la relation “est-un”

Deux interprétations :

Interprétation de la relation “est-un”

Deux interprétations :

- Sous ensemble : chat est-un animal
  $\forall x \quad chat(x) \Rightarrow animal(x)$

Interprétation de la relation “est-un”

Deux interprétations :

- Sous ensemble : chat est-un animal
  $\forall x \quad chat(x) \Rightarrow animal(x)$
- Appartenance : grosMinet est-un chat
  $chat(grosMinet)$

Inférence dans un réseau sémantique

Deux orientations :

- On considère un réseau sémantique comme une conjonction de formules logiques, alors mêmes méthodes que pour un modèle logique
- On considère un réseau sémantique comme un graphe, alors on peut utiliser les techniques de propagation de marqueurs

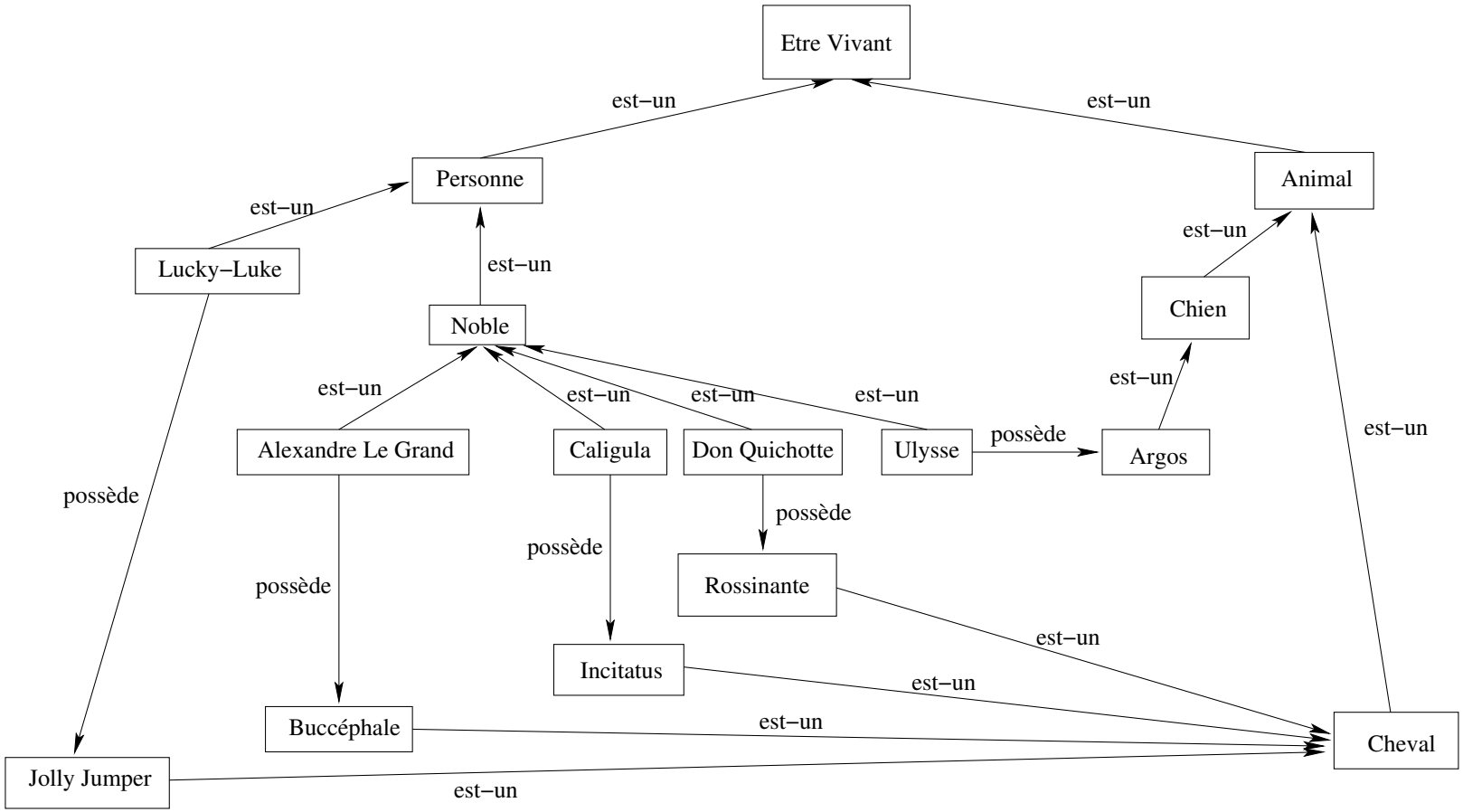
Inférence par marquage

Pour répondre à une question du genre «A est-elle nécessairement une instance de B?»

- On place un marqueur M1 sur A
  - Tant que (le réseau continue à évoluer)
    - Tout lien est-un ayant un marqueur M1 à son origine propage ce marqueur à son extrémité
    - Si le nœud B est marqué par M1, répondre «toute instance de A est nécessairement une instance de B»
- Très bonne adéquation au parallélisme ; bonne expressivité en ajoutant des liens «rôles» ; ajout de liens «de négation» ; ajout de liens «exception»
- Si on propage des valeurs à la place des marqueurs, on se rapproche sensiblement des réseaux connexionnistes ! Mécanismes d'inhibition ; activation sélective de nœuds?

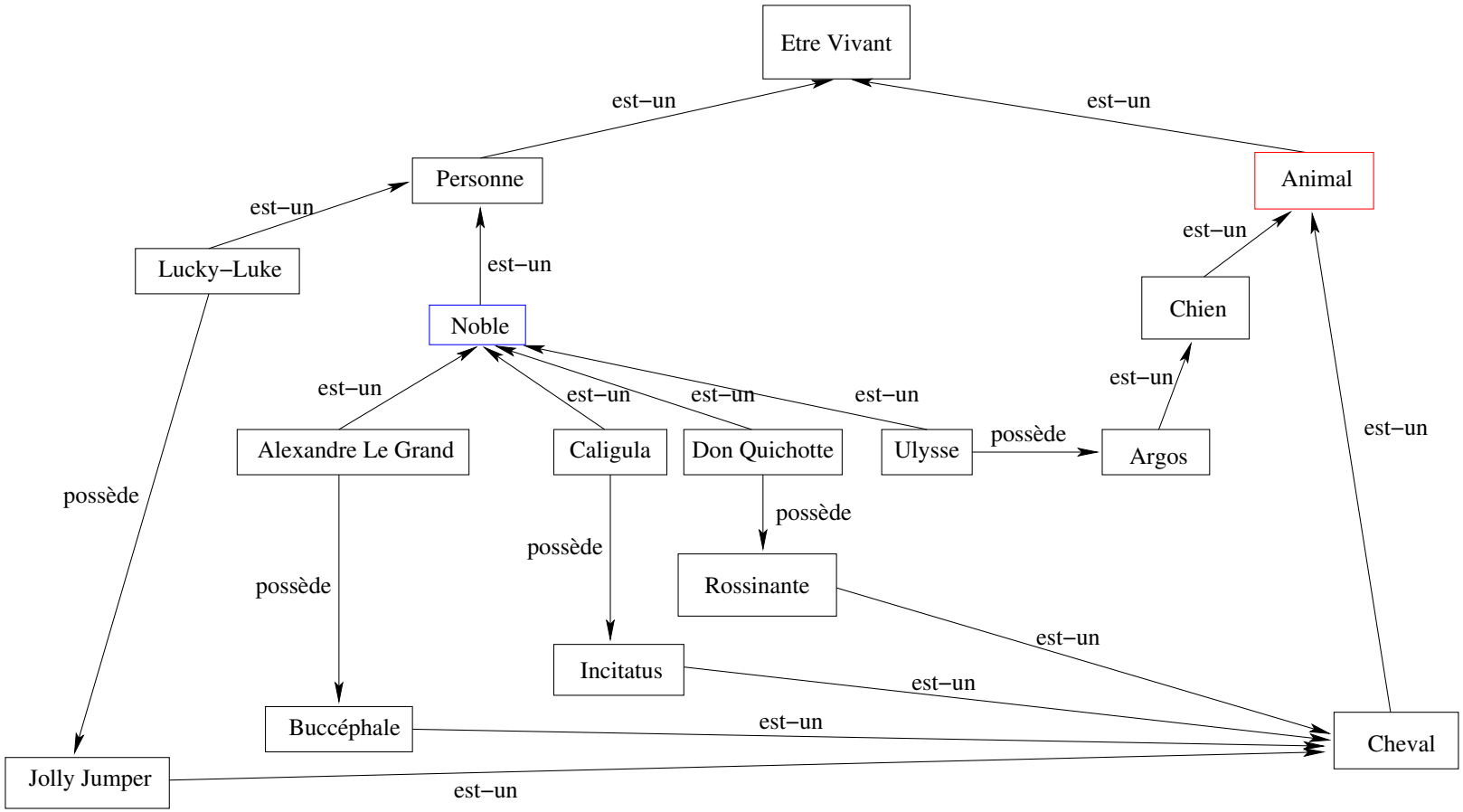
Exemple d'inférence par marquage

Nobles possédant un animal ?



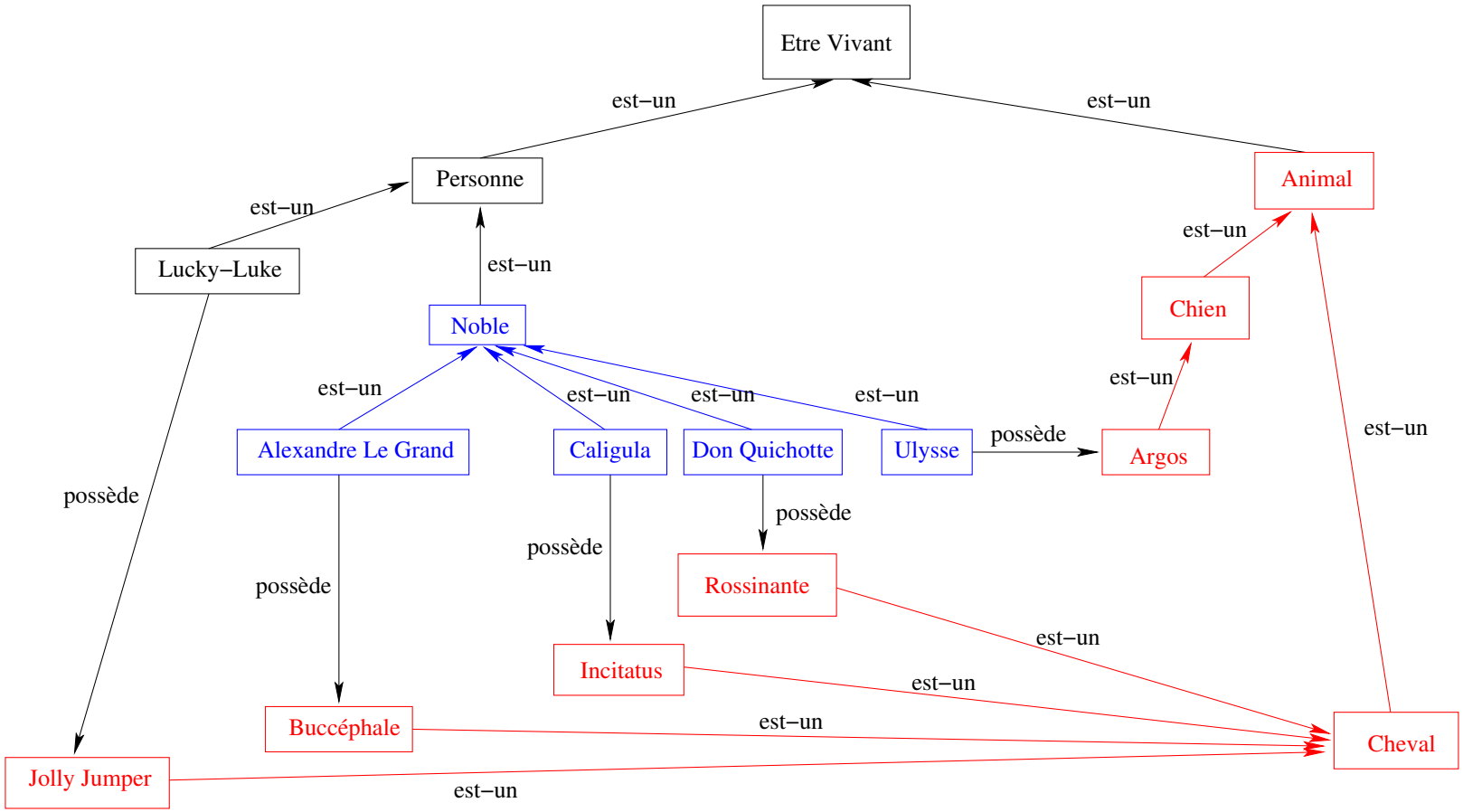
Exemple d'inférence par marquage

Nobles possédant un animal ?



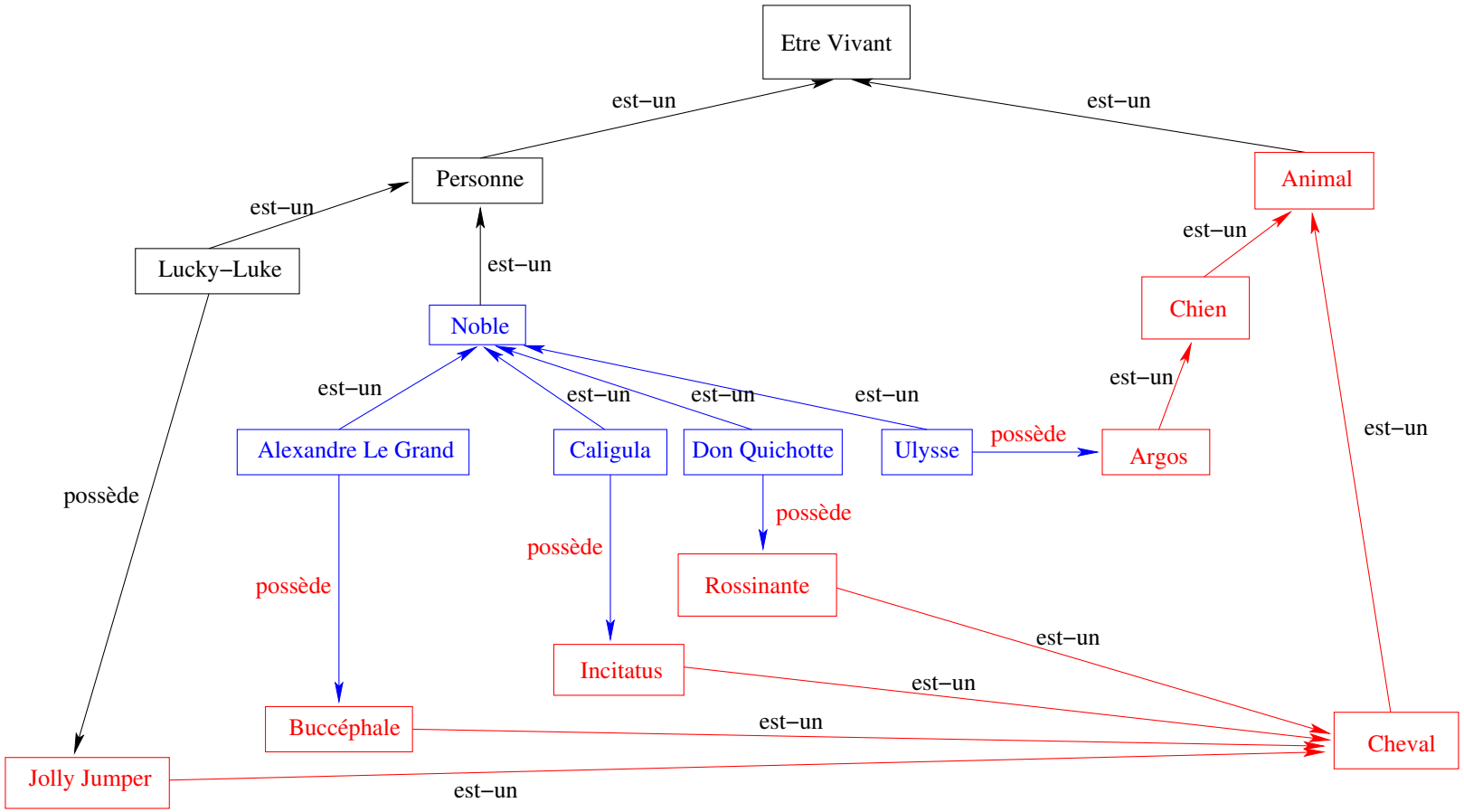
Exemple d'inférence par marquage

Nobles possédant un animal ?



Exemple d'inférence par marquage

Nobles possédant un animal ?



Les Frames de Minsky

Un frame est :

- Une structure de données qui représente des faits essentiels d'une entité structurelle quelconque.

Les Frames de Minsky

Un frame est :

- Une structure de données qui représente des faits essentiels d'une entité structurelle quelconque.
  - Il est composé de slots ;

Les Frames de Minsky

Un frame est :

- Une structure de données qui représente des faits essentiels d'une entité structurelle quelconque.
  - Il est composé de slots ;
  - Un frame est une sorte de réseau de relations et de nœuds ;

Les Frames de Minsky

Un frame est :

- Une structure de données qui représente des faits essentiels d'une entité structurelle quelconque.
  - Il est composé de slots ;
  - Un frame est une sorte de réseau de relations et de nœuds ;
  - Chaque slot peut être un frame ;

Les Frames de Minsky

Un frame est :

- Une structure de données qui représente des faits essentiels d'une entité structurelle quelconque.
  - Il est composé de slots ;
  - Un frame est une sorte de réseau de relations et de nœuds ;
  - Chaque slot peut être un frame ;
  - Chaque slot peut prendre une valeur éventuellement par défaut

Les Frames de Minsky

Un frame est :

- Une structure de données qui représente des faits essentiels d'une entité structurelle quelconque.
  - Il est composé de slots ;
  - Un frame est une sorte de réseau de relations et de nœuds ;
  - Chaque slot peut être un frame ;
  - Chaque slot peut prendre une valeur éventuellement par défaut
  - Un frame est décrit par un triplet `(frame, slot, facette)`.

Facettes

Une facette c'est une information qui exprime

- Les modalités descriptives ou comportementales du slot et/ou de sa valeur :
  - Type de données, valeur par défaut, exceptions, information incomplètes/redondantes ...
  - Différents points de vue sur le slot ;
  - Comment utiliser l'info ;
  - ....
- Demon - Procédure - Spécifie un comportement particulier :
  - `si-besoin` : méthode de calcul du slot
  - `si-ajout` : que faire si valeur ajoutée
  - ...