# Introduction aux Logiques de Description (LD)

Bernard ESPINASSE
Professeur à l'Université d'Aix-Marseille

20 Janvier 2009

- **Introduction aux Logiques de Description (LD)**
- **Logique de Description minimale $\mathcal{ALC}$**
- **Extensions des LD de la famille $\mathcal{AL}$**
- **Inférences dans les LD**



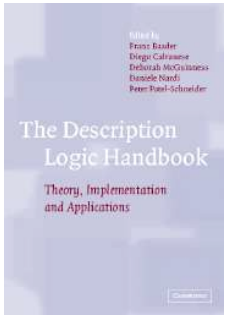
## Références principales

- **Livres et rapports :**
  - F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, The Description Logic Handbook : Theory, Implementation, and Applications, pages 47–100. Cambridge University Press, 2003.
  - A. Napoli, Une introduction aux logiques de descriptions, Rapport de recherche INRIA Loraine, N° 3314, 72 p., décembre 1997.
  - …
- **Cours et tutoriaux :**
  - Cours de R. Schmidt, Univ. Manchester, UK, 2007.
  - Cours de M. Gagnon, Ecole Polytechnique de Montréal, Canada, 2007.
  - Cours de U. Straccia, IST-CNR, Pise, Italy, 2007.
  - Cours de A. Napoli, LORIA-UMR 7503, 2008.
  - Cours de E. Franconi, Free University of Bozen-Bolzano, Italy, 2002.
  - Cours de S. Haddad, LAMSADE, Univ. Paris-Dauphine, 2005.
  - Mémoire de P. Fournier-Viger, Univ. Sherbrooke, Canada, 2005.
  - …



## Références complémentaires

**The Description Logic Handbook : Theory, Implementation and Applications**, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider,Cambridge University Press (ISBN-13: 9780521781763 | ISBN-10: 0521781760)






## Références complémentaires


- D. Nardi, R. J. Brachman. **An Introduction to Description Logics**. In the ***Description Logic Handbook***, 2002, pages 5-44.
- H. J. Levesque and R. J. Brachman. **Expressiveness and tractability in knowledge representation and reasoning**. Computational Intelligence journal 3, 78-93 (1987).
- Donini, F., Lenzerini, M., Nardi, D., Schaerf, A., **Reasoning in Description Logics**, in: Principles of Knowledge Representation and Reasoning, edited by G. Brewka; Studies in Logic, Language and Information, CLSI Publications, pp 193-238, 1996.
- Baader and U. Sattler. **An Overview of Tableau Algorithms for Description Logics**. Studia Logica, 69:5-40, 2001
- D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi**, Reasoning in expressive description logics**, in: A. Robinson and A. Voronkov, editors, Handbook of Automated Reasoning. Elsevier Science Publishers (North-Holland), Amsterdam, 2001, pages 1581-1634.
- Hollunder, B., Nutt, W., **Subsumption Algorithms for Concept Languages**, DFKI report, RR-90-04, Saarbruecken Germany, 1990; extended version of a previously published paper in proc. ECAI'90, pp 348-353.
- Schaerf, A., **Reasoning with individuals in concept languages**, Data and Knowledge Engineering, Vol 13(2), pp 141-176, 1994.

## Plan

- **1. Introduction aux Logiques de Description :**
  - Bref historique : des LD aux LDE
  - Les 2 niveaux de description : TBox et ABox
- **2. Logique de Description minimale $\mathcal{ALC}$ :**
  - Syntaxe du langage $\mathcal{ALC}$
  - Sémantique du langage $\mathcal{ALC}$
  - Interprétation dans $\mathcal{ALC}$
  - Correspondance entre $\mathcal{ALC}$ et la logique des prédicats
- **3. Extensions du langage $\mathcal{AL}$ :**
  - Ajouter de constructeurs de concepts et de rôles  Enoncer des contraintes sur l'interprétation des rôles ($\mathcal{NR}$+):
  - Extension de types primitifs ($\mathcal{D}$) et de rôles à valeurs primitives ($\mathcal{U}$) :
  - Nomenclature des langages de la famille $\mathcal{AL}$
- **4. Inférences dans les LD :**
  - Inférences aux niveaux terminologique (TBox) et factuel (ABox)
  - Comparaison de moteurs d'inférence pour LD
  - Différents types d'algorithmes d'inférences
  - La méthode des tableaux sémantiques



# 1. Introduction aux Logiques de Description

- **Bref historique : des LD aux LDE**
- **Les 2 niveaux de description : TBox et ABox**



## Introduction aux Logiques de Description (1)

- Les **Logiques de Description (LD)** sont des **langages de représentation des connaissances** mettant l'accent sur le **raisonnement**
- **L'objectif majeur** : raisonner efficacement (temps de réponse minimal) pour la prise de décision

**=> importance du rapport expressivité/performance des différentes LD**

- **Une approche ontologique** : pour représenter la connaissance d'un domaine les LD demande la définition :
  - de **catégories générales d'individus**
  - de **relations logiques** que les **individus** ou **catégories** peuvent entretenir **entre eux**.
- Cette approche ontologique est **naturelle** pour le **raisonnement** :
  - si la majorité des interactions se déroulent au niveau des individus, la plus grande partie du raisonnement se fait au niveau des catégories [Russell et Norvig, 2002]



## Historique des Logiques de Description (1)

- **Les LD s'appuient sur** : la Logique des Prédicats, les Schémas (Frames) [Minsky, 1981], les Réseaux Sémantiques [Quillian, 66], …
- **Nombreuses correspondances** : catégories générales d'objets et de relations fait partie de l'héritage dans les schémas et réseaux sémantiques.

**1° génération de LD (1980 - 1990) : langages de représentation des connaissances mettant l'accent sur le raisonnement :**

- **liées aux travaux sur les systèmes à base de connaissances** : KL-ONE [Brachman & Schmolze 85], LOOM [MacGregor & al. 86], …
- **raisonnements et inférences en temps polynomial :**
  - avec des **algorithmes** de vérification de subsomption de type **normalisation/comparaison** (structural subsumption algorithms).
  - réservés à des **LD peu expressives**, sinon **incomplets**, incapables de prouver certaines formules vraies.

## Historique des Logiques de Description (2)

**2° génération de LD (1990 à aujourd'hui) : Logiques de Description Expressives (LDE) :**

- **années 1990** : **nouveaux algorithmes** de vérification de **satisfiabilité à base de tableaux** (tableau-based algorithms) :
  - raisonnant sur des **LD dites expressives** ou **très expressives**
  - en **temps exponentiel**
  - mais en pratique, au **comportement acceptable** [Baader & al., 2003]

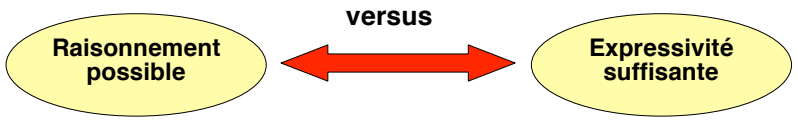
La grande expressivité des LD traitées par ces algorithmes a ouvert la porte à de **nouvelles applications** comme le **Web Sémantique** [Baader, Zou, Horrocks & al., 2003]



## Problématique générale des Logiques de Description

**Recherche d'un compromis [Baader 2000] :**

| | |
|---|---|
| Décidabilité / complexité du raisonnement | Application de concepts pertinents doit être définissable |
| Nécessite un langage de description restreint | Des domaines d'application nécesite des DL très expressives |
| Les systèmes et les résultats de complexité disponibles pour différentes combinaisons de constructeurs | Dispose-t-on en pratique d'algorithmes efficaces pour des DL très expressives ? |






## Applications des Logiques de Description

**Les systèmes à base de LD sont actuellement utilisés dans de nombreuses domaines, notamment :**

- Configuration
- Modélisation conceptuelle
- Optimisation de requêtes
- Sémantique du langage naturel
- I3 (Intelligent Integration of Information)
- Accès à l'information et interfaces intelligentes
- Terminologies et ontologies
- Logiciels de gestion
- Planning
- Web sémantique
- …

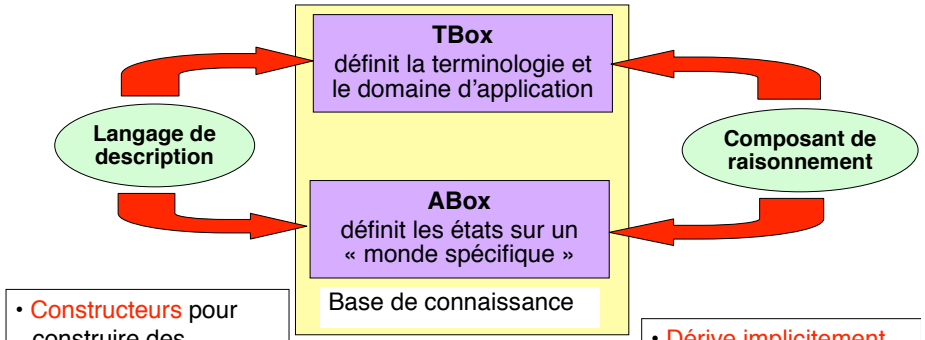


## Les 2 niveaux de description

- **Les LD permettent de représenter la connaissance d'un domaine au travers :**
  - de **concepts** (ou classes) du domaine et
  - de **relations** (ou rôles) pouvant être établies **entre** ces **classes** ou entre les **instances** de ces classes appelées **individus**
- **Modélisation des connaissances avec les LD à 2 niveaux :**
  - **le niveau terminologique ou TBox :**
    - décrit les **connaissances générales d'un domaine**
    - définit les **concepts** (classes) et les **rôles** (relations)
  - **le niveau factuel ou ABox :**
    - décrit les **individus** en les **nommant** et en spécifiant leur classes et attributs (en termes de concepts et de rôles)
    - spécifie des **assertions** portant sur ces individus nommés.
- **Remarque : plusieurs ABox** peuvent être **associées à une même TBox** :
  - chacune représente une **configuration** constituée d'individus,
  - utilise les **concepts** et **rôles** de la **TBox** pour l'exprimer.

# Architecture de systèmes à base de LD

D'après [Baader, 2000]:




- Constructeurs pour construire des concepts et des rôles complexes à partir de concepts et rôles atomiques
- Une sémantique

- Dérive implicitement de la connaissance représentée
- Algorithmes efficaces



# La TBox : entités atomiques et composées

**La TBox contient :**

- **Les Entités Atomiques : concepts atomiques** et **rôles atomiques constituant** les entités élémentaires de la LD
- **4 Concepts et Rôles atomiques prédéfinis minimaux :**
  - le **concept** $\top$ **et le rôle** $\top_R$, les plus généraux de leur catégorie
  - le **concept** $\bot$ **et le rôle** $\bot_R$, les plus spécifiques (ensemble vide)
- **Les Entités Composées :**
  - **concepts** et **rôles atomiques** peuvent être **combinés** au moyen de **constructeurs** pour former des entités **composées**

***Conventions :***

- ***A*** *et* ***B*** *dénotent des* ***concepts atomiques***
- ***C*** *et* ***D*** *dénotent des* ***concepts composés***
- ***R*** *dénote un* ***rôle***
- *les nom de* ***concepts*** *commencent par une* ***Majuscule*** *: Ex : Homme, Femme, …*
- *les noms de* ***rôles*** *par une* ***minuscule*** *: Ex : relationParentEnfant, …*



# La TBox : constructeurs et axiomes terminologiques

- **Les constructeurs** : permettent la **combinaison** de **concepts** et **rôles atomiques** pour former des **entités composées** :

  Ex : le concept composé Mâle ⊓ Femelle : résulte de l'application du constructeur ⊓ aux concepts atomiques Mâle et Femelle, et **s'interprète** comme l'ensemble des individus appartenant aux concepts Mâle et Femelle.

  Les LD se **distinguent** par les **constructeurs** qu'elles proposent :
  - plus elles ont de constructeurs, plus elles sont **expressives**, et ont des chances d'être **non décidables** ou de **complexité très élevée**
  - les LD **trop peu expressives** ne permettent pas de représenter des domaines complexes.
- **Les axiomes terminologiques d'une des 2 formes** :
  - **C ≜ D** (ou C ≡ D) : énonçant des **relations d'équivalence** (de définition) entre concepts : *C équivalent par définition à D*
  - **C ⊑ D** : énonçant des **relations d'inclusion :** *C est inclus dans D*



# La TBox : consistance et subsomption

- **Consistance de concepts** :
  - un **concept** (une classe) est **consistant**, s'il existe **au moins un individu membre de cette classe :**

    Remarque : si on définit une classe (concept) comme étant à la fois une sous-classe des classes Homme et Femme, et que la TBox spécifie aussi que ces 2 classes sont disjointes (aucun individu ne peut à la fois être un Homme et une Femme) : ce nouveau concept est alors inconsistant.

- **Subsomption de concepts** :
  - la **subsomption** consiste à **déduire qu'un concept, cad une classe, est une sous-classe d'une autre classe**, même si ce n'est pas déclaré explicitement dans la base de connaissances :

    Ex : si on spécifie que Humain est une sous-classe de Animal, que Mere est une sous classe de Humain, on peut déduire qu'une Mère est une sous-classe de Animal : Mere ⊑ Animal

## La TBox : interprétation

- **tout concept** est associé à **un ensemble d'individus** dénotés par ce concept
- **Une interprétation $\mathcal{I}$** suppose l'existence :
  - d'un **domaine d'interprétation** $\Delta$ ou $\Delta^{\mathcal{I}}$, ensemble non vide représentant les entités du monde décrit et composé d'individus
  - d'une **fonction d'interprétation** $\mathcal{I}$ ou $\cdot^{\mathcal{I}}$, assignant :
    - à chaque **concept atomique A**, un ensemble $\mathbf{A}^{\mathcal{I}}$ , tel que $\mathbf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
    - à chaque **rôle atomique R**, une relation binaire $\mathbf{R}^{\mathcal{I}}$, telle que $\mathbf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
- l'interprétation $\mathcal{I}$ **satisfait un axiome d'équivalence** $C \equiv D$ **ssi** $C^{\mathcal{I}} = D^{\mathcal{I}}$ *(égalité des ensembles d'individus)*
- l'interprétation $\mathcal{I}$ **satisfait un axiome d'inclusion** $C \sqsubseteq D$ **ssi** $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *(inclusion des ensembles d'individus).*
- l'interprétation $\mathcal{I}$ **satisfait une TBox $\mathcal{T}$ ssi $\mathcal{I}$ satisfait tous les axiomes de la TBox $\mathcal{T}$** *(on dit que $\mathcal{I}$ est un modèle de la TBox $\mathcal{T}$).*



## La ABox : assertions d'appartenance et de rôle

- **Une ABox contient 2 types d'assertions sur des individus :**
  - des **assertions d'appartenance : spécifiant** leur **classe** et leurs **attributs** :

    Ex : Marie est une femme et qu'elle a 2 enfants ; Marie est une Mère (individu instance de la classe mère)
  - des **assertions de rôle** : **spécifiant** les **relations existantes entre individus** :

    Ex : une mère doit avoir au moins un enfant : la ABox devra contenir au moins un autre individu, et une relation entre celui-ci et Marie indiquant qu'il est un de ses enfants.

***Convention :***
*les **individus nommés** sont représentés par des lettres **a, b***



## La ABox : interprétation

- Une **fonction d'interprétation $\mathcal{I}$** ou $\cdot^{\mathcal{I}}$, **associe à chaque nom d'individu nommé a**, un **individu** $\mathbf{a}^{\mathcal{I}}$ tel que $\mathbf{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ (*Les moteurs d'inférence pour LD font souvent l'hypothèse de noms uniques : pour tout individu nommé a et b, $a^{\mathcal{I}} \neq b^{\mathcal{I}}$)*
- **Interprétation d'une Assertion d'appartenance d'une ABox :**
  - étant donnée une **assertion d'appartenance notée C(a)** déclarant que pour cette ABox, il existe un individu nommé a, membre du concept C de la TBox associée : **une interprétation $\mathcal{I}$ satisfait C(a) ssi $a^{\mathcal{I}} \in C^{\mathcal{I}}$**
- **Interprétation d'une Assertion de rôle d'une ABox :**
  - étant donné une **assertion de rôle R(a, b)** déclarant que pour cette ABox, il existe un individu nommé a, en relation avec un individu nommé b par le rôle R (défini dans la TBox associée), tel que a fait partie du domaine de R et b fait partie de l'image de R : **une interprétation $\mathcal{I}$ satisfait R(a,b) ssi $(a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}}$**
- Une **interprétation $\mathcal{I}$ satisfait une ABox $\mathcal{A}$** ($\mathcal{I}$ est un **modèle de la ABox $\mathcal{A}$**) **ssi $\mathcal{I}$ satisfait toutes les assertions de $\mathcal{A}$.**



## Exemple de TBox et de ABox

Soit l'ontologie suivante (inspiré de F. Baader & W. Nutt) :

**TBox :**

Femme ≡ Personne ⊓ Femelle
Homme ≡ Personne ⊓ ¬ Femelle
Mere ≡ Femme ⊓ ∃aEnfant.Personne
Pere ≡ Homme ⊓ ∃aEnfant.Personne
Parent ≡ Pere ⊔ Mere
GrandMere ≡ Mere ⊓ ∃aEnfant.Parent
MereAvecPlusieursEnfants ≡ Mere ⊓ ≥3 aEnfant
MereSansFille ≡ Mere ⊓ ∀aEnfant.¬Femme
Epouse ≡ Femme ⊓ ∃aCommeMari.Homme

**ABox :**

MereSansFille(Marie)
Femme(Alice)
Pere(Pierre)
aEnfant(Marie,Pierre)
aEnfant(Marie,Paul)
aEnfant(Pierre,Alice)

on peut en déduire GrandMere(Marie)

# 2. Logique de Description minimale $\mathcal{ALC}$

- **Syntaxe du langage $\mathcal{ALC}$**
- **Sémantique du langage $\mathcal{ALC}$**
- **Interprétation dans $\mathcal{ALC}$**
- **Correspondance entre $\mathcal{ALC}$ et la logique des prédicats**



## La Logique Descriptive minimale $\mathcal{ALC}$

- **$\mathcal{ALC}$ (Attributive Langage with Complement)** [Schmidt-Schaub & Smolka 88], est **minimale**, dans le sens où une logique moins expressive représente peu d'intérêt
- **$\mathcal{ALC}$** est l'extension de la LD de base **$\mathcal{AL}$** à la négation de concept composé ($\mathcal{C}$ - complément), et à la quantification existentielle complète
- **$\mathcal{ALC}$** est la logique de description la plus importante, car elle constitue la base de toutes les LD pratiques
- **Signature de la LD $\mathcal{ALC}$ :**
  - **La LD $\mathcal{ALC}$ est défini par un tuple ordonné $\Sigma$ = (C, R, O) de 3 alphabets disjoints :**
    - l'ensemble **C de noms de concepts**
    - l'ensemble **R de noms de rôles**
    - l'ensemble **O de noms d'objets** (ou **noms d'individus**)
  - les noms de concepts et de rôles sont aussi appelés **concepts atomiques et rôles atomiques**



## Syntaxe de $\mathcal{AL}$

**Soit :**

- *A un concept **atomique**, C et D des concepts **atomiques ou complexes**, et R une relation (rôle)*
- $\top$ : le concept universel
- $\bot$ : le concept impossible (ou plus spécifique)

**Constructeurs d'$\mathcal{AL}$:**

| | |
|---|---|
| **¬ A** | la négation atomique |
| **C ⊓ D** | l'intersection de concepts |
| **∀R.C** | la restriction de valeur (quantification universelle complète) |
| **∃R.⊤** | la quantification existentielle limitée * |

(*) : Ex : Personne ⊓ ∃aEnfant.⊤ : personne ayant au moins un enfant



## Syntaxe de $\mathcal{ALC}$

**Soit :**

- *C et D des concepts **atomiques ou complexes**, et R une relation (rôle)*
- $\top$ : le concept universel
- $\bot$ : le concept impossible (ou plus spécifique)

**Constructeurs d'$\mathcal{ALC}$:**

| | |
|---|---|
| **¬ C** | non C ou Complément de C |
| **C ⊔ D** | l'union de concepts (C **ou** D) - ($\mathcal{ALC}$-$\mathcal{U}$) |
| **C ⊓ D** | l'intersection de concepts (C **et** D) |
| **∃R.C** | la quantification existentielle (existential restriction)* |
| **∀R.C** | la quantification universelle (universal restriction) |

(*) : Ex :

- ∃aEnfant.Personne: personne ayant au moins un enfant
- ∃aEnfant.Femme: personne ayant au moins une fille

## Syntaxe de $\mathcal{ALC}$ : constructeurs (1)

**Signification intuitive des symboles :**

| Symbole/expression | Signification |
|---|---|
| **concept** | ensemble |
| **rôle** | relation binaire |
| ¬ | ensemble complémentaire |
| ⊔ | ensemble union ($\mathcal{ALCU}$) |
| ⊓ | ensemble intersection |

- **constructeur ¬C : négation (complément) d'un concept** désignant (pour une interprétation), l'ensemble des individus n'appartenant pas au concept atomique C :

  Ex : soit le concept Humain représentant l'ensemble des humains, ¬Humain représente l'ensemble des individus qui ne sont pas des humains



## Syntaxe de $\mathcal{ALC}$ : constructeurs (2)

- **constructeur C ⊔ D** : disjonction (union) de 2 concepts composés désignant (pour une interprétation), l'ensemble des individus membres soit du concept C ou soit du concept D

  Ex : Etudiants ⊔ Enseignant représentant l'ensemble des individus qui sont étudiants OU enseignants

- **constructeur C ⊓ D** : conjonction (intersection) de 2 concepts composés désignant (pour une interprétation) l'ensemble des individus membres à la fois du concept C et du concept D

  Ex : Etudiants ⊓ Male représentant l'ensemble des individus qui sont étudiants ET males



## Syntaxe de $\mathcal{ALC}$ : constructeurs (3)

- **quantificateur existentiel ∃R.C** : désigne (pour une interprétation), l'ensemble des individus, membres du domaine d'un rôle R

  Ex : dans l'interprétation $\mathcal{I}$, modèle de l'ABox et la TBox précédentes (page 20), ∃aEnfant est l'ensemble des individus {Pierre$^{\mathcal{I}}$, Paul$^{\mathcal{I}}$, Alice$^{\mathcal{I}}$}.

- **quantificateur universel ∀R.C :** désigne (pour une interprétation), l'ensemble des individus du domaine d'un rôle R qui sont en relation par R avec un individu du concept C, pour une interprétation donnée.

  Ex : pour la même interprétation $\mathcal{I}$, ∀aEnfant.Femme est l'ensemble des individus {Alice$^{\mathcal{I}}$}.

- **ensemble** ⊤ : désigne (pour une interprétation) l'ensemble de tous les objets/individus

- **ensemble** ⊥ : désigne (pour une interprétation) vide

**Remarque** : $\mathcal{AL}$ **ne permet pas** la **spécification** de **rôles** à l'aide de **constructeurs** (pas de rôles composés).



## Exemple de représentation des connaissances avec $\mathcal{ALC}$ (1)

- Pour déclarer « **un humain est un animal** », on peut utiliser les concepts atomiques Humain et Animal et déclarer l'axiome :

  Humain ⊑ Animal *(Humain est inclus dans Animal)*

- Pour déclarer « **un humain est un animal qui raisonne** », on peut définir le concept Raisonnable et déclarer l'axiome :

  Humain ≡ Animal ⊓ Raisonnable

  **Il y a l'équivalence entre :**

  - le concept Humain représentant l'ensemble des humains,
  - le concept Animal ⊓ Raisonnable représentant l'ensemble des individus appartenant à la fois à la classe Animal et à la classe Raisonnable.

## Exemple de représentation des connaissances avec $\mathcal{ALC}$ (2)

- $\mathcal{AL}$ possède la **négation**, seulement appliquée qu'à un **concept atomique** : la classe des non humains est : ¬Humain
- $\mathcal{ALC}$ possède la **négation** appliquée à un **concept atomique** ou **composé** : la classe des individus qui ne sont pas des animaux raisonnables est : ¬(Animal ⊓ Raisonnable)
- $\mathcal{ALC}$ permet de **définir un concept** par **restrictions** sur des **relations** (rôles) :
  - ∀aEnfant.Femme définit la classe des individus dont tous les enfants sont des femmes
  - ∃aEnfant.Femme définit la classe des individus dont au moins un enfant est une femme



## Exemple de représentation des connaissances avec $\mathcal{ALC}$ (3)

- **Personne ayant au moins un enfant :**

  ∃aEnfant.Humain

- **Personne qui n'a que des filles** peut être défini ainsi :

  ∀aEnfant.Femme

- **Personne qui n'a pas d'enfant** : on restreint la valeur de la relation aEnfant au concept impossible :

  ∀aEnfant.⊥

  pour appartenir à ce concept, un individu doit avoir tous ses enfants appartenant au concept impossible : il ne peut ainsi avoir d'enfant.



## Syntaxe de $\mathcal{ALC}$ : axiomes terminologiques

- **Les axiomes terminologiques sont de la forme :**
  - **C ⊑ D**

  **ou**

  - **C ≜ D**

  avec C et D dénotant des concepts
- Les **définitions de concepts** sont des **axiomes terminologiques** dans lesquels la partie gauche sont des **noms de concepts (ou concepts atomiques)**
- Ils permettent aussi d'**exprimer des propriétés de concepts et de rôles**



## Syntaxe de $\mathcal{ALC}$ : axiomes terminologiques et définitions de concepts

**Soient A un concept atomique et C un concept composé :**

- **Les définitions de concept sont des instructions de la forme :**
  - **A ≜ C** (ou **A ≡ C)** :

    *lire « **A est par définition égal à C** »*

  **ou**
  - **A ⊑ C** :

    *lire « **A est par définition inclus dans C** » ou « **A est par définition subsumé par C** »*
- **Les définitions de concepts de la forme A ⊑ C** sont aussi appelées « définitions primitives de concept » (primitive concept definitions)

**Ex :**

Etudiant ≜ Personne ⊓ ∃aNom.string
⊓ ∃aAdresse.string
⊓ ∃inscritA.ProgrammeFormation

ProgrammeFormation ⊑ ∃composeDe.ModuleFormation

## Syntaxe de $\mathcal{ALC}$ : axiomes terminologiques et propriétés de concepts et de rôles

- **Disjonction de concepts** (disjointness) :
  Homme $\sqsubseteq \neg$ Femme
  *l'intersection des individus hommes et des individus femmes est vide*
- **Couverture** (coverings) :
  $\top \sqsubseteq$ Homme $\sqcup$ Femme
  *un individu est nécessairement un homme ou une femme*
- **Restriction de domaine** (restriction) :
  $\exists$aEnfant.$\top \sqsubseteq$ Parent
  *un parent a au moins un enfant*
- **Plages de restriction ou image** (range restrictions) :
  $\top \sqsubseteq \forall$aEnfant.Personne
  *tout enfant est une personne*
  $\top \sqsubseteq \forall$aFils.Personne
  *tout fils est une personne*



## Syntaxe de $\mathcal{ALC}$ : axiomes terminologiques

- On a la possibilité de **substituer tout nom de concept** (concept atomique) **par sa définition** dans n'importe quelle **expression conceptuelle** (développement des définitions) :
  Ex: Parent $\triangleq$ Pere $\sqcup$ Mere
- On dispose du concept « top » $\top$ qui est satisfait par tout objet et du rôle « top-rôle » $R^{\top}$ qui est satisfait par tout couple d'individus

**On a les relations importantes entre concepts suivantes :**

$\models \exists R.\bot \triangleq \bot$
$\models \exists R.(C \sqcup D) \triangleq \exists R.C \sqcup \exists R.D$
$\models \exists R.(C \sqcap D) \triangleq \exists R.C \sqcap \exists R.D$
$\models \exists R.C \triangleq \neg\exists R.\neg C$
$\models \forall R.\top \triangleq \top$

Ces relations sont vraies ($\models$) dans n'importe quelle interprétation terminologique



## Syntaxe de $\mathcal{ALC}$ : assertions de concepts et de rôles

Soit C un concept, R un nom de rôle et a et b des individus.

les **assertions de concepts** sont de la forme **:**

**a : C** : ***a appartient à la classe C***

les **assertions de rôles** sont de la forme **:**

**(a, b) : R** : ***(a,b) appartient au rôle R***

**Ex :**

Eric : Etudiant
MasterM6 : Programme
(Eric, MasterM6) : estInscrit



## Syntaxe d' $\mathcal{ALC}$ : base de connaissance

- Une **base de connaissance est une paire ($\mathcal{T}$, $\mathcal{A}$)** où $\mathcal{T}$ est une TBox et $\mathcal{A}$ une ABox se référant à $\mathcal{T}$
- **Exemple :**

**TBox :**
Parent $\triangleq$ Personne $\sqcap \exists$aEnfant.Personne
Pere $\triangleq$ Parent $\sqcap$ Male
GrandParent $\triangleq$ Personne $\sqcap \exists$aEnfant.Parent

**ABox :**

| | |
|---|---|
| Paul : Personne | Paul : Male |
| Pierre : Personne | Alice : Personne |
| (Paul, Pierre) : aEnfant | (Pierre, Alice) : aEnfant |

## Sémantique d' $\mathcal{ALC}$ : interprétation

**Une interprétation terminologique $\mathcal{I}$** d'une LD **(O, C, R)** consiste en :

- Un **domaine d'interprétation Δ**, ensemble **non vide,** représentant des entités du monde décrit
- Une **fonction d'interprétation $\mathcal{I}$** associant :
  - à tout individu a ∈ **O**, $\mathcal{I}$ associe un sous-ensemble $\mathcal{I}(a) \in \Delta$
  - à tout concept atomique A ∈ **C**, $\mathcal{I}$ associe un sous-ensemble $\mathcal{I}(A) \subseteq \Delta$
  - à tout rôle atomique R ∈ **R**, $\mathcal{I}$ associe une relation binaire $\mathcal{I}(R) \subseteq \Delta \times \Delta$
- les **autres descriptions possibles de cette fonction $\mathcal{I}$** sont définies par :

$$\begin{aligned}
&\mathcal{I}(\top) = \Delta \\
&\mathcal{I}(\bot) = \varnothing \\
&\mathcal{I}(\neg C) = \Delta \setminus \mathcal{I}(C) \\
&\mathcal{I}(C \sqcap D) = \mathcal{I}(C) \cap \mathcal{I}(D) \\
&\mathcal{I}(C \sqcup D) = \mathcal{I}(C) \cup \mathcal{I}(D) \\
&\mathcal{I}(\forall R.C) = \{a \in \Delta \mid \forall b.(a, b) \in \mathcal{I}(R) \rightarrow b \in \mathcal{I}(C)\} \\
&\mathcal{I}(\exists R.C) = \{a \in \Delta \mid \exists b.(a, b) \in \mathcal{I}(R) \rightarrow b \in \mathcal{I}(C)\}
\end{aligned}$$

- et **l'hypothèse de nom unique** des individus : $\forall$ a, b ∈ **O** on a $\mathcal{I}(a) \neq \mathcal{I}(b)$



## Sémantique d' $\mathcal{ALC}$ : interprétation

**On a l'interprétation des symboles suivants :**

| LD | Interprétation sur Δ |
|---|---|
| Nom d'objet/individu | élément de Δ |
| Nom d'objet/individu | Sous-ensemble de Δ |
| Nom de rôle | Relation binaire sur Δ |

**On dit** :

- si **d** ∈ Δ et o est un nom objet/individu avec **$\mathcal{I}$(o) = d**, alors on dit que **o dénote d** ou **d est une interprétation de o**
- par l'hypothèse de nom unique des individus, il ne peut y avoir qu'un nom d'objet dénotant un élément de Δ



## Sémantique d'$\mathcal{ALC}$ : équivalence de concepts

- On dit que **2 concepts C et D sont équivalents, noté C ≜ D** (C ≡ D), **si on a $\mathcal{I}$(C) = $\mathcal{I}$(D),** quelle que soit l'interprétation $\mathcal{I}$

  Ex : l'équivalence :

  ∀aEnfant.Femme⊓∀aEnfant.Médecin ≜ ∀aEnfant.(Femme⊓Médecin)

  *l'ensemble des personnes dont tous les enfants sont des femmes, et dont tous les enfants sont des médecins, est exactement le même que (équivalent à) l'ensemble des personnes dont tous les enfants sont à la fois femme et médecin.*

  **Tout énoncé de la forme C ≜ D** est appelé dans la TBox **définition**

- **Par définition on a les équivalences suivantes :**

$$\begin{aligned}
&\neg\top \triangleq \bot \\
&\neg\bot \triangleq \top \\
&C \sqcap \neg C \triangleq \bot \\
&C \sqcup \neg C \triangleq \top
\end{aligned}$$



## Sémantique d' $\mathcal{ALC}$ : inclusion de concepts

- On dit que **le concept C inclus le concept D**, **noté C ⊑ D** , ssi on a **$\mathcal{I}$(C) ⊆ $\mathcal{I}$(D), quelle que soit l'interprétation $\mathcal{I}$**
- **Par définition :**

  **soit le concept universel** ⊤ représentant tous les individus du monde représenté, et le **concept impossible** ⊥

  - **pour tout concept C**, on a **l'axiome :**

    **C ⊑ ⊤**
  - **pour un concept C *impossible***, cad qu'aucun individu ne peut appartenir à ce concept, on a **l'axiome :**

    **C ⊑ ⊥**

## Sémantique d'$\mathcal{ALC}$ : exemple d'interprétation

**Soit :**

Personne et Male : 2 concepts (2 noms de concepts)

aEnfant : un role (un nom de role)

**Soit l'interprétation terminologique $\mathcal{I}$ sur $\Delta$ :**

$\Delta$ = {Paul, Pierre, Eric, Alice, Lila}

$\mathcal{I}$ (Personne) = {Paul, Pierre, Eric, Alice}

$\mathcal{I}$ (Male) = {Paul, Pierre, Eric }

$\mathcal{I}$ (aEnfant) = {(Paul, Pierre), (Pierre, Alice), (Pierre, Eric)}

**On en déduit :**

$\mathcal{I}$ (¬Male) = {Alice, Lila }

$\mathcal{I}$ (Personne ⊓ ¬Male) = {Alice}

$\mathcal{I}$ (∃aEnfant.Male) = {Paul, Pierre}



$\mathcal{I}$ (∀aEnfant.Male) = {Pierre, Eric, Alice, Lila }

## Correspondance entre $\mathcal{ALC}$ et la logique des prédicats

- Une **correspondance** existe entre la **LD $\mathcal{AL}$** et la **logique des prédicats du premier ordre** (First Order Logic - FOL) [Baader et Nutt, 2003] :
  - un **concept atomique A** correspond à un **prédicat unaire** $\phi A(x)$
  - un **rôle R** à un **prédicat binaire** $\phi R(x,y)$
  - un **individu** correspond à une **constante**
  - un **concept composé** à une **formule avec 1 variable libre** $\phi C(x)$.
- **avec les règles de passage suivantes :**

| *Constructeurs AL* | | *Logique des prédicats (FOL)* |
|---|---|---|
| $\phi_{\neg C}(x)$ | = | $\neg\phi_C(x)$ |
| $\phi_{C \sqcap D}(x)$ | = | $\phi_C(x) \wedge \phi_D(x)$ |
| $\phi_{C \sqcup D}(x)$ | = | $\phi_C(x) \vee \phi_D(x)$ |
| $\phi_{\exists R.C}(y)$ | = | $\exists x.R(y,x) \wedge \phi_C(x)$ |
| $\phi_{\forall R.C}(y)$ | = | $\forall x.R(y,x) \rightarrow \phi_C(x)$ |



## Correspondance entre $\mathcal{ALC}$ & FOL : exemple

- Tout employé travaille pour une compagnie :
  - ∀E.∃C.(Employe(E) → travaillePour(E,C) ∧ Compagnie(C)
  - Employe ⊑ ∃travaillePour.Compagnie
- Une compagnie a au moins un employé :
  - ∀C.(Compagnie(C) → ∃E.(travaillePour(E, C)))
  - Compagnie ⊑ ∃travaillePour$^{-}$.Employe
- Un manager est un employé :
  - ∀X.(Manager(X) → Employe(X))
  - Manager ⊑ Employe
- Un manager ne doit pas travailler pour plus que 2 compagnies :
  - ∀M.∀X.∀Y.∀Z.(Manager(M) ∧ travaillePour(M,X) travaillePour(M, Z) → (X = Y) ∨ (X = Z) ∨ (Y = Z))
  - Manager ⊑ ≤2 travaillePour.Compagnie
- Une compagnie ne peut pas être un employé en même temps :
  - ∀X.(Compagnie(X) → ¬Employe(X))
  - ⊥ ≡ Compagnie ⊓ Employe
- Pour tout employé travaillant pour une compagnie, on peut automatiquement déduire que la compagnie l'emploie :
  - ∀E.∀C.(travaillePour(E, C) → employer(C, E)
  - travaillePour$^{-}$ ≡ employer



# 3. Les extensions de $\mathcal{AL}$

- **Ajouter de constructeurs de concepts et de rôles**
- **Enoncer des contraintes sur l'interprétation des rôles ($\mathcal{NR}$+):**
- **Extension de types primitifs ($\mathcal{D}$) et de rôles à valeurs primitives ($\mathcal{U}$) :**
- **Nomenclature des langages de la famille $\mathcal{AL}$**

# Rappel du langage $\mathcal{AL}$

**Soit :**

- *A un concept **atomique**, C et D des concepts **atomiques ou complexes**, et R une relation (rôle)*
- $\top$ : le concept universel
- $\bot$ : le concept impossible (ou plus spécifique)

## Constructeurs d'$\mathcal{AL}$:

| | |
|---|---|
| **¬ A** | la négation atomique |
| **C ⊓ D** | l'intersection de concepts |
| **∀R.C** | la restriction de valeur (quantification universelle complète) |
| **∃R.⊤** | la quantification existentielle limitée * |

(*) : Ex : Personne ⊓ ∃aEnfant.⊤ : personne ayant au moins un enfant



# Les extensions du langage $\mathcal{AL}$

**Différentes façons d'étendre $\mathcal{AL}$ [Baader, 2003] :**

- **Ajouter de constructeurs de concepts et de rôles :**
  - $\mathcal{O}$ : permet la description de concepts par l'énumération d'individus nommés,
  - $\mathcal{U}$ : permet l'union de concepts arbitraires,
  - $\mathcal{E}$ : permet la quantification existentielle complète,
  - $\mathcal{C}$ : permet la négation complète,
  - $\mathcal{I}$ : permet les rôles inverses et l'inclusion entre rôles,
  - $\mathcal{F}, \mathcal{Q}, \mathcal{N}$ : 3 variantes de la contrainte de cardinalité sur rôle.
- **Enoncer des contraintes sur l'interprétation des rôles ($\mathcal{NR}$+):**
  - spécification d'un **ensemble de rôles transitifs** $\mathcal{NR}$+ , constitue $\mathcal{R}$+, une extension par ajout de contraintes sur l'interprétation des rôles (désignée par la lettre $\mathcal{R}$+) : permet des rôles transitifs tels que ancêtreDe ou frèreDe.
- **Extension de types primitifs ($\mathcal{D}$) et de rôles à valeurs primitives ($\mathcal{U}$) :**
  - $\mathcal{D}$**:** Ajout à $\mathcal{AL}$ d'un **second domaine d'interprétation** $\Delta^I\mathcal{D}$ disjoint avec $\Delta^I$, représentant l'ensemble des valeurs de type primitif (entiers, chaînes de caractères, entiers positifs …), dont les éléments sont des individus primitifs
  - $\mathcal{U}$ **:** un **nouveau type de rôle** défini comme une relation binaire sur $\Delta^I\mathcal{D}\times\Delta^I\mathcal{D}$, appelé rôles à valeurs primitives. La lettre $\mathcal{U}$ représente l'ensemble de ces rôles, permettant par la spécification d'assertions de rôle telles que u(a, 205006007) et v(a, "Jean Jacques") où u, v$\in\mathcal{U}$.



# Ajout de constructeurs au langage $\mathcal{AL}$ (1)

- colonne 1 : lettre désignant le constructeur,
- colonne 2 : sa syntaxe d'utilisation
- colonne 3 : sa sémantique.

| | | |
|---|---|---|
| $[\mathcal{O}]$ | $\{a_1, a_2, ..., a_n\}$ | $\{a_1^{\mathcal{I}}, a_2^{\mathcal{I}}, ..., a_n^{\mathcal{I}}\}$ |
| $[\mathcal{U}]$ | $C \sqcup D$ | $C^{\mathcal{I}} \cup D^{\mathcal{I}}$ |
| $[\mathcal{E}]$ | $\exists R.C$ | $\{a \in \triangle^{\mathcal{I}} \mid \{\exists b.(a,b) \in R^{\mathcal{I}}\} \wedge b \in C^{\mathcal{I}}\}$ |
| $[\mathcal{C}]$ | $\neg C$ | $\triangle^{\mathcal{I}} \backslash C^{\mathcal{I}}$ |
| $[\mathcal{I}]$ | $R_1^{-1}$ | $\{(y,x) \mid (x,y) \in R_1^{\mathcal{I}}\}$ |
| $[\mathcal{H}]$ | $R_1 \sqsubseteq R_2$ | $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$ |
| $[\mathcal{F}]$ | $= 1R$ | $\{x \in \triangle^{\mathcal{I}} \mid \mid \{y \in \triangle^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}}\}\mid = 1\}$ |
| | $\geq 2R$ | $\{x \in \triangle^{\mathcal{I}} \mid \mid \{y \in \triangle^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}}\}\mid \geq 2\}$ |
| $[\mathcal{N}]$ | $\geq nR$ | $\{a,b \in \triangle^{\mathcal{I}} \mid \mid (a,b) \in R^{\mathcal{I}}\}\mid \geq n\}$ |
| | $\leq nR$ | $\{a,b \in \triangle^{\mathcal{I}} \mid \mid (a,b) \in R^{\mathcal{I}}\}\mid \leq n\}$ |
| | $= nR$ | $\{a,b \in \triangle^{\mathcal{I}} \mid \mid (a,b) \in R^{\mathcal{I}}\}\mid = n\}$ |
| $[\mathcal{Q}]$ | $\geq nR.C$ | $\{a,b \in \triangle^{\mathcal{I}} \mid \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\mid \geq n\}$ |
| | $\leq nR.C$ | $\{a,b \in \triangle^{\mathcal{I}} \mid \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\mid \leq n\}$ |
| | $= nR.C$ | $\{a,b \in \triangle^{\mathcal{I}} \mid \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\mid = n\}$ |



# Ajout de constructeurs au langage $\mathcal{AL}$ (2)

- $\mathcal{O}$ **: Description de concepts par l'énumération d'individus nommés :**
  - individus nommés $\{a_1, a_2, \ldots a_n\}$, et interprétation $\{a^I_1, a^I_2, \ldots a^I_n\}$,
- $\mathcal{U}$ **: Union de concepts arbitraires :**
  - pour représenter l'ensemble des individus qui appartiennent soit à la classe C, soit à la classe D, on écrit la description **C ⊔ D**, et dont l'interprétation est la suivante : $\mathcal{I}$(C ⊔ D) = $\mathcal{I}$(C ) ∪ $\mathcal{I}$(D)
  - très pratique dans les **restrictions de relations** :

    Ex : représenter un magasin qui ne vend que des chats et des chiens :

    **Magasin ⊓ ∀vend.(Chat ⊔ Chien)**

    *pour appartenir à cette classe, une entité doit appartenir à la fois à l'ensemble des magasins et l'ensemble des entités telles que si elles vendent quelque chose, il s'agit nécessairement d'un chat ou d'un chien.*
- $\mathcal{C}$ **: Négation complète :** $\mathcal{ALC}$
  - ¬ C avec C concept atomique ou composé

## Ajout de constructeurs au langage $\mathcal{AL}$ (3)

- **$\varepsilon$ : Quantification existentielle ($\exists$) complète : $\mathcal{ALC}$**
  - Limitation dans $\mathcal{AL}$ : $\exists$ permet de spécifier qu'une entité doit avoir au moins une relation avec un autre objet, mais pas possible de spécifier la classe de cet autre objet

    Ex : définir la classe de ceux qui possèdent au moins un chien.

    ⇒ ***Ajout de la quantification existentielle complète*** **$\exists$R.C**, dont l'interprétation est : $\mathcal{I}(\exists R.C) = \{a \in \Delta \mid \exists b.(a, b) \in \mathcal{I}(R) \wedge b \in \mathcal{I}(C)\}$

    Ex : classe des gens possèdant au moins un chien : **$\exists$possède.Chien**

- **$\mathcal{I}$ : Rôles inverses et l'inclusion entre rôles :**
  - permet de définir un rôle qui est l'inverse d'un autre rôle.

    Ex : si Alice regarde Paul, Paul est regardé par Alice. Si on utilise 2 relations regarde et estRegardéPar, on veut que tout fait de la forme regarde(x, y) implique nécessairement le fait estRegardéPar(y, x)

    ⇒ ***Ajout du constructeur d'inversion*** **estRegardéPar $\equiv$ regarde$^-$**, et dont l'interprétation est $\mathcal{I}(R^-) = \{(y, x) \mid (x, y) \in \mathcal{I}(R)\}$



## Ajout de constructeurs au langage $\mathcal{AL}$ (4)

- **$\mathcal{F}$ : Constructeur de fonctions**
  - permet de spécifier qu'un rôle (relation) est une fonction : aucune entité ne peut être reliée à plus d'une autre entité par cette relation.

    Ex : la relation mariéAvec, qui nous permettrait de définir le concept HommeMarié : **HommeMarié $\equiv$ Homme $\sqcap$ $\exists$mariéAvec.$\top$**

    ⇒ Pour empêcher une instance de cette classe d'être mariée avec plus d'une personne, il faut ***spécifier que la relation mariéAvec est une fonction***, en écrivant un axiome de la forme ***Fun(R)*** indiquant que le rôle R est une fonction.

- **$\mathcal{N}$ : Restriction de cardinalité**
  - pour représenter des concepts comme l'ensemble de ceux qui ont au moins 2 enfants, ou qui ont au plus 4 enfants

    ⇒ ***Ajout de 2 constructeurs $\leq n\ R$ et $\geq n\ R$ :***

    Ex : concept de père qui a exactement 2 enfants :

    **Homme $\sqcap \geq$ 2 aEnfant $\sqcap \leq$ 2 aEnfant**



## Ajout de constructeurs au langage $\mathcal{AL}$ (5)

- **$\mathcal{Q}$ : Restriction de cardinalité qualifiée**
  - Les 2 contructeurs **$\leq$ nR** et **$\geq$ nR** précédents ne permettent pas d'imposer le nombre minimal ou maximal d'entités d'une classe spécifique auquel on peut être lié par une relation

    Ex : dans une logique $\mathcal{ALN}$, si on a la relation possède, on ne peut décrire la classe des gens qui possèdent plus de 2 chiens

  - Pour cela, il faut les **constructeurs de restriction de cardinalité qualifiée** $\mathcal{Q}$ suivants : **$\leq$ nR.C** et **$\geq$ nR.C**

    Ex : classe des gens qui possèdent 2 chiens ou plus :

    **$\geq$ 2 possède.Chien**

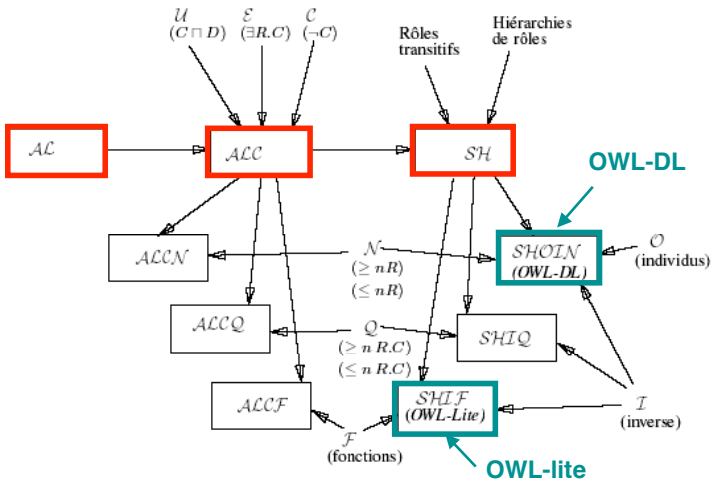


## Nomenclature des langages de la famille $\mathcal{AL}$

- **De façon générale, pour chaque constructeur ajouté**, il faut rajouter la lettre correspondante au nom de la logique originale
- **La LD $\mathcal{ALU\varepsilon}$** est obtenue en rajoutant à la LD $\mathcal{AL}$ :
  - l'union ($\mathcal{U}$) et
  - la quantification existentielle complète ($\varepsilon$)
- **La LD $\mathcal{ALC}$ équivaut à la LD $\mathcal{ALU\varepsilon}$ :** car l'union et la quantification existentielle complète s'expriment par la négation complète et inversement : $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ et $\exists R.C \equiv \neg \forall R.\neg C$ [Baader, 2003]
- **La LD $\mathcal{SH}$ :** LD plus expressive obtenue en rajoutant à la LD $\mathcal{ALC}$ les possibilités :
  - d'établir qu'une relation est une sous-propriété d'une autre relation et
  - de définir une relation transitive
  - on écrira ***Tr(R)*** pour signifier qu'une relation R est transitive, et $R_1 \sqsubseteq R_2$ pour signifier que $R_1$ est une sous-propriété de $R_2$.

# Nomenclature des langages de la famille $\mathcal{AL}$

**« Photo » de famille des LD $\mathcal{AL}$, $\mathcal{ALC}$ et $\mathcal{SH}$** (d'après Gagnon) :






# Exemple d'ontologie (1)

**Ontologie 0 :**

**TBox :**

AnimalDeCompagnie ≡ AnimalDomestique ⊓ (Chien ⊔ Chat)
(Chien ⊓ Chat) ⊑ ⊥
AnimalDomestique ≡ Animal ⊓ ¬AnimalSauvage

**Ontologie 1 :**

**TBox :**

Personne ≡ ∃nom ⊓ (Femme ⊔ Homme)
Livre ⊑ ProduitCulturel ⊓ ∃auteur.Personne ⊓ ∃titre

**ABox :**

Livre(LIV3009)
auteur(LIV3009, CLS)
titre(LIV3009, "Tristes tropiques")
Homme(CLS)
nom(CLS, "Claude Levis-Strauss")



# Exemple d'ontologie (2)

**Ontologie 2:**

***TBox :***

Célibataire ≡ Personne ⊓ ≤ 0 mariéAvec
Personne ⊑ ∃nom ⊓ ∃age
Homme ⊑ Personne
Femme ⊑ Personne
Père ≡ Personne ⊓ ∃aEnfant.Personne
PèreDeFilles ≡ Père ⊓ ∀aEnfant.Femme

⊤ ⊑ ∀aEnfant.Personne (image)
∃aEnfant.⊤ ⊑ Personne (domaine)
aEnfant ≡ aParent$^-$ (rôle inverse)

⊤ ⊑ ≤ 1 estConjointDe (rôle fonctionnel)
⊤ ⊑ ≤ 1 estConjointDe$^-$ (rôle injectif)
estConjointDe ≡ estConjointDe$^-$ (rôle symétrique)
⊤ ⊑ ∀estConjointDe.Personne (image)
∃estConjointDe.⊤ ⊑ Personne (domaine)
mariéAvec ⊑ estConjointDe



# Exemple d'ontologie (2) suite

⊤ ⊑ ∀age (image)
∃age.⊤ ⊑ Personne (domaine)
⊤ ⊑ ∀nom (image)
∃nom.⊤ ⊑ Personne (domaine)

***ABox :***

Homme(Bernard)
age(Bernard, 56)
Femme(Sabine)
age(Sabine, 46)
Homme(Valentin)
age(Valentin, 10)
mariéAvec(Bernard, Sabine)
aEnfant(Bernard, Valentin)

# 4. Inférences dans les LD

- **Inférences aux niveaux terminologique (TBox) et factuel (ABox)**
- **Comparaison de moteurs d'inférence pour LD**
- **Différents types d'algorithmes d'inférences**
- **La méthode des tableaux sémantiques**



## Inférences, raisonnement dans les LD

**Intérêt du raisonnement dans les LD :**

- **Modélisation et maintenance d'ontologies :**
  - Vérifier la **consistance d'une classe** et **calculer la hiérarchie des classes**
  - Très important pour des ontologies larges /multiples auteurs
- **Intégration d'ontologies :**
  - **Découvrir des relations entre ontologies**
  - Calculer la **consistance** et la **hiérarchie** des **classes intégrées**
- **Faire des requêtes sur des classes et instances :**
  - Déterminer si un **ensemble de faits est consistant** par rapport aux ontologies
  - Déterminer si des **individus** sont des **instances de classes**
  - Retrouver des **individus/tuples satisfaisant** une requête
  - Tester si une **classe subsume** (est plus générale que) **une autre classe**

**L'inférence** s'effectue **au niveau terminologique (TBox)** ou **au niveau factuel (ABox)**



## Inférences dans les LD : au niveau TBox (1)

**4 propriétés intéressantes à prouver pour une TBox** [Baader & Nutt, 2003]:

- **Satisfiabilité** : un concept **C** d'une terminologie $\mathcal{T}$ est **satisfiable** ssi il **existe un modèle $\mathcal{I}$ de $\mathcal{T}$ tel que $C^{\mathcal{I}} \neq \emptyset$**
- **Subsomption** : un concept **C** est **subsumé par** un concept **D** pour une terminologie $\mathcal{T}$ **ssi $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$** pour tout modèle $\mathcal{I}$ de $\mathcal{T}$
- **Équivalence** : un concept **C** est **équivalent** à un concept **D** pour une terminologie $\mathcal{T}$ **ssi $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$,** pour tout modèle $\mathcal{I}$ de $\mathcal{T}$
- **Disjonction** (disjointness) : des concepts **C** et **D** sont **disjoints** par rapport à la terminologie $\mathcal{T}$ **ssi $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$,** pour tout modèle $\mathcal{I}$ de $\mathcal{T}$.



## Inférences dans les LD : niveau TBox (2)

- Résoudre des **problèmes d'inférence** dans la TBox, c'est **prouver une de ces 4 propriétés**
- Résoudre des problèmes d'inférence **se réduit généralement** à prouver soit un **subsomption**, soit une **satisfiabilité** (moteurs d'inférence actuels) :
  - **Réduction des problèmes d'inférence d'une TBox à des problèmes de subsomption :**
    - **C** est **insatisfiable** ⇔ **C** est **subsumé** par $\bot$
    - **C** et **D** sont **équivalents** ⇔ **C** est **subsumé** par **D**, et **D** par **C**
    - **C** et **D** sont **disjoints** ⇔ **C** $\sqcap$ **D** est **subsumé** par $\bot$
  - **Réduction des problèmes d'inférence d'une TBox à des problèmes de satisfiabilité :**
    - **C** est **subsumé** par **D** ⇔ **C** $\sqcap$ $\neg$**D** est **insatisfiable**
    - **C** et **D** sont **équivalents** ⇔ **C** $\sqcap$ $\neg$**D** et $\neg$**C** $\sqcap$ **D** sont **insatisfiables**
    - **C** et **D** sont **disjoints** ⇔ **C** $\sqcap$ **D** est **insatisfiable**

# Inférences dans les LD : niveau TBox (3)

**Complexité du raisonnement au niveau TBox** selon l'expressivité de la LD :

| Complexité | LD |
|---|---|
| P | $\mathcal{AL}$, $\mathcal{ALN}$ |
| NP | $\mathcal{AL\varepsilon}$ |
| PSpace | $\mathcal{ALC}$, $\mathcal{AL\varepsilon N}$ |
| ExpTime | $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, |
| NExpTime | … |

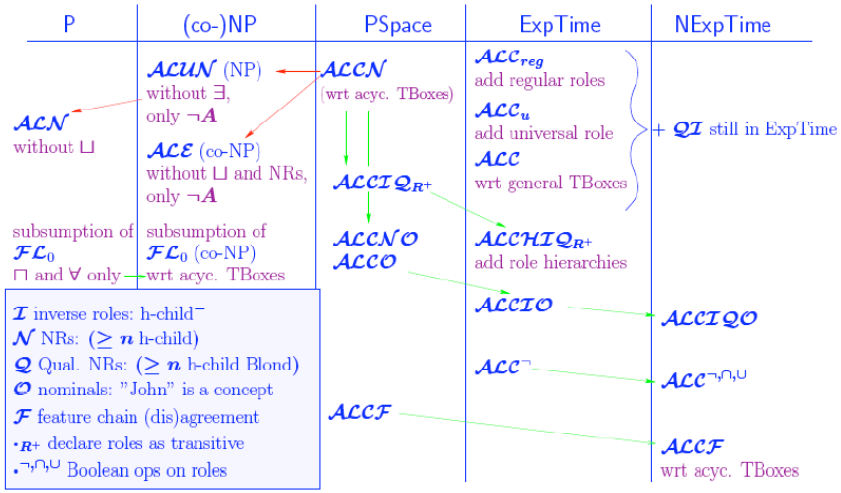
- **P**: classe des problèmes de décision (en entrée un énoncé de problème et en sortie une réponse oui ou non) demandant un **temps polynomial** par rapport à la taille du problème (n), pour obtenir une solution avec une machine de Turing **déterministe**
- **NP**: classe des problèmes nécessitant un **temps polynomial** pour trouver une solution avec une machine de Turing **non déterministe**
- **PSpace** : classe des problèmes demandant une quantité **de mémoire polynomiale** pour une résolution avec une machine de Turing **déterministe** ou **non déterministe**
- **ExpTime** : classe des problèmes solvables par une machine de Turing **déterministe** en un temps $\Theta(2^{p(n)})$ où $p(n)$ est une fonction polynomiale de n
- **NExpTime** : la classe des problèmes solvables par une machine de Turing **non-déterministe** en un temps $\Theta(2^{p(n)})$ où $p(n)$ est une fonction polynomiale de n

**On a P ⊆ NP ⊆ PSpace ⊆ ExpTime ⊆ NExpTime** [Padadimitriou, 1994]



# Inférences dans les LD : niveau TBox (4)

**Some cpmplexity results : http://www.cs.man.ac.uk/~ezolin/dl/**






# Inférences dans les LD : niveau ABox

Le **niveau factuel** (ABox) comprend **4 principaux problèmes d'inférence** [Baader et Nutt, 2003] :

- **Cohérence** : Une ABox $\mathcal{A}$ est **cohérente** par rapport à une TBox $\mathcal{T}$ **ssi il existe un modèle $\mathcal{I}$ de $\mathcal{A}$ et $\mathcal{T}$**
- **Vérification d'instance** : Vérifier par inférence **si une assertion *C(a)* est vraie pour tout modèle $\mathcal{I}$** d'une ABox $\mathcal{A}$ et d'une TBox $\mathcal{T}$
- **Vérification de rôle** : Vérifier par inférence **si une assertion *R(a, b)* est vraie pour tout modèle $\mathcal{I}$** d'une ABox $\mathcal{A}$ et d'une TBox $\mathcal{T}$
- **Problème de récupération** : Pour une ABox $\mathcal{A}$, un concept ***C*** d'une terminologie $\mathcal{T}$, **inférer les individus $a^{\mathcal{I}}_1 \ldots a^{\mathcal{I}}_n \in C^{\mathcal{I}}$** pour tout modèle $\mathcal{I}$ d'une ABox $\mathcal{A}$ et d'une TBox $\mathcal{T}$



# Moteurs d'inférences dans les LDE

Comparaison des principaux moteurs d'inférence pour les LDE (LD Expressives) : **FaCT** [Horrocks, 1998], **Racer** [Haarslev et Möller, 2001], **Pellet** [Sirin et Parsia, 2004], **FaCT++** [Tsarkov et Horrocks, 2004], **F-OWL** [Zou et al., 2004], … :

| Moteur | *Racer* | *FaCT* | *Pellet* | *FaCT++* |
|---|---|---|---|---|
| *LD* | $\mathcal{SHIQ}(\mathcal{D})-$ | $\mathcal{SHIQ}$, $\mathcal{SHF}$ | $\mathcal{SHIN}(\mathcal{D})$, $\mathcal{SHON}(\mathcal{D})$ | $\mathcal{SHIF}(\mathcal{D})$ |
| Implantation | *C++* | *Common Lisp* | *Java* | *C++* |
| Inférence | *TBox/ABox* | *TBox* | *TBox/ABox* | *TBox* |
| API *Java* | oui | oui | natif | oui |
| Mise-à-échelle | bonne | bonne | bonne | bonne |
| *OWL* | *OWL-DL*$\sim^{\dagger}$ | *OWL-DL*$\sim^{\dagger}$ | *OWL-DL*$\sim^{\dagger}$ | *OWL-LITE* |
| Décidabilité | oui (*OWL-LITE*) | oui | oui (*OWL-LITE*) | oui |
| *DIG* | oui | oui | non | ? |
| Moteur | *Surnia* | *Hoolet* | *F-OWL* | |
| *LD* | logique prédicats | logique prédicats | $\mathcal{SHIQ}(\mathcal{D})$ et *RDF* | |
| Implantation | *Python* | *Java* | *Java* | |
| Inférence | *TBox/ABox* | *TBox/ABox* | *TBox/ABox* | |
| API *Java* | ? | oui | oui | |
| Mise-à-échelle | médiocre | médiocre | médiocre | |
| *OWL* | *OWL-FULL*$\sim^{\dagger}$ | *OWL-DL*$\sim^{\dagger}$ | *OWL-FULL*$\sim^{\dagger}$ | |
| Décidabilité | non | non | non | |
| *DIG* | non | non | non | |

# Inférences dans les LD : niveau TBox

- **Calculer les relations de subsomption** entre des noms de concepts dans une TBox est appelé **classification d'une TBox** :

> **La classification d'une TBox $\mathcal{T}$ : c'est déterminer** pour **tous** les noms de concepts **A** et **B** dans $\mathcal{T}$ **si** **A** $\sqsubseteq$ **B** ou **si** **B** $\sqsubseteq$ **A** par rapport à $\mathcal{T}$

- Cela revient à **calculer un ordre** (une hiérarchie) dans les noms de concepts :

  Cet ordre est appelé une **taxonomie**.

- Une **taxonomie T** peut être représentée par un **graphe orienté**.



# Exemple de subsomption par rapport à une TBox (1)

**Exemple** (rappel: le symbole $\equiv$ correspond au symbole $\triangleq$ ) :

**TBox :**

Parent $\triangleq$ Personne $\sqcap$ $\exists$aEnfant.Personne (1)
Pere $\triangleq$ Parent $\sqcap$ Male (2)
GrandParent $\triangleq$ Personne $\sqcap$ $\exists$aEnfant.Parent

**ABox :**

| | |
|---|---|
| Paul : Personne | Paul : Male |
| Pierre : Personne | Alice : Personne |
| (Paul, Pierre) : aEnfant | (Pierre, Alice) : aEnfant |

- **Subsomption dans les concepts :**
  - Personne $\sqcap$ $\exists$aEnfant.Personne $\sqsubseteq$ Personne *par (1) – (3)*
  - Parent $\sqcap$ Male $\sqsubseteq$ Parent *et* Parent $\sqcap$ Male $\sqsubseteq$ Male *par (2) - (4)*
- **Subsomption par rapport à la TBox :**
  - Parent $\sqsubseteq$ Personne *(chaque parent est une personne, par (3))*
  - Pere $\sqsubseteq$ Parent *et* Pere $\sqsubseteq$ Male *par (4)*



# Exemple de subsomption par rapport à une TBox (2)

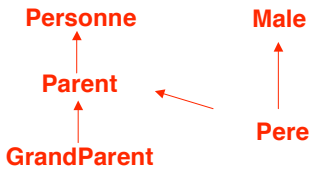
**TBox :**

Parent $\triangleq$ Personne $\sqcap$ $\exists$aEnfant.Personne
Pere $\triangleq$ Parent $\sqcap$ Male
GrandParent $\triangleq$ Personne $\sqcap$ $\exists$aEnfant.Parent

**ABox :**

| | |
|---|---|
| Paul : Personne | Paul : Male |
| Pierre : Personne | Alice : Personne |
| (Paul, Pierre) : aEnfant | (Pierre, Alice) : aEnfant |

**Classification de cette TBox :**






# Consistance de concepts par rapport à une TBox

> **Test de la consistance :** Un **concept C** est **consistant par rapport à une TBox $\mathcal{T}$**, s'il existe une interprétation terminologique $\mathcal{I}$ (satisfaisant $\mathcal{T}$) telle que $\mathcal{I}$(C) n'est pas vide ($\neq \emptyset$)

- La consistance est **très utile pour détecter les incohérences dans les définitions de concepts**

| Concept | Consistance par rapport à la TBox ? |
|---|---|
| C $\sqcap$ $\neg$C | **Non** pour toute Tbox |
| C $\sqcup$ $\neg$C | **Oui** pour toute Tbox |
| $\exists$R.C $\sqcap$ $\forall$R.$\neg$C | **Non** pour toute Tbox |
| $\exists$R.A $\sqcap$ $\forall$R.B | **Non** pour {B $\triangleq$ $\neg$A} |

# Exemple de consistance de concepts par rapport à une TBox

**TBox :**

Parent ≜ Personne ⊓ ∃aEnfant.Personne
Pere ≜ Parent ⊓ Male
GrandParent ≜ Personne ⊓ ∃aEnfant.Parent

**ABox :**

| | |
|---|---|
| Paul : Personne | Paul : Male |
| Pierre : Personne | Alice : Personne |
| (Paul, Pierre) : aEnfant | (Pierre, Alice) : aEnfant |

**Exemple de consistance de concepts :**

- le concept Pere ⊓ ¬Male est **Inconsistant/Incohérent** par rapport à la **TBox**
- tandis que le concept Pere ⊓ Femelle est **consistant/cohérent** par rapport à la **TBox** (il n'y a pas d'indication si les concepts Femelle et Male sont exclusifs)



# Types d'algorithmes d'inférence en LD (1)

**2 grandes classes d'algorithmes de raisonnement pour les LD** :

- **1. Algorithmes de vérification de subsomption de type normalisation - comparaison (NC)**
  - réduisent les problèmes d'inférence à des problèmes de **subsomption,** ceci en **temps polynomial**
  - un processus de normalisation produit les **formes normales de concepts** définis qui sont ensuite effectivement **comparées** à l'aide de **règles de comparaisons structurelles**
  - **Idée de l'algorithme**: si 2 expressions de concepts sont composées de sous-expressions, comparer séparément une sous-expression d'un concept avec toutes les sous-expressions des autres concepts.
  - **Ces algorithmes :**
    - **ont une correction facile à vérifier, mais une complétude difficile à démontrer**
    - ne s'appliquent qu'à des **LD peu expressives**, sans quoi ils sont incomplets, cad qu'ils sont incapables de prouver certaines formules vraies.



# Types d'algorithmes d'inférence en LD (1)

- **2. Algorithmes de vérification de satisfiabilité à base de tableaux**
  - **années 1990** : nouveaux **algorithmes de vérification de satisfiabilité à base de tableaux** (tableau-based algorithms).
  - Ces algorithmes réduisent ainsi les problèmes d'inférence dans les LD à des problèmes de **satisfiabilité**
  - **Idée de l'algorithme** de la **méthode des tableaux sémantiques** : dans la LD considérée devant disposer de la négation (¬), la question *le concept D subsume-t-il le concept C ?* est transformée en *l'expression C ⊓ ¬D est-elle non satisfiable ?*
  - **Ces algorithmes :**
    - raisonnent sur des **LD dites expressives ou très expressives**,
    - **en temps exponentiel**, mais en pratique, le comportement des algorithmes est souvent **acceptable**.
    - **ils ont une complexité et décidabilité plus facile**
    - La forte expressivité des LD traitées a ouvert la porte à de nouvelles applications telles que le **Web sémantique.**



# Tableaux sémantiques : définition

- **Rappel** : réduction des problèmes d'inférence d'une TBox à des problèmes de **satisfiabilité** :
  - **C** est **subsumé** par **D** ⇔ **C ⊓ ¬D** est **insatisfiable**
  - **C** et **D** sont **équivalents** ⇔ **C ⊓ ¬D** et **¬C ⊓ D** sont **insatisfiables**
  - **C** et **D** sont **disjoints** ⇔ **C ⊓ D** est **insatisfiable**
- Une **procédure de tableau sémantique peut être considérée** comme une **procédure pour construire une interprétation satisfaisante de l'assertion d'un concept donné.**
- Les **dérivations** peuvent être établies sous la forme d'un **arbre**, dont les arêtes représentent la succession des rôles entre les éléments du domaine d'interprétation Δ
- Pour des raisons de commodité, on s'intéresse particulièrement aux procédures qui travaillent avec des concepts sous **forme normale négative** (Negation Normal Form - **NNF**).

# Tableaux sémantiques : forme normale négative – NNF (1)

**Forme normale négative (Negation Normal Form – NNF) :**

- Soit C un concept arbitraire dans $\mathcal{ALC}$
- On dit que C est en **forme normale négative (NNF) ssi ¬ apparaît seulement immédiatement devant le nom d'un concept**

**La NNF peut être obtenue en appliquant les règles suivantes :**

| | |
|---|---|
| $\neg\neg C \Rightarrow_{NNF} C$ | |
| $\neg\top \Rightarrow_{NNF} \bot$ | $\neg\bot \Rightarrow_{NNF} \top$ |
| $\neg(C \sqcup D) \Rightarrow_{NNF} \neg C \sqcap \neg D$ | $\neg(C \sqcap D) \Rightarrow_{NNF} \neg C \sqcup \neg D$ |
| $\neg\forall R.C \Rightarrow_{NNF} \exists R.\neg C$ | $\neg\exists R.C \Rightarrow_{NNF} \forall R.\neg C$ |



# Tableaux sémantiques : forme normale négative – NNF (2)

**Exemple :**

Soit le concept ¬(Etudiant ⊓ Heureux)

alors :

¬( Etudiant ⊓ Heureux) $\Rightarrow_{NNF}$ ¬( Etudiant ⊔ ¬ Heureux)

considérons le concept ¬(Mere ⊔ (¬Femelle ⊓ ∃aEnfant.Personne))

alors :

¬( Mere ⊔ (¬femelle ⊓ ∃aEnfant.Personne))

$\Rightarrow_{NNF}$ ¬Mere ⊓ ¬(¬femelle ⊓ ∃aEnfant.Personne)

$\Rightarrow_{NNF}$ ¬Mere ⊓ ((¬¬femelle ⊔ (¬∃aEnfant.Personne))

$\Rightarrow_{NNF}$ ¬Mere ⊓ (femelle ⊔ (¬∃aEnfant.Personne))

$\Rightarrow_{NNF}$ ¬Mere ⊓ (femelle ⊓ ∀aEnfant.¬Personne)



# Tableaux sémantiques : principe (1)

- Un **calcul de tableau sémantique** est une **procédure de preuve formelle**, existant dans plusieurs DL, avec les mêmes caractéristiques :
  - Un tableau sémantique est un **système de réfutation** : étant donné un système initial de contraintes ou tableau T, il tente soit de montrer que **T est non satisfiable**, soit de **construire une interprétation satisfaisante**
  - Un tableau est une suite de séries d'affirmations construites selon certaines **règles d'inférence,** et est généralement énoncé sous la forme d'un **arbre :**
    - Les règles d'inférence stipulent comment un tableau T génère un nouveau tableau dans lequel une branche est transformée en n nouvelles branches.
    - Cela se fait en élargissant la branche à sa feuille, en créant jusqu'à n nœuds successeurs
    - Chacun des nouveaux nœuds contient de nouvelles affirmations



# Tableaux sémantiques : forme générale des règles d'inférence

En général, les **règles d'inférence**, appelées **règles d'expansion** sont de la forme :

$$\frac{X}{X_1 \mid \ldots \mid X_n}$$

Où X et Xi dénotent une ou plusieurs assertions

- Si le **tableau T contient des assertions de la forme X**, alors le tableau est **étendu** en développant la branche qui contient X à sa feuille en créant n nœuds successeurs :
  - Le successeur du nœud 1 contient X1, …, le successeur du nœud n contient Xn.
  - Les assertions dans X sont des **prémisses**, les assertions dans Xi sont des **conclusions**

# Tableaux sémantiques dans $\mathcal{ALC}$ (1)

**5 Règles d'expansion dans $\mathcal{ALC}$ :**

$$(\sqcap)\ \frac{a : C \sqcap D}{a : C,\ a : D} \qquad (\sqcup)\ \frac{a : C \sqcup D}{a : C \mid a : D}$$

$$(\bot)\ \frac{a : \neg A,\ a : A}{a : \bot}$$

$$(\exists)\ \frac{a : \exists R.C}{(a, b) : R,\ b : C}\ \text{where } b \text{ is a new object}$$

$$(\forall)\ \frac{a : \forall R.C,\ (a, b) : R}{b : C}$$

**Condition** : chaque règle est appliquée **une seule fois à la même instance du numerateur.**



# Tableaux sémantiques dans $\mathcal{ALC}$ (2)

**1. Règle ($\sqcap$)** *(règle « et ») :*

$$(\sqcap)\ \frac{a : C \sqcap D}{a : C,\ a : D}$$

- si un tableau donné T contient une **assertion a : C ⊓ D**, alors il peut être **étendu** pour former un **nouveau tableau par addition à la fois de a : C et a : D à la branche contenant a : C ⊓ D**
- si a : C ⊓ D ∈ T et n'a pas été encore étendu, alors on dit que la règle (⊓) est applicable à la branche dans laquelle a : C ⊓ D apparaît, ou la règle (⊓) est applicable au tableau T
- de même pour les autres règles



# Tableaux sémantiques dans $\mathcal{ALC}$ (3)

**2. Règle ($\sqcup$)** *(règle « ou») :*

$$(\sqcup)\ \frac{a : C \sqcup D}{a : C \mid a : D}$$

- si un tableau donné T contient une **assertion a : C ⊔ D**, alors il peut être **étendu** pour former un **nouveau tableau avec 2 nouvelles branches** : la branche contenant a : C ⊔ D est **étendue avec 2 noeuds successeurs contenant a : C et a : D, respectivement**
- la règle (⊔) est une « **branching rule** », car elle crée 2 nouvelles branches
- **intuitivement**, dans cette règle, si a : C ⊔ D est vrai dans une interprétation, alors soit a : C ou a : D doit être vrai



# Tableaux sémantiques dans $\mathcal{ALC}$ (3)

**3. Règle ($\bot$)** *(règle « clash» - clash rule ou closure rule) :*

$$(\bot)\ \frac{a : \neg A,\ a : A}{a : \bot}$$

- si une branche d'un tableau T contient une **assertion a : A et a : ¬A**, où A denote un nom de concept, alors il peut être **étendu** pour former un **nouveau tableau par ajout de a : ⊥ à la branche**.
- a : ⊥ n'est pas satisfiable pour aucune interprétation : une fois que a :⊥ est dérivé, alors on a une inconsistance ou un clash.
- une **branche** est dite « **fermée** » (closed) si elle contient une assertion de la forme a : ⊥. Sinon elle est dite « **ouverte** » (open).
- un **tableau** est dit « **fermé** » (closed) si toutes ses branches sont fermées. Sinon le tableau est « **ouvert** » (open).

# Tableaux sémantiques dans $\mathcal{ALC}$ (3)

**4. Règle (∃) *(règle « some »)* :**

$$(\exists)\ \frac{a : \exists R.C}{(a,b) : R,\ b : C}\ \text{where } b \text{ is a new object}$$

- si un tableau T contient une **assertion a : ∃R.C** alors il peut être étendu et former un **nouveau tableau** en choisissant un nouvel objet b n'apparaissant pas dans la branche et en ajoutant à la fois (a, b) : R et b : C à la branche contenant a : ∃R.C.
- la règle traduit l'intuition que si a : ∃R.C alors a possède un « R-relative » (rôle relatif), nommé b, dans C.

**5. Règle (∀) *(règle « all »)* :**

$$(\forall)\ \frac{a : \forall R.C,\ (a,b) : R}{b : C}$$

- si un tableau T contient **2 assertions a : ∀R.C et (a, b) : R** alors il peut être étendu et former un **nouveau tableau par ajout de b : C** à la branche.
- règle traduisant l'intuition que si a : ∀R.C, alors a est le « R-relative » (rôle relatif) de rien que des b dans C, cad tout b tel que (a, b) : R appartient à C.



# Tableaux sémantiques dans $\mathcal{ALC}$ (3)

**Algorithme de tableau pour tester la satisfiabilité d'un concept :**

- Pour résoudre le problème: **le concept C est-il satisfiable/consistant ?**
  - 1. Le tableau initial est **{x : NNF (C)}** i.e. un ensemble fini d'assertions en forme normale négative
  - 2. Pour chaque branche, appliquer les règles d'expansion
  - 3. Arrêter d'étendre une branche si :
    - Elle contient une **assertion de la forme a : ⊥**, ce qui veut dire que la branche est **fermée**
    - Il n'existe plus aucune règle d'expansion applicable. Dans ce cas la branche est **complète**.
  - 4. Arrêter si une branche complète est trouvée dans le tableau T ou si toutes les branches sont fermées. Sinon, recommencer à 2.



# Tableaux sémantiques dans $\mathcal{ALC}$ (3)

**Validité et complétude du tableau :** Un ensemble d'assertions en FNN est **non satisfiable** ssi l'algorithme du tableau peut être utilisé pour construire un **tableau fermé**

- Plus précisément:
  - **L'entrée est non satisfiable /inconsistante** ssi **toutes les branches** de n'importe quel arbre de dérivation construit à partir de l'entrée sont **fermées**
  - **L'entrée est satisfiable/consistante** ssi il **existe une branche ouverte complète** dans un arbre de dérivation construit à partir de l'entrée



# Tableaux sémantiques dans $\mathcal{ALC}$ : exemple (1)

**Le concept suivant est-il consistant/satisfiable ?** *(Ex. d'après Schmidt)*

(Etudiant ⊓ Heureux) ⊓ (¬Etudiant ⊔ ¬Heureux)

déjà en NNF, d'où le **tableau initial** :

x : (Etudiant ⊓ Heureux) ⊓ (¬Etudiant ⊔ ¬Heureux)

La règle (⊓/et) est applicable à (1) et étend le tableau ainsi :

| | | |
|---|---|---|
| 1. | **x : (Etudiant ⊓ Heureux) ⊓ (¬Etudiant ⊔ ¬Heureux)** | donné |
| 2. | x : Etudiant ⊓ Heureux | par 1, (⊓) |
| 3. | x : ¬Etudiant ⊔ ¬Heureux | par 1, (⊓) |

On doit choisir, soit étendre 2 ou 3. On choisi d'appliquer la règle (⊓/et) à 2, ce qui donne :

| | | |
|---|---|---|
| 1. | x : (Etudiant ⊓ Heureux) ⊓ (¬Etudiant ⊔ ¬Heureux) | donné |
| 2. | **x : Etudiant ⊓ Heureux** | par 1, (⊓) |
| 3. | x : ¬Etudiant ⊔ ¬Heureux | par 1, (⊓) |
| 4. | x : Etudiant | par 2, (⊓) |
| 5. | x : Heureux | par 2, (⊓) |

# Tableaux sémantiques dans $\mathcal{ALC}$ : exemple (2)

On applique la règle (⊔/ou) à 3, et on **obtient 2 branches (6 et 7)** :

1. x : (Etudiant ⊓ Heureux) ⊓ (¬Etudiant ⊔ ¬Heureux) — donné
2. x : Etudiant ⊓ Heureux — par 1, (⊓)
3. **x : ¬Etudiant ⊔ ¬Heureux** — par 1, (⊓)
4. x : Etudiant — par 2, (⊓)
5. x : Heureux — par 2, (⊓)
6. x : ¬Etudiant | 7. x : ¬Heureux — par 3, (⊔)

On considère d'abord la branche de gauche. La règle (⊥/clash) est applicable à 4 et 6 :

1. x : (Etudiant ⊓ Heureux) ⊓ (¬Etudiant ⊔ ¬Heureux) — donné
2. x : Etudiant ⊓ Heureux — par 1, (⊓)
3. x : ¬Etudiant ⊔ ¬Heureux — par 1, (⊓)
4. **x : Etudiant** — par 2, (⊓)
5. x : Heureux — par 2, (⊓)
6. **x : ¬Etudiant** | 7. x : ¬Heureux — par 3, (⊔)
8. x : ⊥ par 4, 6, clash

Closed



# Tableaux sémantiques dans $\mathcal{ALC}$ : exemple (3)

Du fait que la branche de gauche est fermée, on continue la dérivation sur la branche de droite. La règle (⊥/clash) est applicable à 5 et 7 :

1. x : (Etudiant ⊓ Heureux) ⊓ (¬Etudiant ⊔ ¬Heureux) — donné
2. x : Etudiant ⊓ Heureux — par 1, (⊓)
3. x : ¬Etudiant ⊔ ¬Heureux — par 1, (⊓)
4. x : Etudiant — par 2, (⊓)
5. **x : Heureux** — par 2, (⊓)
6. x : ¬Etudiant | 7. **x : ¬Heureux** — par 3, (⊔)
8. x : ⊥ par 4, 6, clash | 9. x : ⊥ par 5, 7, clash

**Closed** | **Closed**



# Tableaux sémantiques dans $\mathcal{ALC}$ : exemple (4)

**Toutes les branches** (il y en a 2) du tableau sont **fermées** (et aucune règle n'est applicable) :

1. x : (Etudiant ⊓ Heureux) ⊓ (¬Etudiant ⊔ ¬Heureux) — donné
2. x : Etudiant ⊓ Heureux — par 1, (⊓)
3. x : ¬Etudiant ⊔ ¬Heureux — par 1, (⊓)
4. x : Etudiant — par 2, (⊓)
5. x : Heureux — par 2, (⊓)
6. x : ¬Etudiant | 7. x : ¬Heureux — par 3, (⊔)
8. x : ⊥ par 4, 6, clash | 9. x : ⊥ par 5, 7, clash

**Closed** | **Closed**

**Puisque toutes les branches sont fermées, on en déduit que le concept en entrée n'est pas satisfiable**



# Tableaux sémantiques dans $\mathcal{ALC}$ : exemple (5)

**Soit l'arbre de dérivation :**