Chapitre 4 :
Les logiques de description

Les logiques de description LD sont des formalismes de représentation des connaissances descendant du système **KL-ONE** (Brachman 2003) qui lui-même est issu à la fois des **réseaux sémantiques** (Ross Quillian 1968) et des **frames** (M. .Minsky 1975).

Un réseau KL-ONE est un ensemble d'éléments en relation les uns avec les autres. Les éléments du réseau sont des ***concepts*** et les relations du réseau sont appelées des ***rôles***.

Les concepts peuvent être *primitifs* (ils possèdent alors des propriétés nécessaires mais pas suffisantes) ou *définis* (ils possèdent alors des propriétés nécessaires et suffisantes).

Si C est un concept défini, le système KL-ONE peut répondre à des questions du type : '' *le concept C* ***subsume****-t-il le concept D*? ou *le concept C est-il subsumé par le concept D*?

KL-ONE est doté d'un ***classifieur*** qui place les nouveaux concepts définis dans la hiérarchie existante (taxonomie).

Phases of Description Logic Research

- Phase 0 (1965-1980): Pre-DL phase
  - Semantic networks, frames, structured inheritance networks
- Phase 1 (1980-1990): Structural subsumption algorithms
  - Implementation of systems
    - KL-ONE, K-Rep, Krypton, Back, LOOM
- Phase 2 (1990-1995) Tableau-based algorithms
  - Focus on propositionally closed DLs
  - Thorough analysis of complexity of reasoning
- Phase 3 (1995-2000) Very expressive DLs
  - Improving Tableau-based methods or conversion to modal logic
- Phase 4 (2000-onwards)
  - Industrial strength system for very expressive DLs with applications to semantic web, bio-medical informatics

Modern Description Logics

- Well-defined formal semantics
  - Fragments of First Order Logic
  - Closely related to propositional modal logic
- Computational properties are well understood
- Reasoning services
  - Practical decision procedures for key problems:
    - satisfiability, subsumption
  - Several implemented reasoning systems are available

Modern Notation

- A man that is married to a doctor, and all of whose children are either doctors or professors.
- Un homme marié à une docteure, don't tous les enfants sont docteurs ou professeurs.

– **Brachman & Levesque notation :**

[AND Man
[EXISTS :married Doctor]
[ALL : hasChild [OR Doctor Professor]]

- **Current Notation:**

Human ⊓ ¬ Female ⊓ (∃married.Doctor)
⊓ (∀hasChild.(Doctor ⊔ Professor))

De la logique modale minimale K à la logique multi-modale minimale $K_{(m)}$

- Les axiomes (A1), (A2) et (A3) de la logique propositionnelle
- Plus l'axiome (A6) de distributivité de la nécessité sur l'implication matérielle:

  (A6) : $(\Box(a \supset b) \supset (\Box a \supset \Box b))$

- Les règles d'inférence (R1) et (R2) de substitution et de *modus ponens*
- *Plus la règle d'inférence de nécessitation* (R6) :

  si x est une formule,

  R6(x) est l'ensemble contenant l'unique élément $\Box x$

- *La logique modale minimale K considère un seul agent, et donc un seul opérateur modal de nécessité, et les modèles modaux correspondants, de la forme <W,R,v>, ont une seule relation d'accessibilité sans propriétés particulières*
- *La logique multi-modale $K_{(m)}$ est une généralisation à m agents, et donc m opérateurs modaux de nécesssité, et les modèles modaux correspondants sont les structures de Kripke de la forme <W,R1,…,Rm,v>, avec m relations d'accessibilité sans propriétés particulières, une par agent.*

Système de déduction $K_{(m)}$

- La logique multi-modale propositionnelle $K_{(m)}$ comporte les axiomes (A1), (A2) et (A3) de la logique propositionnelle ; plus l'axiome (A6) :

  (A6) : $(\Box_i(a \supset b) \supset (\Box_i a \supset \Box_i b))$, pour tout i de 1 à m

(distributivité de la nécessité sur l'implication matérielle)

- Les règles d'inférence sont les règles de substitution et de *modus ponens* (R1) et (R2) auxquelles s'ajoute la règle (R6) :

  (R6) [nécessitation] : si x est une formule, R6(x,i) est l'ensemble contenant l'unique élément $\Box_i x$

- Halpern et Moses [IJCAI 1985] :

Le système multi-modal $K_{(m)}$ est correct et complet :

f théorème ssi f tautologie

De la logique multi-modale minimale $K_{(m)}$ à la logique de description ALC

Schild (IJCAI 1991) : la logique de description ALC est une variante notationnelle de la logique multi-modale $K_{(m)}$

The Description Logic ALC

- **A**ttributive Concept **L**anguage with **C**omplements

  N_C – set of concept names (e.g., Parent, Sister, Student)

  N_R – set of role names (e.g., EmployedBy, MotherOf)

  N_I – set of individual names (e.g., a_1, a_2, …)

- The set of ALC concepts is the smallest set such that:

– The following are concepts:

  $\top$ (top is a concept), $\bot$ (bottom is a concept), every $A \in N_C$ (all atomic concepts are concepts)

– If C and D are concepts and $R \in N_R$ then the following are concepts:

  - $C \sqcap D$ (the intersection of two concepts is a concept)
  - $C \sqcup D$ (the union of two concepts is a concept)
  - $\neg C$ (the complement of a concept is a concept)
  - $\forall R.C$ (the universal restriction of a concept by a role is a concept)
  - $\exists R.C$ (the existential restriction of a concept by a role is a concept)

The Description Logic ALC

- Un concept permet de représenter un ensemble d'individus (relation unaire).
- Un rôle représente une relation binaire entre individus.

The Description Logic ALC

La modélisation des connaissances d'un domaine avec les LD se réalise en deux niveaux. Le premier, le niveau terminologique ou TBox, décrit les connaissances générales d'un domaine alors que le second, le niveau factuel ou ABox, représente une configuration précise.

Une TBox (Terminological Box) comprend la définition des concepts et des rôles, alors qu'une ABox (Assertional Box) décrit les individus en les nommant et en spécifiant, en terme de concepts et de rôles, des assertions qui portent sur ces individus nommés.

Plusieurs ABox peuvent être associés à une même TBox ; chacune représente une configuration constituée d'individus, et utilise les concepts et rôles de la TBox pour l'exprimer.

The Description Logic ALC

- Terminological Axioms (**TBox**)
  - A general concept inclusion axiom has the form $C \sqsubseteq D$ where C and D are concepts
  - Write $C \equiv D$ iff both $C \sqsubseteq D$ and $D \sqsubseteq C$
  - A TBox is a finite set of GCIs
- Assertional Axioms (**ABox**)
  - A concept assertion is a statement of the form $a:C$ where $a \in N_I$ and C is a concept
  - A role assertion is a statement of the form $(a,b):R$ where $a, b \in N_I$ and R is a role
  - An ABox is a finite set of assertional axioms
- Knowledge Base (**KB**)
  - A KB is an ordered pair (T,A) for a TBox T and ABox A
  - A KB is also called ontology

Knowledge Base
-Example-

- **TBox T :**

  Course $\sqsubseteq$ $\neg$Person

  Teacher $\sqsubseteq$ Person $\sqcap$ $\exists$Teaches.Course

  $\exists$Teaches.T $\sqsubseteq$ Person

  Student $\sqsubseteq$ Person $\sqcap$ $\exists$Attends.Course

  $\exists$Attends.T $\sqsubseteq$ Person

- **ABox A :**

  Mary : Person

  CS600 : Course

  Alice : Person $\sqcap$ Teacher

  (Alice; CS600) : Teaches

  (Mary; CS600) : Attends

Interpretation

An *interpretation* $I = (\Delta^I, .^I)$ *consists of a non-empty set* Δ^I, the *domain of I, and a function* $.^I$ *which maps* every concept name to a subset of Δ^I, *every role name* to a subset of $\Delta^I \times \Delta^I$, and every individual name to an element of Δ^I, such that:

$(C \sqcap D)^I = C^I \cap D^I$

$(C \sqcup D)^I = C^I \cup D^I$

$\neg C^I = \Delta^I \setminus C^I$

$(\exists R.C)^I = \{x \in \Delta^I \mid \exists y \in \Delta^I \text{ such that } (x,y) \in R^I \text{ and } y \in C^I\}$

$(\forall R.C)^I = \{x \in \Delta^I \mid \text{for all } y \in \Delta^I, \text{ if } (x,y) \in R^I \text{ then } y \in C^I\}$

Satisfiability and Subsumption

A concept C is *satisfiable if there is an interpretation I with* $C^I \neq \emptyset$. Such an interpretation is called a *model of C.*

A concept D *subsumes a concept C (written* $C \sqsubseteq D$) if $C^I \subseteq D^I$ holds in every interpretation I. Two concepts C and D are *equivalent (written* $C \equiv D$) if they subsume each other.

Subsumption and satisfiability can be reduced to each other:

$C \sqsubseteq D$ iff $C \sqcap \neg D$ is not satisfiable

C satisfiable iff $C \not\sqsubseteq \bot$

Satisfiability and Subsumption with respect to a KB

An interpretation I *satisfies a GCI axiom* $C \sqsubseteq D$, written $I \models C \sqsubseteq D$, *if* $C^I \subseteq D^I$; and I satisfies a TBox T, written $I \models T$, if I satisfies each GCI axiom of T ; such an interpretation is called a *model of T.*

An interpretation I *satisfies a concept assertion a:C, written* $I \models a:C$, *if* $a^I \in C^I$; *I satisfies a role assertion (a,b):R , written* $I \models (a,b):R$, *if* $(a^I,b^I) \in R^I$; and I satisfies an ABox *A, written* $I \models A$, if I satisfies each assertional axiom of A ; such an interpretation is called a *model of A.*

An interpretation I is a model of a KB K=(T,A) *, written* $I \models K$, if it is a model of T and a model of A: $I \models K$ iff $I \models T$ *and* $I \models A$.

A concept C is *satisfiable w.r.t. a KB K if there is a model I of K with* $C^I \neq \emptyset$. Such an interpretation is called a *model of C w.r.t. K. A KB K entails a concept C, written* $K \models C$, if $C^I \neq \emptyset$, *for every model I of K.*

A concept D *subsumes a concept C w.r.t. a KB K (written* $C \sqsubseteq_K D$, or $K \models C \sqsubseteq D$, or K entails $C \sqsubseteq D$) if $C^I \subseteq D^I$ holds in every model I of K. Two concepts C and D are *equivalent w.r.t. K (written* $C \equiv_K D$, or $K \models C \equiv D$) if they subsume each other w.r.t. K.

Satisfiability and Subsumption with respect to a KB

Satisfiability and subsumption w.r.t. to a KB can be reduced to each other:

$$K \models C \sqsubseteq D \text{ iff } K \not\models C \sqcap \neg D$$

$$K \models C \text{ iff } K \not\models C \sqsubseteq \bot$$

Naming Conventions

S : basic DL (ALC) plus transitive roles (e.g., ancestor $\in$ R+)

N : number restrictions (e.g., ≥2hasChild, ≤3hasChild)

Q : Qualified number restrictions (e.g., ≥2hasChild.Doctor)

D : concrete domains (e.g., real, integer, string)

O : Nominals, i.e., individual names:

e.g., Scientists $\sqcap$ ($\exists$hasMet.{Turing})

I : inverse roles (e.g., isChildOf $\equiv$ hasChild^{-}**)**

H : role hierarchy (e.g., hasDaughter $\subseteq$ hasChild)

SHOIN(D) : A ALC description logic with transitive roles, role hierarchies, nominals, inverse roles, number restrictions, and a concrete domain ==> the logic of the language OWL-DL

Tableau Algorithms

- **Model construction**
  - ✓ Prove entailment does not hold by constructing model of KB in which axiom/fact is false
  - ✓ tableau expansion rules used to derive **new ABox facts**
- **Currently the most widely used technique**
  - ✓ Basis for reasoners such as FaCT++, HermiT, Pellet, Racer, …
  - ✓ Standard technique is to negate premise axiom/fact

Tableau Algorithms

- Transform entailment to KB (un)satisfiability
  - $\mathcal{K} \models$ a:C iff $\mathcal{K} \cup$ {a:(¬C)} is *not* satisfiable
  - $\mathcal{K} \models C \sqsubseteq D$ iff $\mathcal{K} \cup$ {a:(C ⊓ ¬D)} is *not* satisfiable (for new a)
- Start with facts explicitly asserted in ABox

  e.g., John:HappyParent, John hasChild Mary
- Use expansion rules to derive new **ABox facts**

  e.g., John:Parent, John:∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)
- Construction fails if obvious contradiction (clash)

  e.g., Mary:Doctor, Mary:¬Doctor

Expansion Rules for ALC

$\sqcap$-rule: if 1. $a : (C_1 \sqcap C_2) \in \mathcal{A}$, and
2. $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$
then set $\mathcal{A}_1 = \mathcal{A} \cup \{a : C_1, a : C_2\}$

$\sqcup$-rule: if 1. $a : (C_1 \sqcup C_2) \in \mathcal{A}$, and
2. $\{a : C_1, a : C_2\} \cap \mathcal{A} = \emptyset$
then set $\mathcal{A}_1 = \mathcal{A} \cup \{a : C_1\}$ and $\mathcal{A}_2 = \mathcal{A} \cup \{a : C_2\}$

$\exists$-rule: if 1. $a : (\exists S.C) \in \mathcal{A}$, and
2. there is no b such that $\{\langle a, b\rangle : S, b : C\} \subseteq \mathcal{A}$,
then set $\mathcal{A}_1 = \mathcal{A} \cup \{\langle a, d\rangle : S, d : C\}$, where d is new in $\mathcal{A}$

$\forall$-rule: if 1. $\{a : (\forall S.C), \langle a, b\rangle : S\} \subseteq \mathcal{A}$, and
2. $b : C \notin \mathcal{A}$
then set $\mathcal{A}_1 = \mathcal{A} \cup \{b : C\}$

- some rules are nondeterministic, e.g., $\sqcup$, $\leq$
- implementations use backtracking search

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

Start with the assertions in the Abox

1. **John: HappyParent** **<concept assertion>**
2. **(John,Mary): hasChild** **<role assertion>**

Showing that K$\models$Mary:Doctor (KB K entails Mary:Doctor, ou Mary:Doctor est une conséquence logique de la base de connaissances K) is equivalent to showing that K$\cup${Mary:$\neg$Doctor} is not satisfiable.

Méthode des tableaux sémantiques :

- Si toutes les branches de l'arbre sont fermées (clash) alors K$\cup${Mary:$\neg$Doctor} est non satisfiable, et donc K$\models$Mary:Doctor (Mary:Doctor est une conséquence logique de K)
- Si la méthode des tableaux arrive à construire une branche ouverte complète, cette dernière correspondra à un modèle de K satsifaisant Mary:$\neg$Doctor, et on aura montré K$\not\models$Mary:Doctor (Mary:Doctor n'est pas une conséquence logique de K).

So the tableaux method proceeds by adding the assertion Mary:$\neg$Doctor to the ABox:

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. **John: HappyParent** **<concept assertion>**
2. **(John,Mary): hasChild** **<role assertion>**
3. **Mary: ¬Doctor**

In 1. replace defined concept HappyParent with the right hand side of the definition:

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. **John: HappyParent** **<concept assertion>**
2. **(John,Mary): hasChild** **<role assertion>**
3. **Mary: ¬Doctor**
4. **John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)**

Apply the $\sqcap$–rule to 4. :

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. **John: HappyParent** **<concept assertion>**
2. **(John,Mary): hasChild** **<role assertion>**
3. **Mary: ¬Doctor**
4. **John: Parent ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)**
5. **John: Parent**
6. **John: ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)**

In 5. replace defined concept Parent with the right hand side of the definition:

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. **John: HappyParent** **<concept assertion>**
2. **(John,Mary): hasChild** **<role assertion>**
3. **Mary: ¬Doctor**
4. **John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)**
5. **John: Parent**
6. **John: $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)**
7. **John: Person $\sqcap$ $\exists$hasChild.Person**

Apply the $\forall$–rule to 2. and 6. :

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. **John: HappyParent** **<concept assertion>**
2. **(John,Mary): hasChild** **<role assertion>**
3. **Mary: ¬Doctor**
4. **John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)**
5. **John: Parent**
6. **John: $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)**
7. **John: Person $\sqcap$ $\exists$hasChild.Person**
8. **Mary: Doctor $\sqcup$ $\exists$hasChild.Doctor**

Apply the $\sqcap$–rule to 7. :

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. John: HappyParent <concept assertion>
2. (John,Mary): hasChild <role assertion>
3. Mary: $\neg$Doctor
4. John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
5. John: Parent
6. John: $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
7. John: Person $\sqcap$ $\exists$hasChild.Person
8. Mary: Doctor $\sqcup$ $\exists$hasChild.Doctor
9. John:Person
10. John: $\exists$hasChild.Person

Apply the $\sqcup$–rule to 8. : choice (branching)

- first Mary: Doctor
- then Mary: $\exists$hasChild.Doctor

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. John: HappyParent <concept assertion>
2. (John,Mary): hasChild <role assertion>
3. Mary: ¬Doctor
4. John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
5. John: Parent
6. John: $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
7. John: Person $\sqcap$ $\exists$hasChild.Person
8. Mary: Doctor $\sqcup$ $\exists$hasChild.Doctor
9. John:Person
10. John: $\exists$hasChild.Person
11. Mary:Doctor

Apply the clash-rule to 3. and 11. :

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. **John: HappyParent** **<concept assertion>**
2. **(John,Mary): hasChild** **<role assertion>**
3. **Mary: ¬Doctor**
4. **John: Parent ⊓ ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)**
5. **John: Parent**
6. **John: ∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)**
7. **John: Person ⊓ ∃hasChild.Person**
8. **Mary: Doctor ⊔ ∃hasChild.Doctor**
9. **John:Person**
10. **John: ∃hasChild.Person**
11. **Mary:Doctor**
12. ⊥

The branch is now closed (clash). We continue our exploration by backtracking to the second choice we left when we applied the ⊔–rule to 8. :

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. John: HappyParent <concept assertion>
2. (John,Mary): hasChild <role assertion>
3. Mary: ¬Doctor
4. John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
5. John: Parent
6. John: $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
7. John: Person $\sqcap$ $\exists$hasChild.Person
8. Mary: Doctor $\sqcup$ $\exists$hasChild.Doctor
9. John:Person
10. John: $\exists$hasChild.Person
11. Mary:Doctor 13.Mary: $\exists$hasChild.Doctor
12. $\bot$

Apply the $\exists$ -rule to 10. :

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. John: HappyParent <concept assertion>
2. (John,Mary): hasChild <role assertion>
3. Mary: ¬Doctor
4. John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
5. John: Parent
6. John: $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
7. John: Person $\sqcap$ $\exists$hasChild.Person
8. Mary: Doctor $\sqcup$ $\exists$hasChild.Doctor
9. John:Person
10. John: $\exists$hasChild.Person
11. Mary:Doctor
12. $\bot$
13. Mary: $\exists$hasChild.Doctor
14. (John,a):hasChild
15. a:Person

Apply the $\exists$ -rule to 13. :

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. John: HappyParent <concept assertion>
2. (John,Mary): hasChild <role assertion>
3. Mary: ¬Doctor
4. John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
5. John: Parent
6. John: $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
7. John: Person $\sqcap$ $\exists$hasChild.Person
8. Mary: Doctor $\sqcup$ $\exists$hasChild.Doctor
9. John:Person
10. John: $\exists$hasChild.Person
11. Mary:Doctor
12. $\bot$
13. Mary: $\exists$hasChild.Doctor
14. (John,a):hasChild
15. a:Person
16. (Mary,b):hasChild
17. b:Doctor

Apply the first GCI axiom of the TBox to 17. :

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. John: HappyParent <concept assertion>
2. (John,Mary): hasChild <role assertion>
3. Mary: ¬Doctor
4. John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
5. John: Parent
6. John: $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
7. John: Person $\sqcap$ $\exists$hasChild.Person
8. Mary: Doctor $\sqcup$ $\exists$hasChild.Doctor
9. John:Person
10. John: $\exists$hasChild.Person
11. Mary:Doctor
12. $\bot$
13. Mary: $\exists$hasChild.Doctor
14. (John,a):hasChild
15. a:Person
16. (Mary,b):hasChild
17. b:Doctor
18. b:Person

Apply the $\forall$–rule to 14. and 6. :

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. John: HappyParent <concept assertion>
2. (John,Mary): hasChild <role assertion>
3. Mary: ¬Doctor
4. John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
5. John: Parent
6. John: $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
7. John: Person $\sqcap$ $\exists$hasChild.Person
8. Mary: Doctor $\sqcup$ $\exists$hasChild.Doctor
9. John:Person
10. John: $\exists$hasChild.Person
11. Mary:Doctor
12. $\bot$
13. Mary: $\exists$hasChild.Doctor
14. (John,a):hasChild
15. a:Person
16. (Mary,b):hasChild
17. b:Doctor
18. b:Person
19. a: Doctor $\sqcup$ $\exists$hasChild.Doctor

Apply the $\sqcup$–rule> to 19. : choice (branching) ➔ first a: Doctor, then a: $\exists$hasChild.Doctor

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

1. John: HappyParent <concept assertion>
2. (John,Mary): hasChild <role assertion>
3. Mary: ¬Doctor
4. John: Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
5. John: Parent
6. John: $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
7. John: Person $\sqcap$ $\exists$hasChild.Person
8. Mary: Doctor $\sqcup$ $\exists$hasChild.Doctor
9. John:Person
10. John: $\exists$hasChild.Person
11. Mary:Doctor
12. $\bot$
13. Mary: $\exists$hasChild.Doctor
14. (John,a):hasChild
15. a:Person
16. (Mary,b):hasChild
17. b:Doctor
18. b:Person
19. a: Doctor $\sqcup$ $\exists$hasChild.Doctor
20. a:Doctor

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

Model of K $\cup$ {Mary: $\neg$Doctor}*:*

I = (Δ^I,.^{I}), with:

Δ^I ={John,Mary,a,b},

Doctor^I={a,b},

Person^I={John,a,b},

happyParent^I={John},

Parent^I={John}

hasChild^I={(John,Mary),(John,a),(Mary,b))}

Algorithmes de vérification de satisfiabilité à base de tableaux sémantiques

- Dans les années 1990, une nouvelle classe d'algorithmes est apparue: les algorithmes de vérification de satisfiabilité à base de tableaux (tableau-based algorithms).
- Ces derniers raisonnent sur des LD dites expressives ou très expressives, mais en temps exponentiel.
- Cependant, en pratique, le comportement des algorithmes est souvent acceptable.
- L'expressivité accrue a ouvert la porte à de nouvelles applications telles que le Web sémantique.

Satisfiabilité

- Les algorithmes pour déterminer la satisfiabilité sont basés sur des techniques de calcul de tableaux.

- La complétude est une propriété très importante pour l'utilisation des LD dans des applications du monde réel.

- Ces algorithmes sont efficaces pour les bases de connaissances moyennes et réelles, même si leur complexité dans la logique est en PSPACE ou EXPTIME.

$$P \subseteq NP \subseteq PSpace \subseteq ExpTime \subseteq NExpTime \text{ (Padadimitriou, 1994).}$$

La méthode des tableaux sémantiques

• La méthode des tableaux sémantiques peut être considérée comme une procédure pour construire une interprétation satisfaisante de l'assertion d'un concept donné.

• Pour des raisons de commodité, nous nous concentrons sur les procédures qui travaillent avec des concepts sous forme normale négative (negation normal form).

Forme normale négative
(Negation normal form)

- Soit C un concept arbitraire dans ALC
- On dit que C est en forme normale négative (FNN) ssi ¬ apparaît seulement immédiatement devant le nom d'un concept
- La FNN peut être calculée en appliquant les règles suivantes:

$$\neg\neg C \Rightarrow_{\text{nnf}} C$$

$$\neg\top \Rightarrow_{\text{nnf}} \bot \qquad \neg\bot \Rightarrow_{\text{nnf}} \top$$

$$\neg(C \sqcup D) \Rightarrow_{\text{nnf}} \neg C \sqcap \neg D \qquad \neg(C \sqcap D) \Rightarrow_{\text{nnf}} \neg C \sqcup \neg D$$

$$\neg\forall R.C \Rightarrow_{\text{nnf}} \exists R.\neg C \qquad \neg\exists R.C \Rightarrow_{\text{nnf}} \forall R.\neg C$$

Forme normale négative

Exemple

Consider the concept $\neg(\text{student} \sqcap \text{happy})$.

Then

$$\neg(\text{student} \sqcap \text{happy}) \Rightarrow_{\text{nnf}} \neg\text{student} \sqcup \neg\text{happy}$$

Consider the concept $\neg(\text{mother} \sqcup (\neg\text{female} \sqcap \exists\text{hasChild.person}))$.

Then

$$\begin{aligned}
&\neg(\text{mother} \sqcup (\neg\text{female} \sqcap \exists\text{hasChild.person})) \\
&\Rightarrow_{\text{nnf}} \neg\text{mother} \sqcap \neg(\neg\text{female} \sqcap \exists\text{hasChild.person}) \\
&\Rightarrow_{\text{nnf}} \neg\text{mother} \sqcap ((\neg\neg\text{female}) \sqcup (\neg\exists\text{hasChild.person})) \\
&\Rightarrow_{\text{nnf}} \neg\text{mother} \sqcap (\text{female} \sqcup (\neg\exists\text{hasChild.person})) \\
&\Rightarrow_{\text{nnf}} \neg\text{mother} \sqcap (\text{female} \sqcup \forall\text{hasChild.}\neg\text{person})
\end{aligned}$$

La méthode des tableaux sémantiques comme système de réfutation

- La méthode des tableaux sémantiques est une procédure de preuve formelle, existant dans plusieurs logiques, mais toujours avec les mêmes caractéristiques
- La méthode des tableaux sémantiques est un système de réfutation : étant donné un système initial de contraintes ou tableau T, il tente soit de montrer que T est insatisfiable, soit d'en construire une interprétation satisfaisante (un modèle).
- Essentiellement, la méthode des tableaux sémantiques peut se résumer en une suite de séries d'affirmations construites sous forme d'un arbre, selon des règles d'inférence, chaque branche de l'arbre correspondant à une des séries d'affirmation.

La méthode des tableaux sémantiques comme système de réfutation

• Les règles d'inférence stipulent comment un tableau T est transformé en un nouveau tableau en transformant une branche en n nouvelles branches. Cela se fait en élargissant la branche à sa feuille, en créant jusqu'à n noeuds successeurs.

• Chacun des nouveaux noeuds contient de nouvelles affirmations.

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 noeuds successeurs.

• Le noeud successeur 1 contient X1, …, le nœud successeur n contient Xn.

• Les assertions dans X sont des prémisses, les assertions dans Xi sont des conclusions.

Règles d'expansion dans ALC

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

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

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

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

Side condition: Each rule is applied only once to the same instance of the numerator.

La règle d'expansion "et"

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

- Meaning of the $(\sqcap)$ rule:

  If a given tableau T contains an assertion $a : C \sqcap D$ then it can be extended to form a new tableau by adding both $a : C$ and $a : D$ to the branch containing $a : C \sqcap D$.

- If $a : C \sqcap D \in T$ and it has not been expanded as yet, then we say the $(\sqcap)$ rule is applicable to the branch in which $a : C \sqcap D$ occurs, or the $(\sqcap)$ rule is applicable to the tableau T.

- Similarly, for the other rules.

La règle d'expansion "ou"

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

- Meaning of the $(\sqcup)$ rule:

  If a tableau T contains an assertion $a : C \sqcup D$ then it can be extended to form a new tableau with two new branches. In particular, the branch containing $a : C \sqcup D$ is extended with two successor nodes containing $a : C$ and $a : D$, respectively.

- The $(\sqcup)$ rule is a branching rule, because it creates two new branches.
- The intuition of the rule is that if $a : C \sqcup D$ is true in an interpretation then either $a : C$ or $a : D$ must be true.

La règle d'expansion "CLASH"

- The $(\bot)$ rule is also called the clash rule or closure rule.

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

- Its meaning:

  If a branch in a tableau T contains two assertions $a : A$ and $a : \neg A$, where A denotes a concept name, then it can be extended to form a new tableau by adding $a : \bot$ to the branch.

- $a : \bot$ is not satisfiable in any interpretation. Thus once $a : \bot$ is derived then we have an inconsistency or a clash.
- A branch is said to be closed if it contains an assertion of the form $a : \bot$. Otherwise, the branch is open.
- A tableau is said to be closed if all its branches are closed. Otherwise, the tableau is open.

La règle d'expansion "some"

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

- Its meaning:

  If a tableau T contains an assertion $a : \exists R.C$ then it can be extended to form a new tableau by choosing a new object name b which does not appear in the branch and adding both $(a, b) : R$ and $b : C$ to the branch containing $a : \exists R.C$.

- The rule reflects our intuition that if $a : \exists R.C$ then a has an R-relative, namely b, in C.

La règle d'expansion "all"

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

- Its meaning:

  If a branch in a tableau T contains two assertions $a : \forall R.C$ and $(a, b) : R$ then it can be extended to form a new tableau by adding $b : C$ to the branch.

- The rule reflects our intuition that if $a : \forall R.C$ then a is the R-relative of nothing but b's in C, i.e. any b such that $(a, b) : R$ belongs to C.

La méthode des tableaux pour tester la satisfiabilité d'un concept

Pour résoudre le problème : est-ce que le concept C est 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 $a : \bot$, 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.

Validité et complétude de la méthode des tableaux

- 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 insatisfiable /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

Exemple

- Est-ce que le concept suivant est consistant /satisfiable ?

$$(\text{student} \sqcap \text{happy}) \sqcap (\neg\text{student} \sqcup \neg\text{happy})$$

- Déjà en FNN
- Tableau initial:

$$x : (\text{student} \sqcap \text{happy}) \sqcap (\neg\text{student} \sqcup \neg\text{happy})$$

Exemple (suite)

The $(\sqcap)$ rule is applicable to 1. So we expand the tableau and get:

1.	$x : (\text{student} \sqcap \text{happy}) \sqcap (\neg\text{student} \sqcup \neg\text{happy})$	given
2.	$x : \text{student} \sqcap \text{happy}$	by 1, $(\sqcap)$
3.	$x : \neg\text{student} \sqcup \neg\text{happy}$	by 1, $(\sqcap)$

Exemple (suite)

There is a choice: we can either expand 2. or 3.

We choose to apply the ($\sqcap$) rule to 2. and get:

1.	$x : (\text{student} \sqcap \text{happy}) \sqcap (\neg\text{student} \sqcup \neg\text{happy})$	given
2.	$x : \text{student} \sqcap \text{happy}$	by 1, ($\sqcap$)
3.	$x : \neg\text{student} \sqcup \neg\text{happy}$	by 1, ($\sqcap$)
4.	$x : \text{student}$	by 2, ($\sqcap$)
5.	$x : \text{happy}$	by 2, ($\sqcap$)

Exemple (suite)

Apply the ($\sqcup$) rule to 3. and get two branches:

1.	$x : (\text{student} \sqcap \text{happy}) \sqcap (\neg\text{student} \sqcup \neg\text{happy})$			given
2.	$x : \text{student} \sqcap \text{happy}$			by 1, ($\sqcap$)
3.	$x : \neg\text{student} \sqcup \neg\text{happy}$			by 1, ($\sqcap$)
4.	$x : \text{student}$			by 2, ($\sqcap$)
5.	$x : \text{happy}$			by 2, ($\sqcap$)
6.	$x : \neg\text{student}$	|	7. $x : \neg\text{happy}$	by 3, ($\sqcup$)

Exemple (suite)

Consider first the left branch. The clash rule is applicable to 4. and 6.

1.	$x : (\text{student} \sqcap \text{happy}) \sqcap (\neg\text{student} \sqcup \neg\text{happy})$		given
2.	$x : \text{student} \sqcap \text{happy}$		by 1, ($\sqcap$)
3.	$x : \neg\text{student} \sqcup \neg\text{happy}$		by 1, ($\sqcap$)
4.	$x : \text{student}$		by 2, ($\sqcap$)
5.	$x : \text{happy}$		by 2, ($\sqcap$)
6.	$x : \neg\text{student}$	7. $x : \neg\text{happy}$	by 3, ($\sqcup$)
8.	$x : \bot$ by 4, 6, clash		
	closed		

Exemple (suite)

Because the left branch is closed, we continue the derivation on the right branch. The clash rule is applicable to 5. and 7.:

1.	$x : (\text{student} \sqcap \text{happy}) \sqcap (\neg\text{student} \sqcup \neg\text{happy})$			given
2.	$x : \text{student} \sqcap \text{happy}$			by 1, ($\sqcap$)
3.	$x : \neg\text{student} \sqcup \neg\text{happy}$			by 1, ($\sqcap$)
4.	$x : \text{student}$			by 2, ($\sqcap$)
5.	$x : \text{happy}$			by 2, ($\sqcap$)
6.	$x : \neg\text{student}$	| 7.	$x : \neg\text{happy}$	by 3, ($\sqcup$)
8.	$x : \bot$ by 4, 6, clash	| 9.	$x : \bot$ by 5, 7, clash	
	closed	|	closed	

Exemple (suite)

All branches (there are two) of the tableau are closed (and no more rules are applicable).

1.	$x : (\text{student} \sqcap \text{happy}) \sqcap (\neg\text{student} \sqcup \neg\text{happy})$			given
2.	$x : \text{student} \sqcap \text{happy}$			by 1, ($\sqcap$)
3.	$x : \neg\text{student} \sqcup \neg\text{happy}$			by 1, ($\sqcap$)
4.	$x : \text{student}$			by 2, ($\sqcap$)
5.	$x : \text{happy}$			by 2, ($\sqcap$)
6.	$x : \neg\text{student}$	7.	$x : \neg\text{happy}$	by 3, ($\sqcup$)
8.	$x : \bot$ by 4, 6, clash	9.	$x : \bot$ by 5, 7, clash	
	closed		closed	

Exemple (suite)

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