L
I
P
6
C
N
R
S

# Logiques de description II

**LREC – Cours 4**

**Jean-Gabriel Ganascia**

*Rappel sur syntaxe et sémantique*

*Démonstration par la méthode des tableaux*

*Démonstration par la subsomption structurelle*

# Logiques terminologiques
# Logiques de description…

**Formalismes inspirés des représentations sémantiques** *(réseaux sémantiques, frame, graphes conceptuels, …)*

***TKRS***: *Terminological Knowledge Representation Systems*

**<u>Deux composants</u>**

- **Classes générales d'individus** – *T-Box*
  - **Propriétés générales des classes**
  - **Relations entre les classes**
- **Instanciation de ces schémas** – *A-Box*
  - **Assertions relatives à des individus**

# La « *famille* » des logiques de description

- **Une logique de description donnée et définie par des concepts, des roles et des opérateurs**
- **La logique $\mathcal{AL}$ (*Attribute Language*) contient uniquement la négation atomique et la quantification existentielle limitée**
  - **Les concepts sont construits en utilisant $\sqcap$, $\neg$, $\exists$ et $\forall$**
- **La plus petite logique de description contenant la logique propositionnelle est $\mathcal{ALC}$ (équivalent à la logique multimodale $K_{(m)}$) – cela signifie $\mathcal{AL}$ et complémentation $\mathcal{C}$**
  - **Les concepts sont construits en utilisant $\sqcap$, $\sqcup$, $\neg$, $\exists$ et $\forall$**
- **$\mathcal{FL}^-$ correspond à $\mathcal{AL}$ sans la négation atomique**
- **$\mathcal{FL}_0$ correspond à $\mathcal{FL}^-$ sans la quantification existentielle limitée**

# $\mathcal{FL}_0$: la plus simple logique de description

## Syntaxe

**Alphabet**

- **concepts atomiques A, B, C, D…**
- **Rôles atomiques r, s, u, v,**
- **Symboles {$\sqcap$, $\forall$, .}**

**Grammaire**

concept ::= <concept atomique> |
<concept> $\sqcap$ <concept> |
$\forall$<role atomic>.<concept>


Jean-Gabriel Ganascia – Faculté des sciences de Sorbonne Université

# $\mathcal{FL}^-$ Syntaxe

**Alphabet**

- **concepts atomiques A, B, C, D…**
- **Rôles atomiques r, s, u, v,**
- **Symboles {$\sqcap$, $\exists$, $\forall$, .}**

**Grammaire**

concept ::= <concept atomique> |
<concept> $\sqcap$ <concept> |
$\exists$<role atomique> |
$\forall$<role atomic>.<concept>

# $\mathcal{EL}$: logique de description minimale - existentielle

## Syntaxe

**Alphabet**

- **concepts atomiques A, B, C, D…**
- **Rôles atomiques r, s, u, v, …**
- **Symboles {⊓,∃, .}**

**Grammaire**

concept ::= <concept atomique> |
<concept> ⊓ <concept> |
∃<role atomic>.<concept>

# $\mathcal{AL}$ Syntaxe

**Alphabet**

- **concepts atomiques A, B, C, D…**
- **Rôles atomiques r, s, u, v, …**
- **Symboles {$\sqcap$, $\exists$, $\forall$, $\neg$, .}**

**Grammaire**

concept ::= <concept atomique> |
<concept> $\sqcap$ <concept> |
$\exists$<role atomique> |
$\neg$<concept atomique> |
$\forall$<role atomic>.<concept>

# $\mathcal{FL}^-$: base de connaissance $\Sigma$ = <TBox, ABox>

- **TBox: axiomes terminologiques C $\sqsubseteq$ D, C $\equiv$ D**
  - **Définitions**
    Parent≡∃aENFANT ⊓ Personne
  - **Subsomptions**
    Homme ⊑ Personne *(⊑: subsomption)*
- **ABox: assertions a:C, <a, b>:R**
  - **Assertions de concepts**
    Jean:Parent
    Jean:Personne ⊓∃aENFANT
    Albert:personne
  - **Assertions de rôles**
    <Jean, Thomas>:aENFANT

# $\mathcal{ALC}$: la plus simple des logiques de description propositionnelles

## Alphabet

- **Ensemble de concepts atomiques A, B, C, D…**
- **Ensemble de rôles atomiques R, S, U, V, …**
- **Symboles {$\sqcup$,$\sqcap$,$\exists$,$\forall$,$\neg$,$\top$,$\bot$,.}**

## Grammaire

  - **$\top$ et $\bot$ sont des concepts**
- **Si C et D sont des concepts:**
  - **$\neg$C est un concept** *(et pas uniquement un concept atomique)*
  - **C $\sqcup$ D et C $\sqcap$ D sont des concepts**
- **Si r est un rôle et C un concept**
  - **$\forall$r.C et $\exists$r.C sont des concepts**

# La « *famille* » des logiques de description: les extensions de $\mathcal{AL}$

- $\mathcal{S}$ est souvent utilisé pour dénoter $\mathcal{ALC}$ avec roles transitifs ($R_+$)
- Des lettres additionnelles indiquent d'autres extensions:
  - $\mathcal{H}$ pour les axiomes d'inclusion de rôles (hiérarchie de rôles aFille $\sqsubseteq$ aEnfant)
  - $\mathcal{O}$ pour noms (classes nominales – singleton, exemple: {Italie})
  - $\mathcal{I}$ pour les rôles inverses (estEnfantDe $\equiv$ aEnfant$^{-1}$)
  - $\mathcal{N}$ pour les restrictions sur les nombres (forme $\exists^{\leq n}r$ ou $\exists^{\geq n}r$)
  - $\mathcal{Q}$ pour les restrictions qualifiées sur les nombres (forme $\exists^{\leq n}r.C$ ou $\exists^{\geq n}r.C$)
- p.e. OWL est $\mathcal{ALC}$ + $R_+$ + hiérarchie de rôles + classes nominales + inversion de rôles + restrictions qualitifées sur les nombres = $\mathcal{SHOIQ}$

# $\mathcal{FL}^-$: Sémantique formelle

**Une interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ consiste en**

- **Un ensemble non vide $\Delta^{\mathcal{I}}$ (le *domaine*)**
- **Un fonction (la *fonction d'interprétation*) qui associe**
  - **À tout concept C, un sous-ensemble $C^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$**
  - **À tout rôle R, un sous-ensemble $R^{\mathcal{I}}$ de $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$**
  - **À tout individu i, un élément $i^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$**

# Sémantique $\mathcal{ALC}$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$
$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

$$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$
$$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.(x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$$

# Sémantique générale

$$
\begin{array}{lcl}
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
\{x\}^{\mathcal{I}} & = & \{x^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} & = & \{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} & = & \{x \mid \forall y.(x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\} \\
(\leqslant n R)^{\mathcal{I}} & = & \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \leqslant n\} \\
(\geqslant n R)^{\mathcal{I}} & = & \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \geqslant n\} \\
(R^{-})^{\mathcal{I}} & = & \{(x, y) \mid (y, x) \in R^{\mathcal{I}}\}
\end{array}
$$

# Qu'est ce que ∀r.C et ∃r.C signifient?

- Un « FouDeChiens » est quelqu'un dont les animaux de compagnie sont tous des chiens, ici {C}

  FouDeChiens = ∀hasPet.Dog

  $\{p \mid \forall\ a, \langle p, a\rangle \in hasPet \rightarrow a \in Dog\}$

  On peut l'écrire plus simplement:

  $\{p \mid \forall\ a, hasPet(p, a) \rightarrow Dog(a)\ \}$

- Un « AmateurDeChiens » est quelqu'un qui possède un chien , ici {A, C}

  AmateurDeChiens = ∃hasPet.Dog

  $\{p \mid \exists a\ hasPet(p, a)\ \&\ Dog(a)\ \}$

| hasPet | |
|---|---|
| A | Fido |
| A | Fluffy |
| B | Tabby |
| C | Rover |
| C | Flip |

| Cat |
|---|
| Fluffy |
| Tabby |

| Dog |
|---|
| Fido |
| Rover |
| Flip |

LIP6 CNRS

# Pas de variables…

$\text{Homme} \sqcap \neg\text{Femme} \sqcap (\exists \text{marié.Médecin}) \sqcap (\forall \text{aEnfant.(Médecin} \sqcup \text{Avocat))}$

- Symbole union ($\sqcup$ ) et intersection ($\sqcap$) de concepts
- **Quantificateur existentiel**: $\exists$marié.Médecin
  - Ensemble des individus mariés à **au moins** un médecin
- **Quantificateur universel**: ($\forall$aEnfant.(Médecin $\sqcup$ Avocat))
  - Ensemble des individus dont **tous** les enfants sont soit médecin, soit avocats
- **Axiomes:** $\exists$aEnfant.Humain $\sqsubseteq$ Humain (**subsomption**)
  - Seuls les êtres humains peuvent avoir des enfants humains
- **Axiomes:** Père $\equiv$ Homme $\sqcap\exists$aEnfant.T

L I P 6 C N R S

# Exemple: formalisation en $\mathcal{ALCN}$

a) Un alexandrin est un vers

b) Il existe deux formes distinctes d'alexandrin : le trimètre et le tétramètre

c) Un alexandrin comporte deux hémistiches et douze syllabes

d) Chaque hémistiche comporte 6 syllabes

e) Tous les alexandrins comportent une césure à la fin de l'hémistiche

f) "Je courus! Et les péninsules démarrées" est un vers écrit par Rimbaud qui a 12 syllabes

g) "Je courus ! Et les péninsules démarrées" ne comporte pas de césure à l'hémistiche

L I P 6 C N R S

# Exemple: formalisation en $\mathcal{ALCN}$

a) Un alexandrin est un vers

b) Il existe deux formes distinctes d'alexandrin: le trimètre et le tétramètre

c) Un alexandrin comporte deux hémistiches et douze syllabes

a) *Alexandrin* $\sqsubseteq$ *Vers*

b) *Trimètre* $\sqsubseteq$ *Alexandrin*
*Tétramètre* $\sqsubseteq$ *Alexandrin*
*Trimètre* $\sqcap$ *Tétramètre* $\sqsubseteq \perp$

c) *Alexandrin* $\sqsubseteq$
$\exists^{\geq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\leq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 12}$ *contient.Syllabe*

# Exemple: formalisation en $\mathcal{ALCN}$

d) Chaque hémistiche comporte 6 syllabes

e) Tous les alexandrins comportent une césure à la fin de l'hémistiche

*d)* $\mathit{Hémistiche} \sqsubseteq \exists^{\geq 6}\, \mathit{contient.Syllabe} \sqcap \exists^{\leq 6}\, \mathit{contient.Syllabe}$

*c)* $\mathit{Alexandrin} \sqsubseteq \exists \mathit{césure.Hémistiche}$

*Rq. césure est un rôle; le second argument comporte la position*

L I P 6 C N R S

# Exemple: formalisation en $\mathcal{ALCN}$

f) “Je courus! Et les péninsules démarrées” est un vers écrit par Rimbaud qui a 12 syllabes

*f)* *<« Je courus! Et les péninsules démarées », Rimbaud>: Auteur*

*« Je courus! Et les péninsules démarées »:Vers* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 12}$ *contient.Syllabe*

g) “Je courus ! Et les péninsules démarrées” ne comporte pas de césure à l’hémistiche

*g)* *« Je courus! Et les péninsules démarées »:*
$\neg\ \exists$*césure.Hémistiche*

# Raisonnement: 4 propriétés

- **Satisfiabilité:** un concept C est satisfiable si et seulement si il existe une interprétation $\mathcal{I}$ telle que $C^{\mathcal{I}} \neq \emptyset$
- **Subsomption:** un concept C est subsumé par D si et seulement si $C^{\mathcal{I}} \subset D^{\mathcal{I}}$ pour toute interprétation $\mathcal{I}$

*Remarque: la subsomption est décidable en temps polynomial pour $\mathcal{FL}^-$*

- **Équivalence:** un concept C est équivalent à un concept D si et seulement si $C^{\mathcal{I}} = D^{\mathcal{I}}$ pour toute interprétation $\mathcal{I}$
- **Incompatibilité:** deux concepts C et D sont incompatibles si et seulement si $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ pour toute interprétation $\mathcal{I}$

# Quatre problèmes

- Satisfiabilité d'une base de connaissance: est-ce que la ABox et la Tbox sont cohérentes l'une avec l'autre?
- Satisfiabilité d'un concept: étant donné une base de connaissance K et un concept C existe-t-il au moins un modèle de K pour lequel l'extension de C n'est pas vide?
- Subsomption: étant donné une base de connaissance K et deux concepts quelconque C et D de K, est-ce que D est plus général que C (ou est-ce que C est subsumé par D)?
- Vérification d'une instance: étant donné une base de connaissance K et une instance a d'un concept C, est-ce que a est une instance de C dans tout modèle de K?

# Réductions à la Subsumption

- C est <u>insatisfiable</u> ssi $C \sqsubseteq \bot$
- C & D sont <u>équivalents</u> ssi $C \sqsubseteq D$ & $D \sqsubseteq C$
- C & D sont <u>incompatibles</u> ssi $C \sqcap D \sqsubseteq \bot$

# Réductions à l'insatisfiabilité
## Possible si la négation est définie…

- **C $\sqsubseteq$ D (C est <u>subsumé</u> par D) ssi C$\sqcap\neg$D insatisfiable**
- **C & D sont <u>équivalents</u> ssi C$\sqcap\neg$D & $\neg$C$\sqcap$D sont insatisfiables**
- **C & D sont <u>incompatibles</u> ssi C$\sqcap$D est insatisfiable**

# Procédures de raisonnement

- **Il existe des algorithmes complets et efficaces pour décider de l'insatisfiabilité, de la subsomption et donc des autres propriétés de certaines logiques de descriptions**

- **Les techniques de raisonnement sont fondées sur la méthode des tableaux (pour l'insatisfiabilité)**
  **sur la subsomption structurelle (pour la subsomption)**
  **sur les automates**
  **sur la résolution…**

*Remarque: il existe des liens entre la méthode des tableaux et d'autres techniques (résolution, ASP, etc.)*

- **La complétude est importante pour l'utilisation des logiques de descriptions dans les applications réelles**

L I P 6 C N R S

L
I
P
6

# Pourquoi y a-t-il un problème?

Personn

## ABox

Françoise:Femme

<Françoise, 26>:Age

<Françoise, Gil>:Enfant

<Françoise,Jean>:Enfant

## TBox

$\text{Homme} \sqsubseteq \text{Personne}$

$\text{Femme} \sqsubseteq \text{Personne}$

$\text{Personne} \sqsubseteq \exists \text{Age}$

$\text{Parent} \equiv \exists \text{Enfant} \sqcap \text{Personne}$

$\text{Père} \equiv \exists \text{Enfant} \sqcap \text{Homme}$

$\text{Mère} \equiv \exists \text{Enfant} \sqcap \text{Femme}$

$\text{Père} \equiv \text{Parent} \sqcap \text{Homme}$

$\text{Mère} \equiv \text{Parent} \sqcap \text{Femme}$

Instances

Françoise

n

# En PROLOG: incomplétude

**TBox**

Homme $\sqsubseteq$ Personne
Femme $\sqsubseteq$ Personne
Personne $\sqsubseteq \exists$Age
Parent $\equiv \exists$Enfant $\sqcap$ Personne
Père $\equiv \exists$Enfant $\sqcap$ Homme
Mère $\equiv \exists$Enfant $\sqcap$ Femme
Père $\equiv$ Parent $\sqcap$ Homme
Mère $\equiv$ Parent $\sqcap$ Femme

**ABox**

Françoise:Femme
<Françoise, 26>:Age
<Françoise, Gil>:Enfant
<Françoise,Jean>:Enfant

```
?- femme(eve).
?- homme(adam).
```

```
femme(francoise).
age(francoise, 26).
enfant(francoise, gil).
enfant(francoise, jean).
```

```
personne(X):- homme(X).
personne(X) :- femme(X).
age(X,A) :- personne(X).
parent(X) :- enfant(X,Y),personne(X).
mere(X) :- parent(X), femme(X).
pere(X) :- enfant(X, Y), homme(X).
```

```
mere(eve).
pere(adam).
personne(eve).
personne(adam).
```

# Complexité: compromis

- Plus le pouvoir d'expression d'un langage est grand, plus les procédures de démonstration sont complexes
- En général les langages pour lesquels des procédures complètes existent sont insuffisants pour exprimer ce que l'on souhaite

| Description logic | Subsumption computation complexity |
|---|---|
| $\mathcal{AL}$ | PTIME |
| $\mathcal{ALC}$ | PSPACE |
| $\mathcal{SHIF}$ | EXPTIME |
| $\mathcal{SHOIN}$ | NEXPTIME |

- Questions:
  – Un langage étant donné, existe-t-il une procédure de démonstration complète?
  – Si ce n'est pas le cas, peut-on comparer les procédures incomplètes?
  – Une procédure complète étant donnée, existe-t-il des algorithmes efficaces en temps et en place?

# Navigateur pour la description de la complexité des procédures de preuve

http://www.cs.man.ac.uk/~ezolin/dl/

## Complexity of reasoning in Description Logics

**Note: the information here is (always) incomplete and updated often**

Base description logic: $\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements

$\mathcal{ALC} ::= \bot \mid \top \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$

**Concept constructors:**

- ☐ $\mathcal{F}$ – functionality[2]: $(\leq 1\ R)$
- ☐ $\mathcal{N}$ – (unqualified) number restrictions: $(\geq n\ R)$, $(\leq n\ R)$
- ☐ $\mathcal{Q}$ – qualified number restrictions: $(\geq n\ R.C)$, $(\leq n\ R.C)$
- ☐ $\mathcal{O}$ – nominals: $\{a\}$ or $\{a_1, \ldots, a_n\}$ ("one-of")
- ☐ $\mu$ – least fixpoint operator: $\mu X.C$

Forbid ⌃ complex roles[5] in number restrictions[6]

**Role constructors:** trans reg

- ☐ $\mathcal{I}$ – role inverse: $R^-$
- ☐ $\sqcap$ – role intersection[3]: $R \sqcap S$
- ☐ $\sqcup$ – role union: $R \sqcup S$
- ☐ $\neg$ – role complement: $\neg R$ full ⌃
- ☐ $\circ$ – role chain (composition): $R \circ S$
- ☐ $*$ – reflexive-transitive closure[4]: $R^*$
- ☐ $id$ – concept identity: $id(C)$

**TBox (concept axioms):**

- ◉ empty TBox
- ○ acyclic TBox ($A \equiv C$, $A$ is a concept name; no cycles)
- ○ general TBox ($C \sqsubseteq D$, for arbitrary concepts $C$ and $D$)

**RBox (role axioms):** OWL-Lite OWL-DL OWL 1.1

- ☐ $\mathcal{S}$ – role transitivity: $Tr(R)$
- ☐ $\mathcal{H}$ – role hierarchy: $R \sqsubseteq S$
- ☐ $\mathcal{R}$ – complex role inclusions: $R \circ S \sqsubseteq R$, $R \circ S \sqsubseteq S$
- ☐ $s$ – some additional features (click to see them)

Reset You have selected a Description Logic: $\mathcal{ALC}$

# $\mathcal{ALC}$ + empty TBox

**TBox (concept axioms):**

- (selected) empty TBox
- acyclic TBox ($A \equiv C$, $A$ is a concept name; no cycles)
- general TBox ($C \sqsubseteq D$, for arbitrary concepts $C$ and $D$)

**RBox (role axioms):**

OWL-Lite
OWL-DL
OWL 1.1

- $\mathcal{S}$ – role transitivity: $Tr(R)$
- $\mathcal{H}$ – role hierarchy: $R \sqsubseteq S$
- $\mathcal{R}$ – complex role inclusions: $R \circ S \sqsubseteq R$, $R \circ S \sqsubseteq S$
- $s$ – some additional features (click to see them)

Reset

You have selected a Description Logic: $\mathcal{ALC}$

| Complexity[7] of reasoning problems[8] | | |
|---|---|---|
| Concept satisfiability | **PSpace-complete** | • Hardness for $\mathcal{ALC}$: see [80].<br>• Upper bound for $\mathcal{ALCQ}$: see [12, Theorem 4.6]. |
| ABox consistency | **PSpace-complete** | • Hardness follows from that for concept satisfiability.<br>• Upper bound for $\mathcal{ALCQO}$: see [17, Appendix A]. |
| **Important properties of the Description Logic** | | |
| Finite model property | **Yes** | $\mathcal{ALC}$ is a notational variant of the multi-modal logic $\mathbf{K}_m$ (cf. [77]), for which the finite model property can be found in [4, Sect. 2.3]. |
| Tree model property | **Yes** | $\mathcal{ALC}$ is a notational variant of the multi-modal logic $\mathbf{K}_m$ (cf. [77]), for which the tree model property can be found in [4, Proposition 2.15]. |

# $\mathcal{ALC}$ + acyclic TBox

| TBox (concept axioms): | RBox (role axioms): |
|---|---|
| ○ empty TBox<br>◉ acyclic TBox ($A \equiv C$, $A$ is a concept name; no cycles)<br>○ general TBox ($C \subseteq D$, for arbitrary concepts $C$ and $D$) | OWL-Lite<br>OWL-DL<br>OWL 1.1<br>☐ $\mathcal{S}$ – role transitivity: $Tr(R)$<br>☐ $\mathcal{H}$ – role hierarchy: $R \subseteq S$<br>☐ $\mathcal{R}$ – complex role inclusions: $R \circ S \subseteq R$, $R \circ S \subseteq S$<br>☐ $s$ – some additional features (click to see them) |

Reset You have selected a Description Logic: $\mathcal{ALC}$

| Complexity[7] of reasoning problems[8] | | |
|---|---|---|
| Concept satisfiability | **PSpace-complete** | • Hardness: see empty TBox.<br>• Upper bound for $\mathcal{ALCQO}$: see [17, Appendix A].<br>• An automata-based PSpace algorithm for $\mathcal{ALC}$ with acyclic TBoxes is given in [44]. |
| ABox consistency | **PSpace-complete** | • Hardness follows from that for concept satisfiability.<br>• Upper bound for $\mathcal{ALCQO}$: see [17, Appendix A]. |
| **Important properties of the Description Logic** | | |
| Finite model property | **Yes** | For all sublogics of $\mathcal{SHOQ}$. This is mentioned in [63], where a similar result is obtained in Corollary 4.3 for $\mathcal{SHOQ}$ extended with concrete domains and keys. (I did not find a "proper" reference for $\mathcal{SHOQ}$ or its sublogics.) |
| Tree model property | **Yes** | For all sublogics of $\mathcal{ALCFI}_{reg}$ with any TBoxes; see [2, p.189, Theorem 5.6]. |

# Tbox définitoire – acyclique

- Une **inclusion générale de concepts** est de la forme $C \sqsubseteq D$ où C et D sont des concepts.
- Une interprétation $\mathcal{I}$ est un modèle de si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- $\mathcal{I}$ est un modèle de la Tbox $\mathcal{T}$ si c'est un modèle de toutes les inclusions de concepts de $\mathcal{T}$.
- $C \equiv D$ est une abréviation pour $C \sqsubseteq D$ et $D \sqsubseteq C$
- Un axiome de la forme $A \equiv C$ où A est un nom de concept est appelé une **définition** de A
- Une Tbox $\mathcal{T}$ est dite **définitoire** si elle ne contient que des définitions avec les restrictions additionnelles suivantes:
  - $\mathcal{T}$ contient au plus une définition pour chaque nom de concept
  - $\mathcal{T}$ est acyclique

# $\mathcal{ALC}$ + arbitrary TBox

| **TBox (concept axioms):** | **RBox (role axioms):** |
|---|---|
| ○ empty TBox<br>○ acyclic TBox ($A \equiv C$, $A$ is a concept name; no cycles)<br>◉ general TBox ($C \sqsubseteq D$, for arbitrary concepts $C$ and $D$) | OWL-Lite<br>OWL-DL<br>OWL 1.1<br>☐ $\mathcal{S}$ – role transitivity: $Tr(R)$<br>☐ $\mathcal{H}$ – role hierarchy: $R \sqsubseteq S$<br>☐ $\mathcal{R}$ – complex role inclusions: $R \circ S \sqsubseteq R$, $R \circ S \sqsubseteq S$<br>☐ $s$ – some additional features (click to see them) |

Reset You have selected a Description Logic: $\mathcal{ALC}$

| **Complexity[7] of reasoning problems[8]** | | |
|---|---|---|
| Concept satisfiability | **ExpTime-complete** | • Hardness: originally proved in [77]; see also [2, Theorem 3.27].<br>• Upper bound: an ExpTime tableaux algorithm is given in [33]. |
| ABox consistency | **ExpTime-complete** | • Hardness follows from ExpTime-hardness of concept satisfiability w.r.t. general TBoxes.<br>• Upper bound even for $\mathcal{SHIQ}$ was proved in [12, Corollary 6.30]. |
| **Important properties of the Description Logic** | | |
| Finite model property | **Yes** | For all sublogics of $\mathcal{SHOQ}$. This is mentioned in [63], where a similar result is obtained in Corollary 4.3 for $\mathcal{SHOQ}$ extended with concrete domains and keys. (I did not find a "proper" reference for $\mathcal{SHOQ}$ or its sublogics.) |
| Tree model property | **Yes** | For all sublogics of $\mathcal{ALCFI}_{reg}$ with any TBoxes; see [2, p.189, Theorem 5.6]. |

# Une méthode des tableaux en logique des propositions - $\mathcal{ALC}$

Soit une description de concept $C_0$ mise sous *forme normale négative*. Ceci signifie que l'on rentre les négations (voir Lotrec)

L'algorithme commence avec la ABox $\mathcal{A}_0 := \{C_0(x_0)\}$

Il applique systématiquement des règles de transformation qui préservent la cohérence

Soit une suite finie de ABox $\mathcal{S} = \{\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2 \ldots, \mathcal{A}_k\}$

Cet ensemble est cohérent ssi $\exists i, 0 \leq i \leq k$ tel que est $\mathcal{A}_i$ cohérent et complète (on ne peut plus appliquer de transformation).

Lorsque les règles de transformation sont appliquées à $\mathcal{S}$, l'algorithme prend une ABox de $\mathcal{S}$ et la remplace

- soit par une nouvelle ABox $\mathcal{A}'$ (règle α)
- soit par deux nouvelles ABox $\mathcal{A}'$ et $\mathcal{A}''$ (règle β)

# Méthode des tableaux appliquée à la logique de description $\mathcal{ALC}$

**Première étape:** les expressions du niveau terminologique (TBox) sont normalisées en repoussant les négations devant les concepts atomiques (*mise sous forme normale négative*)

**Application des règles de normalisation**:

$\neg\neg\varphi \equiv \varphi$

$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$

$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$

$\neg\forall r.C \equiv \exists r.\neg C$

$\neg\exists r.C \equiv \forall r.\neg C$

$\neg(A \sqsubseteq B) \equiv A \sqcap \neg B$

$A \sqsubseteq B \equiv \neg A \sqcup B$

# Méthode des tableaux appliquée à la logique de description $\mathcal{ALC}$

**Deuxième étape:**

## Construction tableau initial:

Pour tout les faits C(x) de la ABox, on ajoute x:C dans le tableau

# Méthode des tableaux pour $\mathcal{ALCN}$

1. On veut montrer que $C_0$ (sous forme normal) est satisfiable

2. On cherche un modèle de la Abox A = $\{x_0:C_0\}$, $x_0$ étant un nouveau symbole de constante
    1. Appliquer les règles de transformation
    2. Si, à un moment, une ABox <u>complète</u> est engendrée, alors $C_0$ est satisfiable

3. Si aucune ABox complète n'est trouvé, $C_0$ est insatisfiable

# Méthode des tableaux pour $\mathcal{ALC}$
# Les règles pour les connecteurs

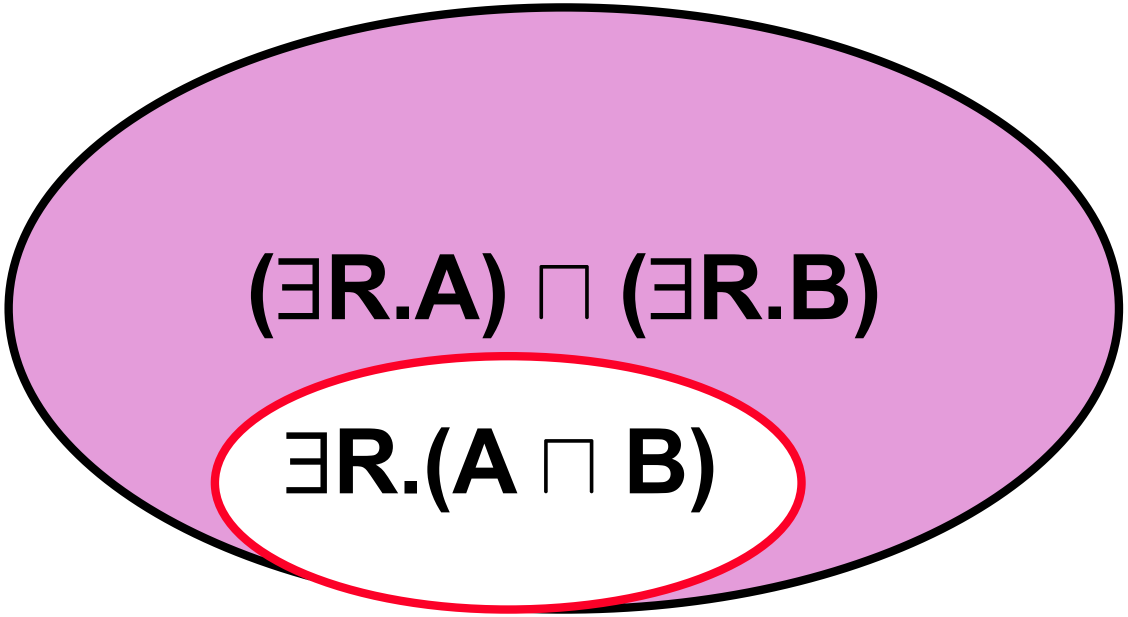
- $\sqcap$-règle (règle α):
  - **Condition:** $\mathcal{A}$ contient $(C1 \sqcap C2)(x)$, mais on n'a pas $C1(x)$ et $C2(x)$
  - **Action:** $\mathcal{A}' := \mathcal{A} \cup \{C1(x), C2(x)\}$
- $\sqcup$-règle (règle β):
  - **Condition:** $\mathcal{A}$ contient $(C1 \sqcup C2)(x)$, mais ni $C1(x)$ ni $C2(x)$
  - **Action (choix non-déterministe):**

    $\mathcal{A}' := \mathcal{A} \cup \{C1(x)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{C2(x)\}$

# Méthode des tableaux pour $\mathcal{ALC}$
# Règles pour les quantificateurs

- **∃-règle:**
  - **Condition:** $\mathcal{A}$ contient (∃r.C)(x), mais il n'existe pas de constante z telle que C(z) et r(x,z) dans $\mathcal{A}$
  - **Action:** $\mathcal{A}'$ := $\mathcal{A} \cup$ {C(z),r(x,z)} z étant une constante n'apparaissant pas déjà dans $\mathcal{A}$
- **∀-règle:**
  - **Condition:** $\mathcal{A}$ contient (∀r.C)(x) et r(x,y), mais C(y) n'est pas dans $\mathcal{A}$
  - **Action:** $\mathcal{A}'$ = $\mathcal{A} \cup$ {C(y)}

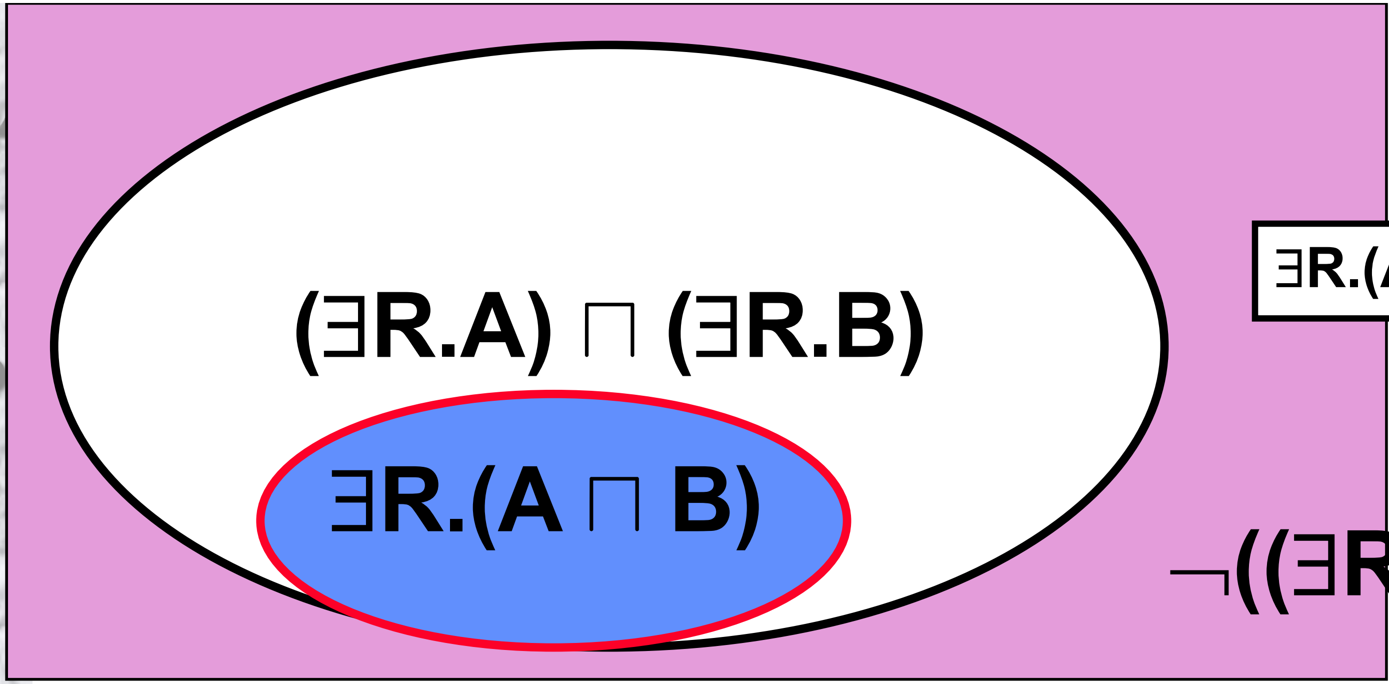
# Exemple




- $\exists R.(A \sqcap B) \sqsubseteq (\exists R.A) \sqcap (\exists R.B)$

# Exemple

$\boxed{\exists R.(A \sqcap B) \sqsubseteq (\exists R.A) \sqcap (\exists R.B)}$




- **Prouver $\exists R.(A \sqcap B) \sqsubseteq (\exists R.A) \sqcap (\exists R.B)$ valide est équivalent à prouver que sa négation, $\neg(\exists R.(A \sqcap B) \sqsubseteq (\exists R.A) \sqcap (\exists R.B))$** est **insatisfiable**

*Remarque*: $\neg(p \sqsubseteq q)$ *est équivalent à* $(p \sqcap \neg q)$

- **Il faut donc prouver l'insatisfiabilité de $\exists R.(A \sqcap B) \sqcap \neg((\exists R.A) \sqcap (\exists R.B))$**

# Rappel règles de normalisation

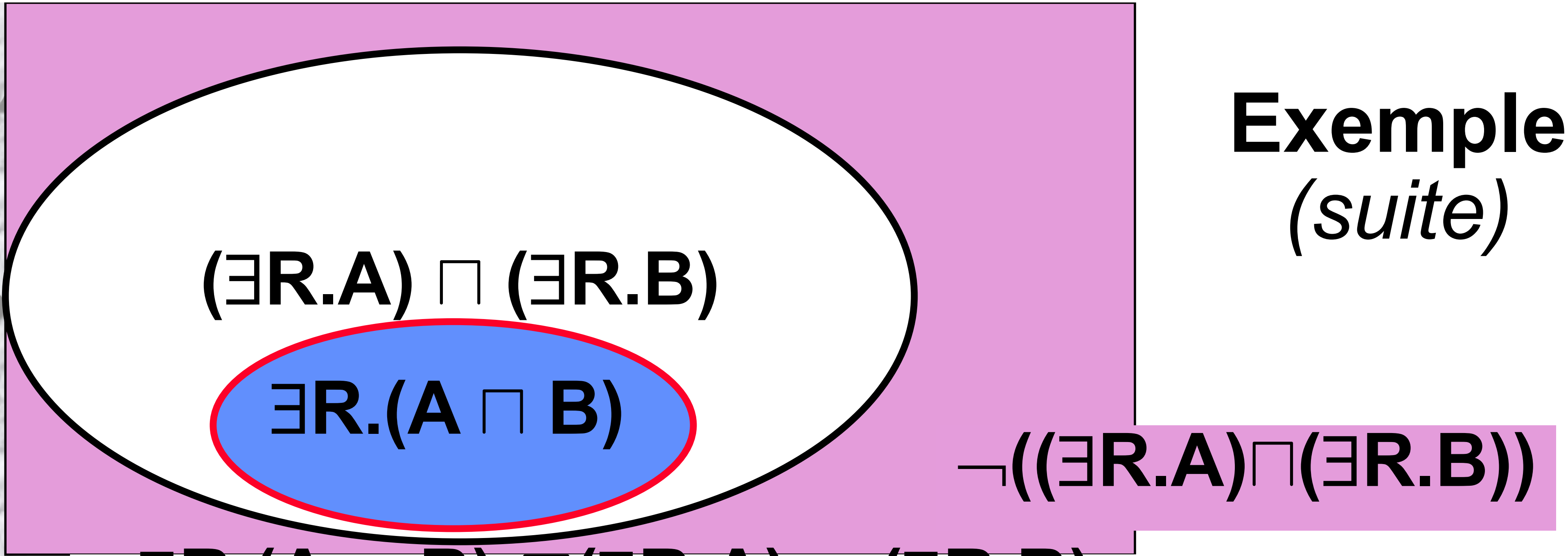
$\neg\forall r.C \equiv \exists r.\neg C$

$\neg\exists r.C \equiv \forall r.\neg C$

$\neg(A \sqsubseteq B) \equiv A \sqcap \neg B$

$A \sqsubseteq B \equiv \neg A \sqcup B$

## Exemple
*(suite)*




- $\exists R.(A \sqcap B) \sqsubseteq (\exists R.A) \sqcap (\exists R.B)$
- **Négation:** $\neg(\exists R.(A \sqcap B) \sqsubseteq (\exists R.A) \sqcap (\exists R.B))$
- $\exists R.(A \sqcap B) \sqcap \neg((\exists R.A) \sqcap (\exists R.B))$ **insatisfiable**
- **Etape 1: normalisation (application règles)**
- $\exists R.(A \sqcap B) \sqcap (\neg(\exists R.A) \sqcup \neg(\exists R.B))$
- $\exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B))$

# Exemple - suite

## Etape 2: tableaux

$\exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$\downarrow$ **$\sqcap$-règle**

$\exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$\exists R.(A \sqcap B)$

$((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$\downarrow$ **$\sqcup$-règle 1**

$\exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$\exists R.(A \sqcap B)$

$((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$(\forall R.\neg A)$

# Exemple - suite

**$\sqcup$-règle 1**

$\exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B))$
$\exists R.(A \sqcap B)$
$((\forall R.\neg A) \sqcup (\forall R.\neg B))$
$(\forall R.\neg A)$

**$\exists$-règle**

$\exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B))$
$\exists R.(A \sqcap B)$
$((\forall R.\neg A) \sqcup (\forall R.\neg B))$
$(\forall R.\neg A)$
$R(x,y)$
$(A \sqcap B)(y)$

- **$\exists$-règle:**
  - **Condition:** $\mathcal{A}$ contient $(\exists R.C)(x)$, mais il n'existe pas de constante z telle que $C(z)$ et $R(x,z)$ dans $\mathcal{A}$
  - **Action:** $\mathcal{A}' := \mathcal{A} \cup \{C(z), R(x,z)\}$ z étant une constante n'apparaissant pas déjà dans $\mathcal{A}$
- **$\forall$-règle:**
  - **Condition:** $\mathcal{A}$ contient $(\forall R.C)(x)$ et $R(x,y)$, mais $C(y)$ n'est pas dans $\mathcal{A}$
  - **Action:** $\mathcal{A}' = \mathcal{A} \cup \{C(y)\}$

# Exemple - suite

$\exists$-règle

$\exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$\exists R.(A \sqcap B)$

$((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$(\forall R.\neg A)$

$R(x,y)$, $(A \sqcap B)(y)$

$\forall$-règle

$\exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$\exists R.(A \sqcap B)$

$((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$(\forall R.\neg A)$

$\neg A(y)$

**R(x,y)**, $(A \sqcap B)(y)$

- **$\exists$-règle:**
  - **Condition:** $\mathcal{A}$ contient $(\exists R.C)(x)$, mais il n'existe pas de constante z telle que $C(z)$ et $R(x,z)$ dans $\mathcal{A}$
  - **Action:** $\mathcal{A}' := \mathcal{A} \cup \{C(z), R(x,z)\}$ z étant une constante n'apparaissant pas déjà dans $\mathcal{A}$
- **$\forall$-règle:**
  - **Condition:** $\mathcal{A}$ contient $(\forall R.C)(x)$ et $R(x,y)$, mais $C(y)$ n'est pas dans $\mathcal{A}$
  - **Action:** $\mathcal{A}' = \mathcal{A} \cup \{C(y)\}$

L I P 6 C N R S

# Exemple - suite

$\forall$-règle

$\exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$\exists R.(A \sqcap B)$

$((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$(\forall R.\neg A)$

$R(x,y)$, $\neg A(y)$

$(A \sqcap B)(y)$

$\sqcap$-règle

$\exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$\exists R.(A \sqcap B)$

$((\forall R.\neg A) \sqcup (\forall R.\neg B))$

$(\forall R.\neg A)$

$R(x,y)$ $\neg A(y)$

$(A \sqcap B)(y)$, $A(y)$, $B(y)$

contradiction

# Démonstration avec la méthode des tableaux

$Homme \sqsubseteq Personne$
$Femme \sqsubseteq Personne$
$Personne \sqsubseteq \exists Age$
$Parent \equiv \exists Enfant \sqcap Personne$
$Père \equiv \exists Enfant \sqcap Homme$
$Mère \equiv \exists Enfant \sqcap Femme$
$Père \equiv Parent \sqcap Homme$
$Mère \equiv Parent \sqcap Femme$

Eve:Mère
Adam:Pere

$Homme \sqsubseteq Personne$
$Femme \sqsubseteq Personne$
$Personne \sqsubseteq \exists Age$
$Parent \equiv \exists Enfant \sqcap Personne$
$Père \equiv Parent \sqcap Homme$
$Mère \equiv Parent \sqcap Femme$

Eve:Mère
Adam:Pere

$Homme \sqsubseteq Personne$
$Femme \sqsubseteq Personne$
$Personne \sqsubseteq \exists Age$
$Parent \sqsubseteq \exists Enfant \sqcap Personne$
$Père \sqsubseteq Parent \sqcap Homme$
$Mère \sqsubseteq Parent \sqcap Femme$

Eve:Mère
Adam:Pere

$\exists Enfant \sqcap Personne \sqsubseteq Parent$
$Parent \sqcap Homme \sqsubseteq Père$
$Parent \sqcap Femme \sqsubseteq Mère$

# Démonstration avec la méthode des tableaux

$Homme \sqsubseteq Personne$
$Femme \sqsubseteq Personne$
$Personne \sqsubseteq \exists Age$
$Parent \sqsubseteq \exists Enfant \sqcap Personne$
$Père \sqsubseteq Parent \sqcap Homme$
$Mère \sqsubseteq Parent \sqcap Femme$
$\exists Enfant \sqcap Personne \sqsubseteq Parent$
$Parent \sqcap Homme \sqsubseteq Père$
$Parent \sqcap Femme \sqsubseteq Mère$

Eve:Mère
Adam:Pere

**Pour simplifier, on peut se restreindre à Une T-Box définitoire (non cyclique)**

C N R S

LIP6 CNRS

# Tbox définitoire – acyclique

- Une **inclusion générale de concepts** est de la forme $C \sqsubseteq D$ où C et D sont des concepts.
- Une interprétation $\mathcal{I}$ est un modèle de si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- $\mathcal{I}$ est un modèle de la Tbox $\mathcal{T}$ si c'est un modèle de toutes les inclusions de concepts de $\mathcal{T}$.
- $C \equiv D$ est une abréviation pour $C \sqsubseteq D$ et $D \sqsubseteq C$
- Un axiome de la forme $A \equiv C$ où A est un nom de concept est appelé une **définition** de A
- Une Tbox $\mathcal{T}$ est dite **définitoire** si elle ne contient que des définitions avec les restrictions additionnelles suivantes:
  - $\mathcal{T}$ contient au plus une définition pour chaque nom de concept
  - $\mathcal{T}$ est acyclique

# Démonstration avec la méthode des tableaux

Homme $\sqsubseteq$ Personne
Femme $\sqsubseteq$ Personne
Personne $\sqsubseteq \exists$Age
Parent $\sqsubseteq \exists$Enfant $\sqcap$ Personne
Père $\sqsubseteq$Parent $\sqcap$ Homme
Mère $\sqsubseteq$Parent $\sqcap$ Femme
$\exists$Enfant $\sqcap$ Personne $\sqsubseteq$Parent
Parent $\sqcap$ Homme $\sqsubseteq$Père
Parent $\sqcap$ Femme $\sqsubseteq$ Mère

Eve:Mère
Adam:Pere

Pour simplifier, on peut se restreindre à Une T-Box définitoire (non cyclique)

Homme $\sqsubseteq$ Personne
Femme $\sqsubseteq$ Personne
~~Personne $\sqsubseteq \exists$Age~~
Parent $\equiv \exists$Enfant $\sqcap$ Personne
Père $\equiv$ Parent $\sqcap$ Homme
Mère $\equiv$ Parent $\sqcap$ Femme
~~$\exists$Enfant $\sqcap$ Personne $\sqsubseteq$Parent~~
~~Parent $\sqcap$ Homme $\sqsubseteq$Père~~
~~Parent $\sqcap$ Femme $\sqsubseteq$ Mère~~

Eve:Mère
Adam:Pere

C
N
R
S

# Démonstration avec la méthode des tableaux

Homme $\sqsubseteq$ Personne
Femme $\sqsubseteq$ Personne
Personne $\sqsubseteq \exists$Age
Parent $\sqsubseteq \exists$Enfant $\sqcap$ Personne
Père $\sqsubseteq$Parent $\sqcap$ Homme
Mère $\sqsubseteq$Parent $\sqcap$ Femme
$\exists$Enfant $\sqcap$ Personne $\sqsubseteq$Parent
Parent $\sqcap$ Homme $\sqsubseteq$Père
Parent $\sqcap$ Femme $\sqsubseteq$ Mère

Eve:Mère
Adam:Pere

Pour simplifier, on peut se restreindre à
Une T-Box définitoire (non cyclique)
Les définitions sont des inclusions
On se limite aux femmes et aux mères

Femme $\sqsubseteq$ Personne
Parent $\sqsubseteq \exists$Enfant $\sqcap$ Personne
Mère $\sqsubseteq$Parent $\sqcap$ Femme
$\exists$Enfant $\sqcap$ Personne $\sqsubseteq$Parent
Parent $\sqcap$ Femme $\sqsubseteq$ Mère

Eve:Mère

$\neg$Femme $\sqcup$ Personne
Parent $\sqcup$ $\neg$ $\exists$Enfant $\sqcup$ $\neg$Personne
Mère $\sqcup$ $\neg$Parent $\sqcup$ $\neg$Femme
($\exists$Enfant $\sqcap$ Personne) $\sqcup$ $\neg$Parent
(Parent $\sqcap$ Femme) $\sqcup$ $\neg$Mère

Eve:Mère

$\neg$Femme $\sqcup$ Personne
Parent $\sqcup$ $\forall\neg$Enfant $\sqcup$ $\neg$Personne
Mère $\sqcup$ $\neg$Parent $\sqcup$ $\neg$Femme
$\exists$Enfant $\sqcup$ $\neg$Parent
Personne $\sqcup$ $\neg$Parent
Femme $\sqcup$ $\neg$Mère
Parent $\sqcup$ $\neg$Mère

Eve:Mère

# Démonstration

¬Femme ⊔ Personne
Parent ⊔ ∀¬Enfant ⊔ ¬Personne
Mère ⊔ ¬Parent ⊔ ¬Femme
∃Enfant ⊔ ¬Parent
Personne ⊔ ¬Parent
Femme ⊔ ¬Mère
Parent ⊔ ¬Mère

Eve:Mère

---

¬Femme ⊔ Personne
Parent ⊔ ∀¬Enfant ⊔ ¬Personne
Mère ⊔ ¬Parent ⊔ ¬Femme
∃Enfant ⊔ ¬Parent
Personne ⊔ ¬Parent
Femme ⊔ ¬Mère
Parent ⊔ ¬Mère

Eve:Mère
Femme

---

¬Femme ⊔ Personne
Parent ⊔ ∀¬Enfant ⊔ ¬Personne
Mère ⊔ ¬Parent ⊔ ¬Femme
∃Enfant ⊔ ¬Parent
Personne ⊔ ¬Parent
Femme ⊔ ¬Mère
Parent ⊔ ¬Mère

Eve:Mère
¬Mère

# Tableau de droite

¬Femme ⊔ Personne
Parent ⊔ ∀¬Enfant ⊔ ¬Personne
Mère ⊔ ¬Parent ⊔ ¬Femme
∃Enfant ⊔ ¬Parent
Personne ⊔ ¬Parent
Femme ⊔ ¬Mère
Parent ⊔ ¬Mère
Eve:Mère
¬Mère

¬Femme ⊔ Personne
Parent ⊔ ∀¬Enfant ⊔ ¬Personne
Mère ⊔ ¬Parent ⊔ ¬Femme
∃Enfant ⊔ ¬Parent
Personne ⊔ ¬Parent
Femme ⊔ ¬Mère
Parent ⊔ ¬Mère
Eve:Mère
¬Mère
Eve:¬Mère

**Contradiction**

# Tableau de gauche

¬Femme ⊔ Personne
Parent ⊔ ∀¬Enfant ⊔ ¬Personne
Mère ⊔ ¬Parent ⊔ ¬Femme
∃Enfant ⊔ ¬Parent
Personne ⊔ ¬Parent
Femme ⊔ ¬Mère
Parent ⊔ ¬Mère

Eve:Mère
Femme

¬Femme ⊔ Personne
Parent ⊔ ∀¬Enfant ⊔ ¬Personne
Mère ⊔ ¬Parent ⊔ ¬Femme
∃Enfant ⊔ ¬Parent
Personne ⊔ ¬Parent
Femme ⊔ ¬Mère
Parent ⊔ ¬Mère

Eve:Mère
Femme
Eve:Femme

Les modèles comprendront tous Eve:Femme

# Méthode des tableaux pour $\mathcal{ALCN}$

- **$\geq$-règle:**
  - **Condition:** $\mathcal{A}$ contient $(\geq^n r)(x)$, mais il n'existe pas n constantes $z_1, \ldots z_n$ telle que pour tout i, $r(x,z_i)$ dans $\mathcal{A}$
  - **Action:** $\mathcal{A}' := \mathcal{A} \cup \{r(x, y_i) \mid 0<i<n+1\}$ les $y_i$ étant des constantes distinctes n'apparaissant pas déjà dans $\mathcal{A}$
- **$\leq$-règle:**
  - **Condition:** $\mathcal{A}$ contient les noms individuels distincts $y_1, y_2, \ldots, y_{n+1}$ tels que $(\leq^n r)(x)$ et $r(x,y_1), \ldots r(x,y_{n+1})$ sont dans $\mathcal{A}$ mais $y_i$ avec $y_i \neq y_j$ pour un i tel que $0<i<n+2$ n'est pas dans $\mathcal{A}$
  - **Action:** pour chaque paire $y_i$, $y_j$ telle que $0<i<j<n+2$, **et** $y_i \neq y_j$ la ABox $\mathcal{A}_{i,j}' := [y_i / y_j]\mathcal{A}$ est obtenue en remplaçant chaque occurrence de $y_i$ par $y_j$ dans $\mathcal{A}$

# Exemple: formalisation en $\mathcal{ALCN}$

a) Un alexandrin est un vers

b) Il existe deux formes distinctes d'alexandrin : le trimètre et le tétramètre

c) Un alexandrin comporte deux hémistiches et douze syllabes

d) Chaque hémistiche comporte 6 syllabes

e) Tous les alexandrins comportent une césure à la fin de l'hémistiche

f) "Je courus! Et les péninsules démarrées" est un vers écrit par Rimbaud qui a 12 syllabes

g) "Je courus ! Et les péninsules démarrées" ne comporte pas de césure à l'hémistiche

h) "Je courus ! Et les péninsules démarrées" n'est pas un alexandrin

# Formalisation en $\mathcal{ALCN}$
## Ajout négation conclusion

a) *Alexandrin* $\sqsubseteq$ *Vers*

b) *Alexandrin* $\sqsubseteq$ *Trimètre*
*Alexandrin* $\sqsubseteq$ *Tétramètre*
*Trimètre* $\sqcap$ *Tétramètre* $\sqsubseteq \perp$

c) *Alexandrin* $\sqsubseteq$ *Alexandrin* $\sqcap$
$\exists^{\geq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\leq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe*

d) *Hémistiche* $\sqsubseteq$ *Hémistiche* $\sqcap$
$\exists^{\geq 6}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 6}$ *contient.Syllabe*

e) *Alexandrin* $\sqsubseteq$ *Alexandrin* $\sqcap$
$\exists$*césure.Hémistiche*

f) *<« Je courus! Et les péninsules démarées », Rimbaud>: Auteur*
*« Je courus! Et les péninsules démarées »:Vers* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 12}$ *contient.Syllabe*

g) *« Je courus! Et les péninsules démarées »:*
$\neg\,\exists$*césure.Hémistiche*

***h) « Je courus! Et les péninsules démarées »: Alexandrin***

# Formalisation en $\mathcal{ALCN}$
## Ajout négation conclusion

a) $Alexandrin \sqsubseteq Vers$

b) $Alexandrin \sqsubseteq Trimètre$
$Alexandrin \sqsubseteq Tétramètre$
$Trimètre \sqcap Tétramètre \sqsubseteq \perp$

c) $Alexandrin \sqsubseteq Alexandrin \sqcap \exists^{\geq 2} contient.Hémistiche \sqcap \exists^{\leq 2} contient.Hémistiche \sqcap \exists^{\geq 12} contient.Syllabe \sqcap \exists^{\leq 12} contient.Syllabe$

d) $Hémistiche \sqsubseteq Hémistiche \sqcap \exists^{\geq 6} contient.Syllabe \sqcap \exists^{\leq 6} contient.Syllabe$

e) $\neg Alexandrin \sqcup (Alexandrin \sqcap \exists césure.Hémistiche)$

f) <« Je courus! Et les péninsules démarées », Rimbaud>: Auteur
« Je courus! Et les péninsules démarées »: $Vers \sqcap \exists^{\geq 12} contient.Syllabe \sqcap \exists^{\leq 12} contient.Syllabe$

g) « Je courus! Et les péninsules démarées »: $\neg \exists césure.Hémistiche$

**h) « Je courus! Et les péninsules démarées »: Alexandrin**

L I P 6 C N R S

# Application Règle-$\sqcup$ sur e)

a) *Alexandrin* $\sqsubseteq$ *Vers*

b) *Alexandrin* $\sqsubseteq$ *Trimètre*
*Alexandrin* $\sqsubseteq$ *Tétramètre*
*Trimètre* $\sqcap$ *Tétramètre* $\sqsubseteq \bot$

c) *Alexandrin* $\sqsubseteq$ *Alexandrin* $\sqcap$
$\exists^{\geq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\leq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe*

d) *Hémistiche* $\sqsubseteq$ *Hémistiche* $\sqcap$
$\exists^{\geq 6}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 6}$ *contient.Syllabe*

e) $\neg$***Alexandrin*** $\sqcup$ ***(Alexandrin*** $\sqcap$ $\exists$***césure.Hémistiche)***

f) *<« Je courus! Et les péninsules démarées », Rimbaud>: Auteur*
*« Je courus! Et les péninsules démarées »:Vers* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 12}$ *contient.Syllabe*

g) *« Je courus! Et les péninsules démarées »:*
$\neg\ \exists$*césure.Hémistiche*

h) *« Je courus! Et les péninsules démarées »: Alexandrin*

# Règle-$\sqcup$ tableau 1

a) *Alexandrin* $\sqsubseteq$ *Vers*

b) *Alexandrin* $\sqsubseteq$ *Trimètre*
*Alexandrin* $\sqsubseteq$ *Tétramètre*
*Trimètre* $\sqcap$ *Tétramètre* $\sqsubseteq \perp$

c) *Alexandrin* $\sqsubseteq$ *Alexandrin* $\sqcap$
$\exists^{\geq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\leq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe*

d) *Hémistiche* $\sqsubseteq$ *Hémistiche* $\sqcap$
$\exists^{\geq 6}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 6}$ *contient.Syllabe*

e) $\neg$*Alexandrin*

f) *<« Je courus! Et les péninsules démarées », Rimbaud>: Auteur*
*« Je courus! Et les péninsules démarées »:Vers* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 12}$ *contient.Syllabe*

g) *« Je courus! Et les péninsules démarées »:*
$\neg \exists$*césure.Hémistiche*

h) *« Je courus! Et les péninsules démarées »: Alexandrin*

**Clash entre e et h**

L
I
P
6

C
N
R
S

# Règle-$\sqcup$ - tableau 2

a) *Alexandrin* $\sqsubseteq$ *Vers*

b) *Alexandrin* $\sqsubseteq$ *Trimètre*
*Alexandrin* $\sqsubseteq$ *Tétramètre*
*Trimètre* $\sqcap$ *Tétramètre* $\sqsubseteq \bot$

c) *Alexandrin* $\sqsubseteq$ *Alexandrin* $\sqcap$
$\exists^{\geq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\leq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe*

d) *Hémistiche* $\sqsubseteq$ *Hémistiche* $\sqcap$
$\exists^{\geq 6}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 6}$ *contient.Syllabe*

e) *Alexandrin* $\sqcap$
$\exists$*césure.Hémistiche)*

f) *<« Je courus! Et les péninsules démarées », Rimbaud>: Auteur*

g) *« Je courus! Et les péninsules démarées »:Vers* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 12}$ *contient.Syllabe*

h) *« Je courus! Et les péninsules démarées »:*
$\neg\, \exists$*césure.Hémistiche*

i) *« Je courus! Et les péninsules démarées »: Alexandrin*

# Règle-$\sqcup$ - tableau 2 Application règle sur e)

a) $Alexandrin \sqsubseteq Vers$

b) $Alexandrin \sqsubseteq Trimètre$
$Alexandrin \sqsubseteq Tétramètre$
$Trimètre \sqcap Tétramètre \sqsubseteq \bot$

c) $Alexandrin \sqsubseteq Alexandrin \sqcap \exists^{\geq 2} contient.Hémistiche \sqcap \exists^{\leq 2} contient.Hémistiche \sqcap \exists^{\geq 12} contient.Syllabe$

d) $Hémistiche \sqsubseteq Hémistiche \sqcap \exists^{\geq 6} contient.Syllabe \sqcap \exists^{\leq 6} contient.Syllabe$

e) $Alexandrin \sqcap \exists césure.Hémistiche)$

f) <« Je courus! Et les péninsules démarées », Rimbaud>: Auteur

g) « Je courus! Et les péninsules démarées »:$Vers \sqcap \exists^{\geq 12} contient.Syllabe \sqcap \exists^{\leq 12} contient.Syllabe$

h) « Je courus! Et les péninsules démarées »: $\neg \exists césure.Hémistiche$

i) « Je courus! Et les péninsules démarées »: Alexandrin

# Règle-$\sqcup$ - tableau 2 Application règle sur e)

a) *Alexandrin* $\sqsubseteq$ *Vers*

b) *Alexandrin* $\sqsubseteq$ *Trimètre*
*Alexandrin* $\sqsubseteq$ *Tétramètre*
*Trimètre* $\sqcap$ *Tétramètre* $\sqsubseteq \bot$

c) *Alexandrin* $\sqsubseteq$ *Alexandrin* $\sqcap$
$\exists^{\geq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\leq 2}$ *contient.Hémistiche* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe*

d) *Hémistiche* $\sqsubseteq$ *Hémistiche* $\sqcap$
$\exists^{\geq 6}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 6}$ *contient.Syllabe*

e) *Alexandrin*

f) $\exists$*césure.Hémistiche*

g) *<« Je courus! Et les péninsules démarées », Rimbaud>: Auteur*

h) *« Je courus! Et les péninsules démarées »:Vers* $\sqcap$
$\exists^{\geq 12}$ *contient.Syllabe* $\sqcap$
$\exists^{\leq 12}$ *contient.Syllabe*

i) *« Je courus! Et les péninsules démarées »:*
$\neg\exists$*césure.Hémistiche*

j) *« Je courus! Et les péninsules démarées »: Alexandrin*

**Clash entre f) et i)**

# Tbox définitoire – acyclique

- Une **inclusion générale de concepts** est de la forme $C \sqsubseteq D$ où C et D sont des concepts.
- Une interprétation $\mathcal{I}$ est un modèle de si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- $\mathcal{I}$ est un modèle de la Tbox $\mathcal{T}$ si c'est un modèle de toutes les inclusions de concepts de $\mathcal{T}$.
- $C \equiv D$ est une abréviation pour $C \sqsubseteq D$ et $D \sqsubseteq C$
- Un axiome de la forme $A \equiv C$ où A est un nom de concept est appelé une **définition** de A
- Une Tbox $\mathcal{T}$ est dite **définitoire** si elle ne contient que des définitions avec les restrictions additionnelles suivantes:
  - $\mathcal{T}$ contient au plus une définition pour chaque nom de concept
  - $\mathcal{T}$ est acyclique

# Autre exemple: Tbox « définitoire »

- **Tbox**

$\mathrm{Woman} \equiv \mathrm{Person} \sqcap \mathrm{Female}$

$\mathrm{Man} \equiv \mathrm{Person} \sqcap \neg\mathrm{Woman}$

$\mathrm{Mother} \equiv \mathrm{Woman} \sqcap \exists \mathrm{hasChild.Person}$

$\mathrm{Father} \equiv \mathrm{Man} \sqcap \exists \mathrm{hasChild.Person}$

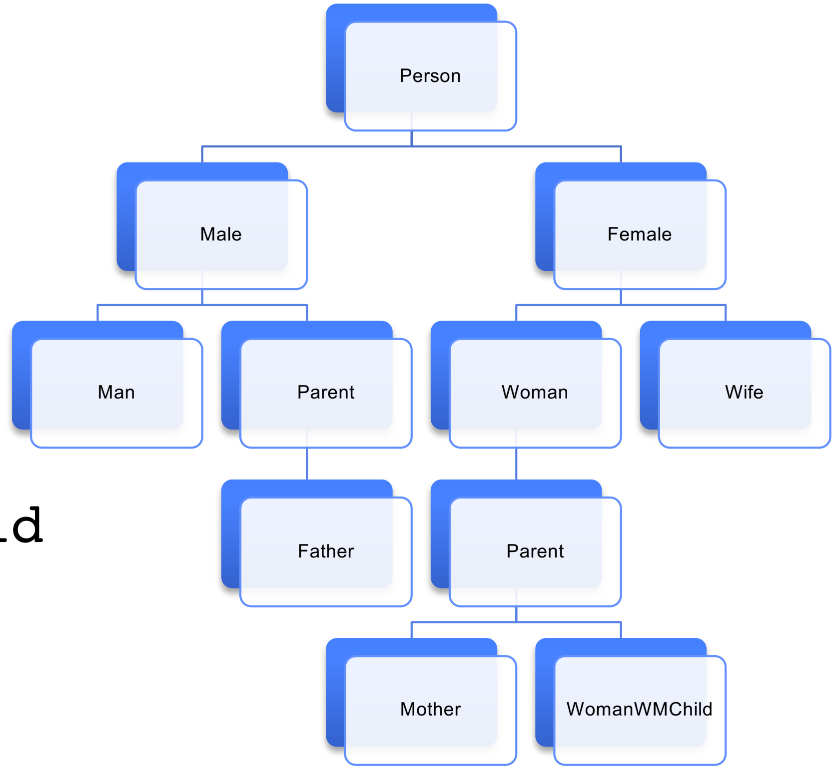
$\mathrm{Parent} \equiv \mathrm{Father} \sqcup \mathrm{Mother}$

$\mathrm{GrandMother} \equiv \mathrm{Mother} \sqcap \exists \mathrm{hasChild.Parent}$

$\mathrm{MotherWMChild} \equiv \mathrm{Mother} \sqcap \geq 3\ \mathrm{hasChild}$

$\mathrm{WomanWDaughter} \equiv \mathrm{Mother} \sqcap \forall \mathrm{hasChild.}\neg\mathrm{Woman}$

$\mathrm{Wife} \equiv \mathrm{Woman} \sqcap \exists \mathrm{hasHusband.Man}$



# Tbox définitoire

- Une **inclusion générale de concepts** est de la forme C $\sqsubseteq$ D où C et D sont des concepts.
- Une interprétation $\mathcal{I}$ est un modèle de si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- $\mathcal{I}$ est un modèle de la Tbox $\mathcal{T}$ si c'est un modèle de toutes les inclusions de concepts de $\mathcal{T}$.
- C $\equiv$ D est une abréviation pour C $\sqsubseteq$ D et D $\sqsubseteq$ C
- Un axiome de la forme A $\equiv$ C où A est un nom de concept est appelé une **définition** de A
- Une Tbox $\mathcal{T}$ est dite **définitoire** si elle ne contient que des définitions avec les restrictions additionnelles suivantes:
  - $\mathcal{T}$ contient au plus une définition pour chaque nom de concept
  - $\mathcal{T}$ est acyclique

# Tbox non définitoire
# Cycle

Homme ⊑ Personne
Femme ⊑ Personne
Personne ⊑∃Age
Parent ⊑ ∃Enfant ⊓ Personne
Père ⊑Parent ⊓ Homme
Mère ⊑Parent ⊓ Femme
∃Enfant ⊓ Personne ⊑Parent
Parent ⊓ Homme ⊑Père
Parent ⊓ Femme ⊑ Mère

Eve:Mère
Adam:Pere

Femme ⊑ Personne
Parent ⊑ ∃Enfant ⊓ Personne
Mère ⊑Parent ⊓ Femme
∃Enfant ⊓ Personne ⊑Parent
Parent ⊓ Femme ⊑ Mère

Eve:Mère

¬Femme ⊔ Personne
Parent ⊔ ¬ ∃Enfant ⊔ ¬Personne
Mère ⊔ ¬Parent ⊔ ¬Femme
(∃Enfant ⊓ Personne) ⊔ ¬Parent
(Parent ⊓ Femme) ⊔ ¬Mère

Eve:Mère

¬Femme ⊔ Personne
Parent ⊔ ∀¬Enfant ⊔ ¬Personne
Mère ⊔ ¬Parent ⊔ ¬Femme
∃Enfant ⊔ ¬Parent
Personne ⊔ ¬Parent
Femme ⊔ ¬Mère
Parent ⊔ ¬Mère

Eve:Mère

C N R S

Homme $\sqsubseteq$ Personne
Femme $\sqsubseteq$ Personne
Personne $\sqsubseteq \exists$Age
Parent $\equiv \exists$Enfant $\sqcap$ Personne
Père $\equiv$Parent $\sqcap$ Homme
Mère $\equiv$Parent $\sqcap$ Femme

Eve:Mère
Adam:Pere

# Tbox Définitoire Absence de cycles

Femme $\sqsubseteq$ Personne
Homme $\sqsubseteq$ Personne
Parent $\equiv \exists$Enfant $\sqcap$ Personne
Mère $\equiv$Parent $\sqcap$ Femme
Père $\equiv$Parent $\sqcap$ Homme

Eve:Mère
Adam:Père
¬Eve:Femme

Démonstration: réécriture
Eve:Mère se réécrit en
Eve:Parent$\sqcap$ Femme puis en
Eve: $\exists$Enfant $\sqcap$ Personne $\sqcap$ Femme

6 C N R S

# Autre exemple: Tbox « définitoire » Réécriture de la Tbox

- **Tbox**

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬Woman

Mother ≡ Woman ⊓ ∃hasChild.Person

Father ≡ Man ⊓ ∃hasChild.Person

Parent ≡ Father ⊔ Mother

GrandMother ≡ Mother ⊓ ∃hasChild.Parent

MotherWMChild≡ Mother ⊓ ≥3 hasChild

WomanWDaughter ≡ Mother ⊓ ∀hasChild.¬Woman

Wife ≡ Woman ⊓ ∃hasHusband.Man

- **Abox**

MotherWDaughter(MARY)

Father(PETER)

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

GrandMother(MARY) ?

# Autre exemple: Tbox « définitoire »
# Réécriture de la Tbox

- **Tbox**

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬Woman

Mother ≡ Woman ⊓ ∃hasChild.Person

Father ≡ Man ⊓ ∃hasChild.Person

Parent ≡ Father ⊔ Mother

GrandMother ≡ Mother ⊓ ∃hasChild.Parent

MotherWMChild≡ Mother ⊓ ≥3 hasChild

WomanWDaughter ≡ Mother ⊓ ∀hasChild.¬Woman

Wife ≡ Woman ⊓ ∃hasHusband.Man

- **Abox**

MotherWDaughter(MARY)

Father(PETER)

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

GrandMother(MARY) ?

# Autre exemple: Tbox « définitoire »
# Réécriture de la Tbox

- **Tbox**

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬Woman

Mother ≡ Woman ⊓ ∃hasChild.Person

Father ≡ Man ⊓ ∃hasChild.Person

Parent ≡ Father ⊔ Mother

GrandMother ≡ Mother ⊓ ∃hasChild.Parent

MotherWMChild≡ Mother ⊓ ≥3 hasChild

WomanWDaughter ≡ Mother ⊓ ∀hasChild.¬Woman

Wife ≡ Woman ⊓ ∃hasHusband.Man

- **Abox**

MotherWDaughter(MARY)

Father(PETER)

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

GrandMother(MARY) ?

# Autre exemple: Tbox « définitoire »
# Réécriture de la Tbox

- **Tbox**

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬Woman

Mother ≡ Woman ⊓ ∃hasChild.Person

Father ≡ Man ⊓ ∃hasChild.Person

Parent ≡ Father ⊔ Mother

GrandMother ≡ Mother ⊓ ∃hasChild.Parent

MotherWMChild≡ Mother ⊓ ≥3 hasChild

WomanWDaughter ≡ Mother ⊓ ∀hasChild.¬Woman

Wife ≡ Woman ⊓ ∃hasHusband.Man

- **Abox**

MotherWDaughter(MARY)

Father(PETER)

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

GrandMother(MARY) ?

# Autre exemple: Tbox « définitoire »
# Réécriture de la Tbox

L I P 6 C N R S

- **Tbox**

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬Woman

Mother ≡ Woman ⊓ ∃hasChild.Person

Father ≡ Man ⊓ ∃hasChild.Person

Parent ≡ Father ⊔ Mother

GrandMother ≡ Mother ⊓ ∃hasChild.Parent

MotherWMChild≡ Mother ⊓ ≥3 hasChild

WomanWDaughter ≡ Mother ⊓ ∀hasChild.¬Woman

Wife ≡ Woman ⊓ ∃hasHusband.Man

- **Abox**

MotherWDaughter(MARY)

Father(PETER)

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

GrandMother(MARY) ?

# Autre exemple: Tbox « définitoire »
# Réécriture de la Tbox

- **Tbox**

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬Woman

Mother ≡ Woman ⊓ ∃hasChild.Person

Father ≡ Man ⊓ ∃hasChild.Person

Parent ≡ Father ⊔ Mother

GrandMother ≡ Mother ⊓ ∃hasChild.Parent

MotherWMChild≡ Mother ⊓ ≥3 hasChild

WomanWDaughter ≡ Mother ⊓ ∀hasChild.¬Woman

Wife ≡ Woman ⊓ ∃hasHusband.Man

- **Abox**

MotherWDaughter(MARY)

Father(PETER)

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

GrandMother(MARY) ?

# Remplacement définitions de la Tbox *(en absence de définitions circulaires)*

- **Tbox**

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬(Person ⊓ Female)

Mother ≡ (Person ⊓ Female) ⊓ ∃hasChild.Person

Father ≡ (Person ⊓ ¬(Person ⊓ Female)⊓∃hasChild.Person

Parent ≡ ((Person ⊓ ¬(Person ⊓ Female) ⊓ ∃hasChild.Person)
⊔ ((Person ⊓ Female) ⊓ ∃hasChild.Person)

GrandMother ≡ (Person ⊓ Female) ⊓ ∃hasChild.Person ⊓
∃hasChild.(Person ⊓ ∃hasChild.Person)

MotherWMChild≡ (Person ⊓ Female) ⊓ ∃hasChild.Person ⊓
≥3 hasChild

WomanWDaughter ≡ (Person ⊓ Female) ⊓ ∃hasChild.Person ⊓
∀hasChild.¬(Person ⊓ Female)

Wife ≡ (Person ⊓ Female) ⊓ ∃hasHusband.(Person ⊓ ¬(Person
⊓ Female))

# Transformation en NNF

- **Tbox**

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬Female

Mother ≡ Person ⊓ Female ⊓ ∃hasChild.Person

Father ≡ Person ⊓ ¬Female ⊓ ∃hasChild.Person

Parent ≡ Person ⊓ ∃hasChild.Person

GrandMother ≡ (Person ⊓ Female) ⊓ ∃hasChild.Person ⊓
∃hasChild.(Person ⊓ ∃hasChild.Person)

MotherWMChild≡ Person ⊓ Female ⊓ ≥3 hasChild

WomanWDaughter ≡ (Person ⊓ Female) ⊓
∀hasChild.(¬Person ⊔ ¬Female)

Wife ≡ Person ⊓ Female ⊓ ∃hasHusband.(Person ⊓ ¬Female)

# Réécriture de la Abox avec la Tbox

- **Abox**

¬GrandMother(MARY)

¬Person(MARY) ⊔ ¬Female(MARY) ⊔
¬∃hasChild.Person(MARY) ⊔
¬∃hasChild.(Person ⊓ ∃hasChild.Person)(MARY)

- **Tbox**

Woman ≡ Person ⊓ Female

Man ≡ Person ⊓ ¬Female

Mother ≡ Person ⊓ Female ⊓ ∃hasChild.Person

Father ≡ Person ⊓ ¬Female ⊓ ∃hasChild.Person

Parent ≡ Person ⊓ ∃hasChild.Person

GrandMother ≡ Person ⊓ Female ⊓ ∃hasChild.Person
⊓ ∃hasChild.(Person ⊓ ∃hasChild.Person)

MotherWMChild≡ Person ⊓ Female ⊓ ≥3 hasChild

WomanWDaughter ≡ (Person ⊓ Female) ⊓ ∀hasChild.(¬Person ⊔ ¬Female)

Wife ≡ Person ⊓ Female ⊓ ∃hasHusband.(Person ⊓ ¬Female)

- **Abox**

MotherWDaughter(MARY)

Father(PETER)

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Person(HARRY)

- **Réécritures Abox**

Person(MARY)
Female(MARY)
∀hasChild.(¬Person ⊔ ¬Female)(MARY)

Person(PETER)
¬Female(PETER)
∃hasChild.Person(PETER)

L I P 6 C N R S

# Réécriture de la Abox avec la Tbox

- **Abox**

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

¬Person(MARY) ⊔ ¬Female(MARY) ⊔
¬∃hasChild.Person(MARY) ⊔
¬∃hasChild.(Person ⊓ ∃hasChild.Person)(MARY)

Person(MARY)
Female(MARY)
∀hasChild.(¬Person ⊔ ¬Female)(MARY)

Person(PETER)
¬Female(PETER)
∃hasChild.Person(PETER)

- **Abox: terme réécrits**

MotherWDaughter(MARY)

Father(PETER)

¬GrandMother(MARY)

# Tableaux 1

- **Abox**

```
hasChild(MARY, PETER)
hasChild(MARY, PAUL)
hasChild(PETER, HARRY)
Person(PAUL)
Person(HARRY)
¬Person(MARY) ⊔ ¬Female(MARY) ⊔ ¬∃hasChild.Person(MARY) ⊔
   ¬∃hasChild.(Person ⊓ ∃hasChild.Person)(MARY)
Person(MARY)
   Female(MARY)
   ∀hasChild.(¬Person ⊔ ¬Female)(MARY)
Person(PETER)
   ¬Female(PETER)
   ∃hasChild.Person(PETER)
```

$R_{\sqcup}$

Clash

```
…
¬Person(MARY)
Person(MARY)
…
```

2

# Tableau 2

- **Abox**

```
hasChild(MARY, PETER)
hasChild(MARY, PAUL)
hasChild(PETER, HARRY)
Person(PAUL)
Person(HARRY)
¬Female(MARY) ⊔ ¬∃hasChild.Person(MARY) ⊔
   ¬∃hasChild.(Person ⊓ ∃hasChild.Person)(MARY)
Person(MARY)
   Female(MARY)
   ∀hasChild.(¬Person ⊔ ¬Female)(MARY)
Person(PETER)
   ¬Female(PETER)
   ∃hasChild.Person(PETER)
```

$R_{\sqcup}$

Clash

```
…
¬Female(MARY)
Female(MARY)
…
```

3

# Tableau 3

- **Abox**

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

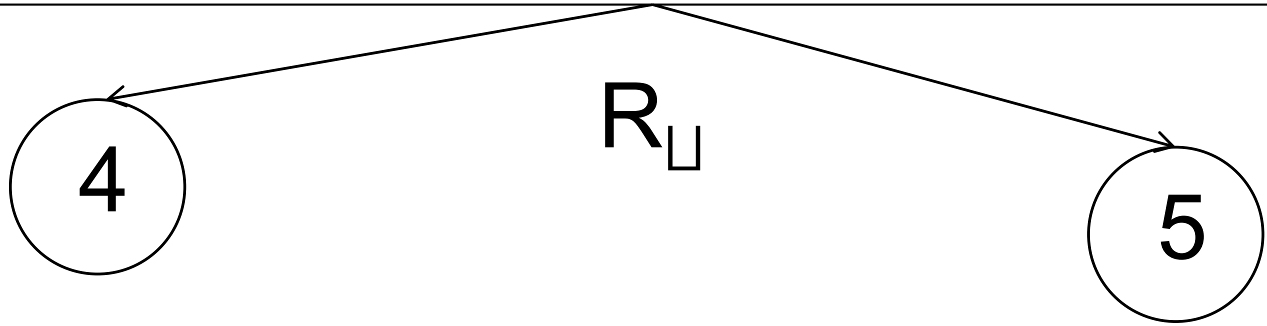
Person(PAUL)

Person(HARRY)

¬∃hasChild.Person(MARY) ⊔
  ¬∃hasChild.(Person ⊓ ∃hasChild.Person)(MARY)

Person(MARY)
  Female(MARY)
  ∀hasChild.(¬Person ⊔ ¬Female)(MARY)

Person(PETER)
  ¬Female(PETER)
  ∃hasChild.Person(PETER)



# Tableau 4

- **Abox**

```
hasChild(MARY, PETER)
hasChild(MARY, PAUL)
hasChild(PETER, HARRY)
Person(PAUL)
Person(HARRY)
¬∃hasChild.Person(MARY)
    → ∀hasChild.¬Person(MARY)
Person(MARY)
   Female(MARY)
   ∀hasChild.(¬Person ⊔ ¬Female)(MARY)
Person(PETER)
   ¬Female(PETER)
   ∃hasChild.Person(PETER)
```

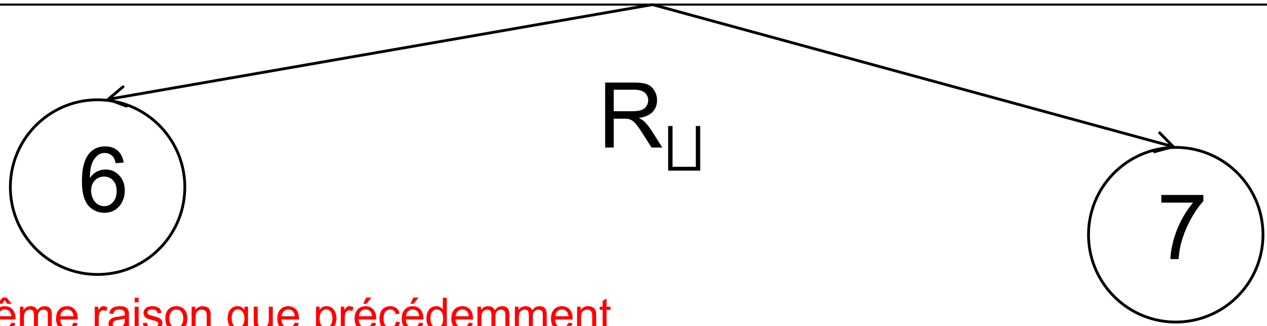
## Clash

En utilisant la règle-∀ sur **∀hasChild.¬Person(MARY)** si **hasChild(MARY, y)** on doit ajouter **¬Person(y)**. Or, comme **hasChild(Mary, PAUL)** on doit ajouter **¬Person(PAUL)**, ce qui produit un clash avec **Person(PAUL)**

# Tableau 5

- **Abox**

```
hasChild(MARY, PETER)
hasChild(MARY, PAUL)
hasChild(PETER, HARRY)
Person(PAUL)
Person(HARRY)
¬∃hasChild.(Person ⊓ ∃hasChild.Person)(MARY)
∀hasChild.¬Person(MARY) ⊔ ∀hasChild.∀hasChild.¬Person(MARY)
Person(MARY)
   Female(MARY)
   ∀hasChild.(¬Person ⊔ ¬Female)(MARY)
Person(PETER)
   ¬Female(PETER)
   ∃hasChild.Person(PETER)
```




Clash – même raison que précédemment

# Tableau 7

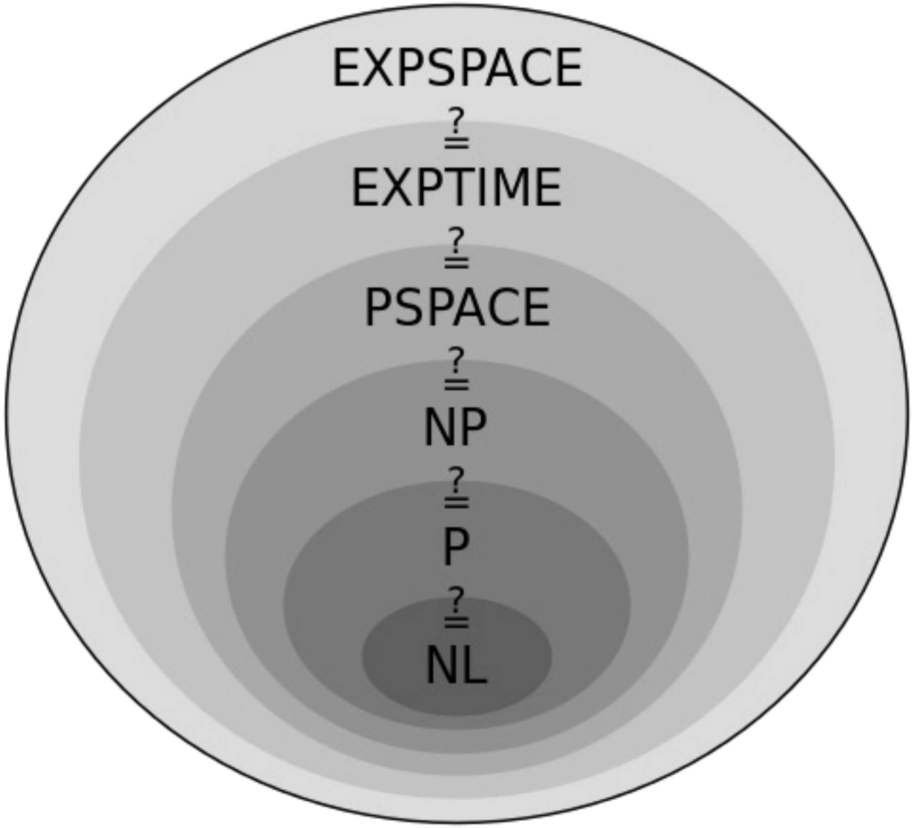
- **Abox**

```
hasChild(MARY, PETER)
hasChild(MARY, PAUL)
hasChild(PETER, HARRY)
Person(PAUL)
Person(HARRY)
∀hasChild.∀hasChild.¬Person(MARY)
∀hasChild.¬Person(PETER)
¬Person(HARRY)
Person(MARY)
   Female(MARY)
   ∀hasChild.(¬Person ⊔ ¬Female)(MARY)
Person(PETER)
   ¬Female(PETER)
   ∃hasChild.Person(PETER)
```

Clash

On applique toujours la règle-∀

# Rappels sur la complexité




- **P:** temps polynomial, machine déterministe
- **NP:** temps polynomial, machine non déterministe
- **EXPTIME:** temps exponentiel, machine déterministe
- **NEXPTIME:** temps exponentiel, machine non déterministe
- **L:** espace logarithmique, machine déterministe
- **NL:** espace logarithmique, machine non déterministe
- **PSPACE:** espace polynomial, machine déterministe (= **NPSPACE**)
- **EXSPACE**: espace exponentiel

# Propriétés logiques et algorithmiques

- Théorème: il est décidable de savoir si un concept de $\mathcal{ALC}$ est satisfiable
- Théorème: la satisfiabilité et la consistance d'une description de concept d'$\mathcal{ALC}$ est décidable avec une complexité spatiale polynomiale.
- Théorème: la consistance d'une $\mathcal{ALC}$-Abox est complète avec une complexité spatiale polynomiale
- Théorème: la satisfiabilité dans $\mathcal{ALCN}$ est complète avec une complexité spatiale polynomiale
- Lorsqu'il y a une Tbox, l'algorithme est en temps exponentiel
- Les restrictions sur les nombres ne posent pas de problèmes
- Les rôles transitifs sont plus problématiques… Interaction avec hiérarchies de rôles…

# Propriétés logiques et algorithmiques (suite)

- **Les rôles inverses et nominaux ne sont pas très problématiques**
  - **satisfiabilité de concepts dans $\mathcal{ALCQO}$ et $\mathcal{ALCI}$ avec des rôles transitifs est encore polynomiale en espace**
  - **en revanche, la satisfiabilité de concepts de $\mathcal{ALCIO}$ est exponentielle en temps (ExpTime)**
- **Enfin, les rôles nominaux, inverses et les restrictions de nombres ont ensemble des effets catastrophiques…**
  - **La satisfiabilité de $\mathcal{ALCQIO}$ est NExpTime, même si la satisfiabilité de concepts dans $\mathcal{ALCQI}$ , $\mathcal{ALCIO}$ et $\mathcal{ALCOQ}$ eu égard à une Tbox est ExpTime**

# Subsomption structurelle

- Langage: $\mathcal{FL}_0$
  - Conjonction de concepts   C ⊓ D
  - Restriction $\forall$r.C
- Mise sous forme normale dans $\mathcal{FL}_0$

$$C \equiv A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r_1.C_1 \sqcap \ldots \sqcap \forall r_n.C_n$$

# $\mathcal{FL}_0$: la plus simple logique de description

## Syntaxe

**Alphabet**

- **concepts atomiques A, B, C, D…**
- **Rôles atomiques r, s, u, v,**
- **Symboles {$\sqcap$, $\forall$, .}**

**Grammaire**

concept ::= <concept atomique> |
<concept> $\sqcap$ <concept> |
$\forall$<role atomic>.<concept>

LIP6 CNRS

# Algorithme de subsomption structurelle dans $\mathcal{FL}_0$

**$C \sqsubseteq D$**

**1. Normalisation**

$$C \equiv A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r_1.C_1 \sqcap \ldots \sqcap \forall r_n.C_n$$

$$D \equiv B_1 \sqcap \ldots \sqcap B_k \sqcap \forall s_1.D_1 \sqcap \ldots \sqcap \forall s_l.D_l$$

**2. Vérifier récursivement:**

$\forall i \leq k\ \exists j \leq m$ tel que $A_j \sqsubseteq B_i$

$\forall i \leq l\ \exists j \leq n$ tel que $s_i = r_j$ , $C_j \sqsubseteq D_i$

# $\mathcal{FL}_0$: la plus simple logique de description

## Exemple

Homme $\sqsubseteq$ Personne

Femme $\sqsubseteq$ Personne

Enfant $\equiv$ Personne $\sqcap \forall$aParent.Personne

PetitFils $\equiv$ Homme$\sqcap\forall$aParent.Enfant

PetiteFille $\equiv$ Femme$\sqcap\forall$aParent.Enfant

**Grammaire**

concept ::= <concept atomique> |
<concept> $\sqcap$ <concept> |
$\forall$<role atomic>.<concept>

# Démonstration par Subsomption Structurelle

Homme ⊑ Personne

Femme ⊑ Personne

Enfant ≡ Personne ⊓∀aParent.Personne

PetitFils ≡ Homme⊓∀aParent.Enfant

PetiteFille ≡ Femme⊓∀aParent.Enfant

$C \sqsubseteq D$ $\quad C \equiv A_1 \sqcap \ldots \sqcap A_m \sqcap \forall r_1.C_1 \sqcap \ldots \sqcap \forall r_n.C_n$

$D \equiv B_1 \sqcap \ldots \sqcap B_k \sqcap \forall s_1.D_1 \sqcap \ldots \sqcap \forall s_l.D_l$

$\forall i \leq k\ \exists j \leq m$ tel que $A_j \sqsubseteq B_i$

$\forall i \leq l\ \exists j \leq n$ tel que $s_i = r_j$ , $C_j \sqsubseteq D_i$

Démontrer par subsomption structurelle que PetitFils ⊑ Enfant

PetitFils ≡ Homme⊓∀aParent.Enfant et

Enfant ≡ Personne ⊓∀aParent.Personne

1. $\forall i \leq k\ \exists j \leq m$ tel que $A_j \sqsubseteq B_i$

Le seul concept de Enfant est Personne or ∃Homme ∊ PetitFils tels que Homme ⊑ Personne

2. $\forall i \leq l\ \exists j \leq n$ tel que $s_i = r_j$ , $C_j \sqsubseteq D_i$

Le seul rôle est aParent; le concept qui qualifie aParent dans Enfant est Personne

Le concept qui qualifie aParent dans PetitsFils est Enfant

Or Enfant ≡ Personne ⊓∀aParent.Personne ⊑Personne (car A ⊓ B ⊑ A)

# 2nd Algorithme de subsomption structurelle dans $\mathcal{FL}_0$

**C ⊑ D**

En utilisant la règle de réécriture $\forall r.(C \sqcap D) = \forall r.C \sqcap \forall r.D$ avec l'associativité et la commutativité, tout concept de $\mathcal{FL}_0$ peut se mettre sous la forme d'une conjonction de $\forall r_1 \ldots \forall r_m.A$ avec $m \geq 0$, $r_1, \ldots, r_m$ étant des noms de rôles et A un nom de concept atomique.

Le terme $\forall \emptyset.A$ correspondant au concept T (true), tout couple de concepts C et D contenant les concepts $A_1, \ldots A_k$ peut se mettre sous la forme:

$C \equiv \forall u_1.A_1 \sqcap \ldots \sqcap \forall u_k.A_k$ et $D \equiv \forall v_1.A_1 \sqcap \ldots \sqcap \forall v_k.A_k$

où les $u_i$ et les $v_i$ sont des suites finies (éventuellement vides) de mots sur l'alphabet des noms de rôles.

# $2^{nd}$ Algorithme de subsomption structurelle dans $\mathcal{FL}_0$ (suite)



Les concepts C et D étant mis sous cette forme normale:

$C \equiv \forall u_1.A_1 \sqcap \ldots \sqcap \forall u_k.A_k$ et $D \equiv \forall v_1.A_1 \sqcap \ldots \sqcap \forall v_k.A_k$

où les $u_i$ et les $v_i$ sont des suites finies de mots sur l'alphabet des noms de rôles.

**C $\sqsubseteq$ D si et seulement si** $u_i \supseteq v_i$ pour tout i, $1 \leq i \leq k$

Comme la taille de ces formes normales est polynomiale et que les tests d'inclusions $u_i \supseteq v_i$ sont polynomiaux, la subsomption peut être testée en temps polynomial dans $\mathcal{FL}_0$

**Possibilité d'extension à des Tbox**

**Complexité:** coNP-complet avec des Tbox définitoires et

ExpTime-complet avec des Tbox générales

L I P 6 C N R S

# $\mathcal{EL}$: logique de description minimale - existentielle

## Syntaxe

**Alphabet**

- **concepts atomiques A, B, C, D…**
- **Rôles atomiques r, s, u, v, …**
- **Symboles {$\sqcap$, $\exists$, .}**

**Grammaire**

concept ::= <concept atomique> |
<concept> $\sqcap$ <concept> |
$\exists$<role atomic>.<concept>

# Algorithme de subsomption structurelle dans $\mathcal{EL}$

- **La complexité de la subsomption reste polynomiale dans $\mathcal{EL}$, même en présence de Tbox non definitoire**

- **Quatre étapes:**
  1. **Normaliser la Tbox**
  2. **Traduire la Tbox normalisée dans un graphe**
  3. **Compléter le graphe avec des règles de complétion**
  4. **Eliminer les relations de subsomption du graphe normalisé**

# Extension de $\mathcal{FL}_0$

- Langage: $\mathcal{FL}_0$
  - Conjonction  C ⊓ D
  - Restriction  ∀r.C
- Langage: $\mathcal{EL}$
  - Conjonction  C ⊓ D
  - Restriction  ∃r.C
- Langage: $\mathcal{ALN}$
  - $\mathcal{AL}$ (C ⊓ D, ∀r.C , T, ⊥, ¬A, ∃r.T)
  - Restrictions sur cardinalités ($\geq^n$r, $\leq^n$r)

**Subsomption structurelle**

- Langage: $\mathcal{ALCN}$
  - $\mathcal{ALN}$ (C ⊓ D, ∀r.C , T, ⊥, ¬A, ∃r.T, $\geq^n$r, $\leq^n$r)
  - Négation sans restriction

**Tableaux**

# Méthode des tableaux

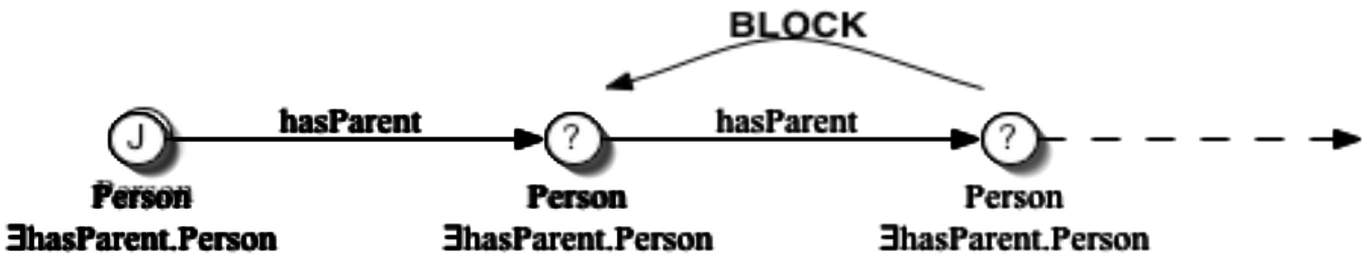
- Méthode clef de raisonnement qui se ramène à l'(in)satisfiabilité
  - Par exemple $C \sqsubseteq D$ dans la base de connaissances $\mathcal{K}$ ssi $\mathcal{K} \cup \{x:(C \sqcap \neg D)\}$ n'est pas satisfiable
- Etat de l'art des systèmes de logique de description (hautement optimisés) utilisant des méthodes de tableaux pour décider de la satisfiabilité (cohérence) d'une base de connaissances
- Les algorithmes implantant la méthode des tableaux travaillent en essayant de construire des exemples concrets (modèles) cohérents avec les axiomes:
  - On part des exemples (ABox)
  - Explication de la structure impliquée par des concepts complexes et par les axiomes terminologiques (TBox)
    - » Décomposition syntaxique utilisant les règles d'expansion des tableaux
    - » Inférence des contraintes sur les (éléments des) modèles

# Résumé sur la méthode des tableaux

- Les règles de tableaux rules correspondent aux constructeurs ($\sqcap$, $\exists$ etc)
  - exemple John:(Person $\sqcap$ Doctor) $\dashrightarrow$ John:Person and John:Doctor
- Arrêt quand il n'y a plus de règle application ou qu'un clash intervient
  - Un Clash est une contradiction, A(x), $\neg$A(x)
- Quelques règles non déterministes (par exemple, $\sqcup$)
  - En pratique, cela signifie une recherche
- Vérification de cycle (blocage) souvent requis pour assurer la terminaison
  - exemple:

    {Person $\sqsubseteq$ $\exists$hasParent.Person,

    John:Person}



# Résumé

- Les logiques de description sont des formalismes logiques de représentation des connaissances
  - Il sont connus pour être au fondement des langages d'ontologies comme OWL
- Les motivations pour la conception d'OWL tiennent à l'existence de procédures de décision fondées sur la méthode des tableaux et à leur implémentation
  - Mais il n'y a pas de procédure/implémentation pour OWL DL/$\mathcal{SHOIN}$ (jusqu'à maintenant),
- Des algorithmes $\mathcal{SHOIQ}$ résolvent ce (très embarrassant) problème
  - Mais les règles introduisent une nouvelle forme de non déterminisme