

Logiques de description II

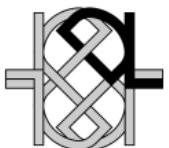
LREC – Cours 4

Jean-Gabriel Ganascia

Rappel sur syntaxe et sémantique

Démonstration par la méthode des tableaux

Démonstration par la subsomption structurelle



Références bibliographiques

- *Description Logics*, Franz Baader, Ian Horrocks, Ulrike Sattler, in « Handbook of Knowledge Representation », éditeurs Frank Van Harmelen, Vladimir Lifschitz, Bruce Porte, chapitre 3, pp. 135-179
- *The Description Logic Handbook: Theory, Implementation and Applications*, F. Baader, E. Franconi, B. Hollunder, B. Neble, H.-J. Profitlich, Cambridge University Press, 2003



Logiques terminologiques Logiques de description...

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

TKRS: *Terminological Knowledge Representation Systems*

Deux composants

- **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 ***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 ***ALC*** (équivalent à la logique multimodale $K_{(m)}$) – cela signifie ***AL*** et complémentation ***C***
 - Les concepts sont construits en utilisant \sqcap , \sqcup , \neg , \exists et \forall
- ***FL***- correspond à ***AL*** sans la négation atomique
- ***FL₀*** correspond à ***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>
```



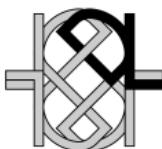
\mathcal{FL} - Syntaxe

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 { \sqcap , \exists , .}**

Grammaire

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



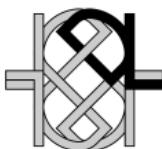
\mathcal{AL} Syntaxe

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

 $\Sigma = \langle \text{TBox}, \text{ABox} \rangle$

- **TBox: axiomes terminologiques $C \sqsubseteq D, C = D$**
 - Définitions
 - Parent = $\exists a \text{ENFANT} \sqcap \text{Personne}$
 - Subsomptions
 - Homme \sqsubseteq Personne (\sqsubseteq : *subsumption*)
- **ABox: assertions $a:C, \langle a, b \rangle:R$**
 - Assertions de concepts
 - Jean:Parent
 - Jean:Personne $\sqcap \exists a \text{ENFANT}$
 - Albert:personne
 - Assertions de rôles
 - $\langle \text{Jean, Thomas} \rangle: \text{aENFANT}$



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

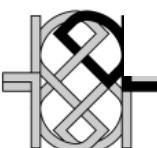
Alphabet

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

Grammaire

- \top et \perp 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 ($estEnfantDe^{-1}$ $aEnfant^{-1}$)
 - \mathcal{N} pour les restrictions sur les nombres (de la forme $\forall R, >nR \text{ ou } \exists^{\leq n}, \exists^{\geq n}$)
 - \mathcal{Q} pour les restrictions qualifiées sur les nombres (of form $\exists n R.C, >n R.C$)
- p.e. OWL est $\mathcal{ALC} + R_+ + \text{hierarchie de rôles} + \text{classes nominales} + \text{inversion de rôles} + \text{restrictions qualifiées sur les nombres} = SHOIQ$

\mathcal{FL} : Sémantique formelle

Une interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ 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}

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

Extensions \mathcal{ALC}

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

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



Sémantique générale

$(C \sqcap D)^{\mathcal{I}}$	$=$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$(C \sqcup D)^{\mathcal{I}}$	$=$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$(\neg C)^{\mathcal{I}}$	$=$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\{x\}^{\mathcal{I}}$	$=$	$\{x^{\mathcal{I}}\}$
$(\exists R.C)^{\mathcal{I}}$	$=$	$\{x \mid \exists y. \langle x, y \rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
$(\forall R.C)^{\mathcal{I}}$	$=$	$\{x \mid \forall y. (x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
$(\leq n R)^{\mathcal{I}}$	$=$	$\{x \mid \#\{y \mid \langle x, y \rangle \in R^{\mathcal{I}}\} \leq n\}$
$(\geq n R)^{\mathcal{I}}$	$=$	$\{x \mid \#\{y \mid \langle x, y \rangle \in R^{\mathcal{I}}\} \geq n\}$
$(R^-)^{\mathcal{I}}$	$=$	$\{(x, y) \mid (y, x) \in R^{\mathcal{I}}\}$



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 $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$.
- \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



Sémantique formelle: modèle

Si $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ est une interprétation

- $a:C$ est satisfait par \mathcal{I} si $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\langle a,b \rangle : R$ est satisfait par \mathcal{I} si $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$
- Une interprétation \mathcal{I} est dite être un **modèle** de la ABox ($\mathcal{I} \models \mathcal{A}$) si toutes les assertions de \mathcal{A} sont satisfaites par \mathcal{I} .
- Une interprétation \mathcal{I} est dite être un **modèle** de la TBox ($\mathcal{I} \models \mathcal{T}$) si tous les axiomes de \mathcal{T} sont satisfaites par \mathcal{I} .

Une interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ est dite être un **modèle** de la base de connaissance Σ si tous les axiomes de Σ sont satisfaites par \mathcal{I} .

Une base de connaissance Σ est **satisfiable** si elle admet un modèle



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 insatisfiable ssi $C \sqsubseteq \perp$
- C & D sont équivalents ssi $C \sqsubseteq D \& D \sqsubseteq C$
- C & D sont incompatibles ssi $C \sqcap D \sqsubseteq \perp$



Réductions à l'insatisfiabilité

Possible si la négation est définie...

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

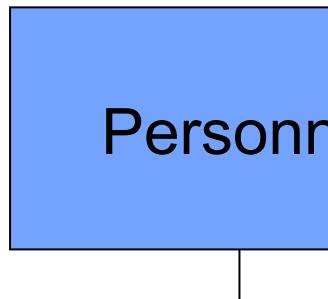


Procédures de raisonnement

- Il existe des algorithmes **complets** et **efficaces** pour décider de la **satisfiabilité**, 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 la satisfiabilité) 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



Pourquoi v a-t-il un problème?



TBox

Homme ⊑ Personne

Femme ⊑ Personne

Personne ⊑ ⊨ Age

Parent ≡ ⊨ Enfant ∩ Personne

Père ≡ ⊨ Enfant ∩ Homme

Mère ≡ ⊨ Enfant ∩ Femme

Père ≡ Parent ∩ Homme

Mère ≡ Parent ∩ Femme

ABox

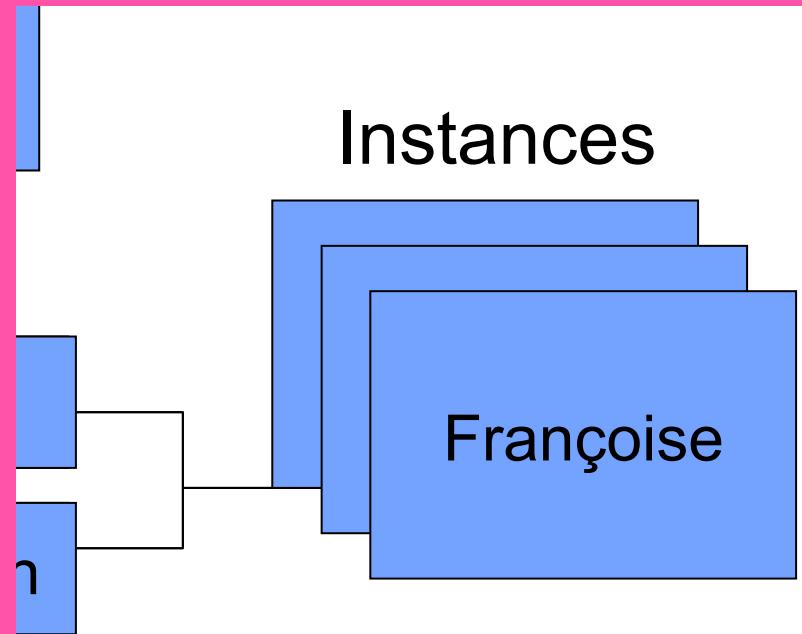
Françoise:Femme

<Françoise, 26>:Age

<Françoise, Gil>:Enfant

<Françoise, Jean>:Enfant

Instances



TBox

Homme ⊑ **Personne**

Femme ⊑ **Personne**

Personne ⊑ $\exists \text{Age}$

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

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

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

Père ≡ **Parent** $\sqcap \text{Homme}$

Mère ≡ **Parent** $\sqcap \text{Femme}$

?- femme(eve) .

?- homme(adam) .

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) .

En PROLOG: incomplétude

ABox

Françoise:Femme

<Françoise, 26>:Age

<Françoise, Gil>:Enfant

<Françoise, Jean>:Enfant

femme(francoise) .

age(francoise, 26) .

enfant(francoise, gil) .

enfant(francoise, jean) .

mere(eve) .

pere(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



Complexity of reasoning in Description Logics

Note: the information here is (always) incomplete and [updated](#) often

Base description logic: *A*tttributive *L*anguage with *C*omplements

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



Concept constructors:

- F - functionality²: $(\leq 1 R)$
- N - (unqualified) number restrictions: $(\geq n R)$, $(\leq n R)$
- Q - qualified number restrictions: $(\geq n R.C)$, $(\leq n R.C)$
- O - nominals: $\{a\}$ or $\{a_1, \dots, a_n\}$ ("one-of")
- μ - least fixpoint operator: $\mu X.C$

Forbid complex roles⁵ in number restrictions⁶

Role constructors:

- I – role inverse: R^-
- \cap – role intersection³: $R \cap S$
- \cup – role union: $R \cup S$
- \neg – role complement: $\neg R$
- \circ – role chain (composition): $R \circ S$
- $*$ – reflexive-transitive closure⁴: R^*
- id – concept identity: $id(C)$

trans reg

TBox (concept axioms):

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

RBox (role axioms):

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

OWL-Lite
OWL-DL
OWL 1.1

Reset

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



\mathcal{ALC} + empty TBox

TBox (concept axioms): <ul style="list-style-type: none"> <input checked="" type="radio"/> empty TBox <input type="radio"/> acyclic TBox ($A \equiv C$, A is a concept name; no cycles) <input type="radio"/> general TBox ($C \sqsubseteq D$, for arbitrary concepts C and D) 	RBox (role axioms): <ul style="list-style-type: none"> <input type="checkbox"/> S- role transitivity: $\text{Tr}(R)$ <input type="checkbox"/> \mathcal{H}- role hierarchy: $R \sqsubseteq S$ <input type="checkbox"/> \mathcal{R}- complex role inclusions: $R \circ S \sqsubseteq R$, $R \circ S \sqsubseteq S$ <input type="checkbox"/> s- some additional features (click to see them)
Reset	You have selected a Description Logic: \mathcal{ALC}
Complexity⁷ of reasoning problems⁸	
Concept satisfiability	PSpace-complete
ABox consistency	PSpace-complete
Important properties of the Description Logic	
Finite model property	Yes
Tree model property	Yes



\mathcal{ALC} + acyclic TBox

TBox (concept axioms): <ul style="list-style-type: none"> <input type="radio"/> empty TBox <input checked="" type="radio"/> acyclic TBox ($A \equiv C$, A is a concept name; no cycles) <input type="radio"/> general TBox ($C \subseteq D$, for arbitrary concepts C and D) 		RBox (role axioms): <ul style="list-style-type: none"> <input type="checkbox"/> \mathcal{S}- role transitivity: $\text{Tr}(R)$ <input type="checkbox"/> \mathcal{H}- role hierarchy: $R \subseteq S$ <input type="checkbox"/> \mathcal{R}- complex role inclusions: $R \circ S \subseteq R$, $R \circ S \subseteq S$ <input type="checkbox"/> s- some additional features (click to see them)
<input type="button" value="Reset"/>		You have selected a Description Logic: \mathcal{ALC}
Complexity⁷ of reasoning problems⁸		
Concept satisfiability	PSpace-complete	<ul style="list-style-type: none"> • <u>Hardness</u>: see empty TBox. • <u>Upper bound</u> for \mathcal{ALCQO}. see [17, Appendix A]. • An automata-based PSpace algorithm for \mathcal{ALC} with acyclic TBoxes is given in [44].
ABox consistency	PSpace-complete	<ul style="list-style-type: none"> • <u>Hardness</u> follows from that for concept satisfiability. • <u>Upper bound</u> 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 $\mathcal{ALCFI}_{\text{reg}}$ with any TBoxes; see [2, p.189, Theorem 5.6].



\mathcal{ALC} + arbitrary TBox

TBox (concept axioms):		RBox (role axioms):
<input type="radio"/> empty TBox <input type="radio"/> acyclic TBox ($A \equiv C$, A is a concept name; no cycles) <input checked="" type="radio"/> general TBox ($C \sqsubseteq D$, for arbitrary concepts C and D)		<input type="checkbox"/> S - role transitivity: $\text{Tr}(R)$ <input type="checkbox"/> H - role hierarchy: $R \sqsubseteq S$ <input type="checkbox"/> R - complex role inclusions: $R \circ S \sqsubseteq R$, $R \circ S \sqsubseteq S$ <input type="checkbox"/> s - some additional features (click to see them)
Reset You have selected a Description Logic: \mathcal{ALC}		
Complexity⁷ of reasoning problems⁸		
Concept satisfiability	ExpTime-complete	<ul style="list-style-type: none"> <u>Hardness</u>: originally proved in [77]; see also [2, Theorem 3.27]. <u>Upper bound</u>: an ExpTime tableaux algorithm is given in [33].
ABox consistency	ExpTime-complete	<ul style="list-style-type: none"> <u>Hardness</u> follows from ExpTime-hardness of concept satisfiability w.r.t. general TBoxes. <u>Upper bound</u> even for $SHIQ$ was proved in [12, Corollary 6.30].
Important properties of the Description Logic		
Finite model property	Yes	For all sublogics of $SHIQ$. This is mentioned in [63], where a similar result is obtained in Corollary 4.3 for $SHIQ$ extended with concrete domains and keys. (I did not find a "proper" reference for $SHIQ$ or its sublogics.)
Tree model property	Yes	For all sublogics of $\mathcal{ALCFI}_{\text{reg}}$ with any TBoxes; see [2, p.189, Theorem 5.6].



Une méthode des tableaux en logique des propositions

Étape 1: normalisation – mise sous **Forme Normale Négative (NNF – Negative Normal Form)**

Les négations n' interviennent que devant un nom de proposition

Repousser les négations vers l' intérieur à l' aide des lois de Morgan, de la suppression de la double négation

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

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

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



Méthode des tableaux en logique des propositions (suite)

Étape 2: **construction d' un tableau**

Racine: la formule sous *forme normale négative*

Construction des successeurs d' un tableau T à l' aide des quatre règles R_{\wedge} et R_{\vee} .

On s' arrête quand on ne peux plus appliquer de règle

$$\begin{array}{c} R_{\wedge} \frac{T}{T \cup \{\varphi, \psi\}} \boxed{\text{Si } \varphi \wedge \psi \in T \\ \text{et } (\varphi \notin T \\ \text{ou } \psi \notin T)} \\[10mm] R_{\vee} \frac{T}{T \cup \{\varphi\} | T \cup \{\psi\}} \boxed{\text{Si } \varphi \vee \psi \in T \\ \text{et } \varphi \notin T \\ \text{et } \psi \notin T} \end{array}$$



Méthode des tableaux

Définitions:

- **Tableau contradictoire:** il contient simultanément p et $\neg p$ (on dit qu'il contient un *clash*)
- **Tableau complet:** aucune règle ne lui est applicable
- **Un tableau est dit fermé s'il contient un clash, ouvert sinon**

L'algorithme part de la forme normale négative NNF de la ABox sur un tableau $\{\{C_0(x_0)\}\}$

On applique systématiquement les règles sur tous les tableaux.

La réponse est « satisfiable » si l'un des tableaux de l'arbre engendré est ouvert, insatisfiable sinon.

$$R_{\wedge} \frac{T}{T \cup \{\varphi, \psi\}}$$

Si $\varphi \wedge \psi \in T$
et ($\varphi \notin T$
ou $\psi \notin T$)

$$R_{\vee} \frac{T}{T \cup \{\varphi\} | T \cup \{\psi\}}$$

Si $\varphi \vee \psi \in T$
et ($\varphi \notin T$
et $\psi \notin T$)



Méthode des tableaux

Définitions:

- **Tableau contradictoire**: il contient simultanément p et $\neg p$ (on dit un qu'il contient un *clash*)
- Tableau complet: aucune règle ne lui est applicable
- Un tableau est dit *fermé* s'il contient un clash, *ouvert* sinon

L'algorithme part de la forme normale négative NNF de la ABox sur un tableau $\{\{C_0(x_0)\}\}$

On applique systématiquement les règles sur tous les tableaux.

La réponse est « satisfiable » si l'un des tableaux de l'arbre engendré est ouvert, insatisfiable sinon.

$$R_{\wedge} \frac{T}{T \cup \{\varphi, \psi\}}$$

Si $\varphi \wedge \psi \in T$
et ($\varphi \notin T$
ou $\psi \notin T$)

$$R_{\vee} \frac{T}{T \cup \{\varphi\} | T \cup \{\psi\}}$$

Si $\varphi \vee \psi \in T$
et ($\varphi \notin T$
et $\psi \notin T$)



Lemmes

- 1. Il n'y a pas de séquence infinie d'application des règles $\mathcal{S}_0 = \{\{C_0(x_0)\}\} \rightarrow \mathcal{S}_1 \rightarrow \mathcal{S}_1 \rightarrow \dots$**
- 2. Si \mathcal{S}' est obtenu à partir d'un ensemble fini \mathcal{S} de tableaux par application des règles de transformation, alors \mathcal{S}' est consistant ssi \mathcal{S} l'est.**
- 3. Tout tableau fermé est inconsistent**
- 4. Tout tableau complet et ouvert est consistant**



Troisième étape

- Soit T un tableau complet

Le modèle $M[T]$ qui satisfait φ est construit ainsi:

Si p , une proposition, appartient à T , alors p est vrai dans $M[T]$, sinon p est faux



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

Soit une description de concept C_0 mise sous *forme normale négative*.

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 ABoxes $\mathcal{S} = \{\mathcal{A}_1, \mathcal{A}_2 \dots, \mathcal{A}_k\}$

Cet ensemble est cohérent ssi il existe i , $1 \leq i \leq k$ tel que est \mathcal{A}_i cohérent.

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}'
- soit par deux nouvelles ABox \mathcal{A}' et \mathcal{A}''



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 suivantes:

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

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

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

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

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



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

Deuxième étape:

Construction tableau initial:

Pour tout fait $C(x)$ 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 complète 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}

- \Box -règle:

- **Condition:** \mathcal{A} contient $(C_1 \Box C_2)(x)$, mais on n'a pas $C_1(x)$ et $C_2(x)$
 - **Action:** $\mathcal{A}' := \mathcal{A} \cup \{C_1(x), C_2(x)\}$

- \sqcup -règle:

- **Condition:** \mathcal{A} contient $(C_1 \sqcup C_2)(x)$, mais ni $C_1(x)$ ni $C_2(x)$
 - **Action (choix non-déterministe):**

$$\mathcal{A}' := \mathcal{A} \cup \{C_1(x)\}, \mathcal{A}'' := \mathcal{A} \cup \{C_2(x)\}$$



Méthode des tableaux pour \mathcal{ALC}

- **\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:** $\mathcal{A}' = \mathcal{A} \cup \{C(y)\}$



Méthode des tableaux pour \mathcal{ALCN}

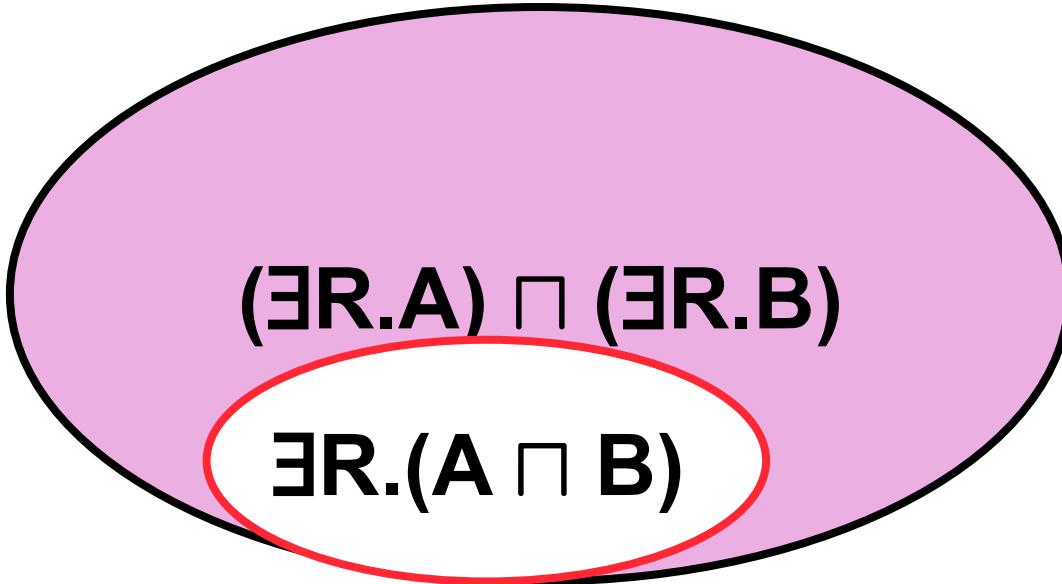
- **\geq -règle:**

- **Condition:** \mathcal{A} contient $(\geq n R)(x)$, mais il n'existe pas n constantes z_1, \dots, 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, \dots, y_{n+1} tels que $(\leq n R)(x)$ et $R(x, y_1), \dots, R(x, y_{n+1})$ 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}

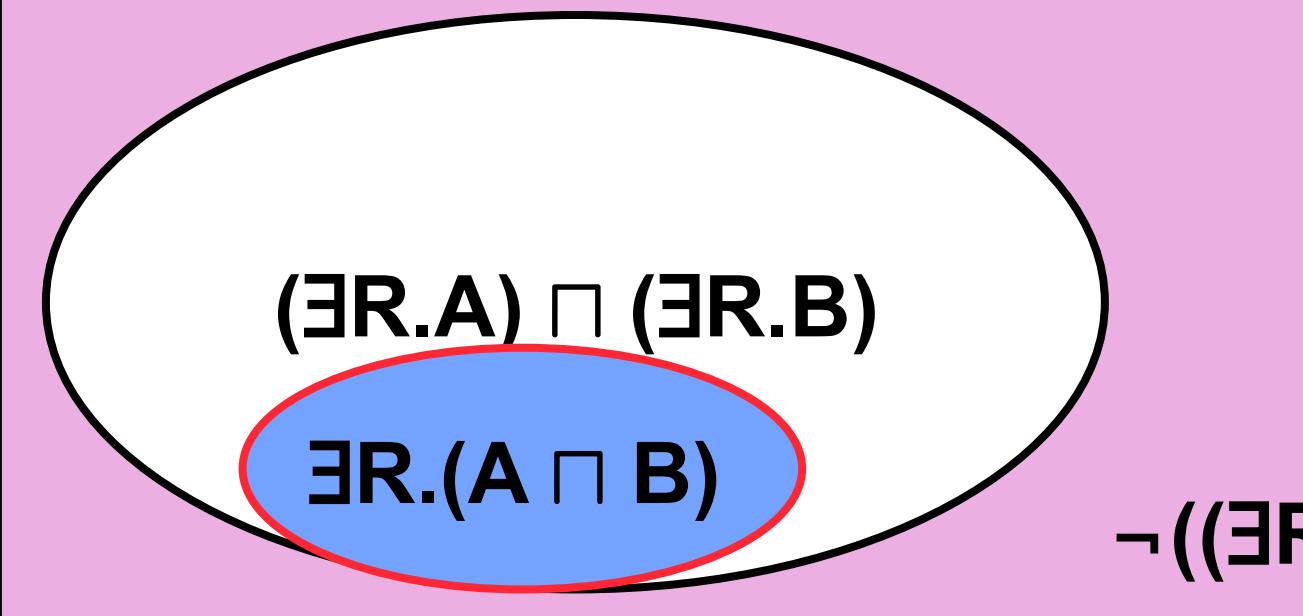




Exemple

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





Exemple

- Prouver $\exists R.(A \sqcap B) \sqsubseteq (\exists R.A) \sqcap (\exists R.B)$ valide
- Négation: $\neg(\exists R.(A \sqcap B) \sqsubseteq (\exists R.A) \sqcap (\exists R.B))$ insatisfiable
- $\neg(\neg \exists R.(A \sqcap B) \sqcup ((\exists R.A) \sqcap (\exists R.B)))$
- $\neg \exists R.(A \sqcap B) \sqcap \neg((\exists R.A) \sqcap (\exists R.B))$
- $\exists R.(A \sqcap B) \sqcap \neg((\exists R.A) \sqcap (\exists R.B))$





Exemple (suite)

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

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

Exemple - suite

Etape 2: tableaux

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

\sqcap -règle

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

\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

$$\begin{array}{l} \exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B)) \\ \exists R.(A \sqcap B) \\ ((\forall R.\neg A) \sqcup (\forall R.\neg B)) \\ (\forall R.\neg A) \end{array}$$

\exists -règle

$$\begin{array}{l} \exists R.(A \sqcap B) \sqcap ((\forall R.\neg A) \sqcup (\forall R.\neg B)) \\ \exists R.(A \sqcap B) \\ ((\forall R.\neg A) \sqcup (\forall R.\neg B)) \\ (\forall R.\neg A) \\ \text{R}(x,y), (A \sqcap B)(y) \end{array}$$


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

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





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

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

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

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

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

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

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

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

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

Eve: Mère

Adam: Père

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

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

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

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

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

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

Eve: Mère

Adam: Père

Démonstration avec la méthode des tableaux

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

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

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

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

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

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

Eve: Mère

Adam: Père

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

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

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

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

Démonstration avec la méthode des tableaux

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

C \neg Femme ⊑ Personne
N Parent ⊑ \neg ∃Enfant ⊑ \neg Personne
R Mère ⊑ \neg Parent ⊑ \neg Femme
S $(\exists$ Enfant ⊓ Personne) ⊑ \neg Parent
 $($ Parent ⊓ Femme) ⊑ \neg Mère
Eve:Mère

\neg Femme ⊑ Personne
Parent ⊑ \forall \neg Enfant ⊑ \neg Personne
Mère ⊑ \neg Parent ⊑ \neg Femme
 \exists Enfant ⊑ \neg Parent
Personne ⊑ \neg Parent
Femme ⊑ \neg Mère
Parent ⊑ \neg Mère
Eve:Mère





Démonstration

I
P
6

C
N
R
S

$\neg \text{Femme} \sqcup \text{Personne}$
 $\text{Parent} \sqcup \forall \neg \text{Enfant} \sqcup \neg \text{Personne}$
 $\text{Mère} \sqcup \neg \text{Parent} \sqcup \neg \text{Femme}$
 $\exists \text{Enfant} \sqcup \neg \text{Parent}$
 $\text{Personne} \sqcup \neg \text{Parent}$
 $\text{Femme} \sqcup \neg \text{Mère}$
 $\text{Parent} \sqcup \neg \text{Mère}$
Eve:Mère

$\neg \text{Femme} \sqcup \text{Personne}$
 $\text{Parent} \sqcup \forall \neg \text{Enfant} \sqcup \neg \text{Personne}$
 $\text{Mère} \sqcup \neg \text{Parent} \sqcup \neg \text{Femme}$
 $\exists \text{Enfant} \sqcup \neg \text{Parent}$
 $\text{Personne} \sqcup \neg \text{Parent}$
 $\text{Femme} \sqcup \neg \text{Mère}$
 $\text{Parent} \sqcup \neg \text{Mère}$
Eve:Mère
Femme

$\neg \text{Femme} \sqcup \text{Personne}$
 $\text{Parent} \sqcup \forall \neg \text{Enfant} \sqcup \neg \text{Personne}$
 $\text{Mère} \sqcup \neg \text{Parent} \sqcup \neg \text{Femme}$
 $\exists \text{Enfant} \sqcup \neg \text{Parent}$
 $\text{Personne} \sqcup \neg \text{Parent}$
 $\text{Femme} \sqcup \neg \text{Mère}$
 $\text{Parent} \sqcup \neg \text{Mère}$
Eve:Mère
 $\neg \text{Mère}$

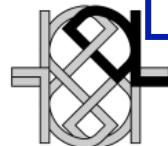


Tableau de droite

Contradiction

Jean-Gabriel

$\neg \text{Femme} \sqcup \text{Personne}$
 $\text{Parent} \sqcup \forall \neg \text{Enfant} \sqcup \neg \text{Personne}$
 $\text{Mère} \sqcup \neg \text{Parent} \sqcup \neg \text{Femme}$
 $\exists \text{Enfant} \sqcup \neg \text{Parent}$
 $\text{Personne} \sqcup \neg \text{Parent}$
 $\text{Femme} \sqcup \neg \text{Mère}$
 $\text{Parent} \sqcup \neg \text{Mère}$
 Eve:Mère
 $\neg \text{Mère}$

$\neg \text{Femme} \sqcup \text{Personne}$
 $\text{Parent} \sqcup \forall \neg \text{Enfant} \sqcup \neg \text{Personne}$
 $\text{Mère} \sqcup \neg \text{Parent} \sqcup \neg \text{Femme}$
 $\exists \text{Enfant} \sqcup \neg \text{Parent}$
 $\text{Personne} \sqcup \neg \text{Parent}$
 $\text{Femme} \sqcup \neg \text{Mère}$
 $\text{Parent} \sqcup \neg \text{Mère}$
 Eve:Mère
 $\neg \text{Mère}$
 $\text{Eve:}\neg \text{Mère}$

e CURIE

Les modèles
comprendront
tous
Eve:Femme

Tableau de gauche

$\neg \text{Femme} \sqcup \text{Personne}$
 $\text{Parent} \sqcup \forall \neg \text{Enfant} \sqcup \neg \text{Personne}$
 $\text{Mère} \sqcup \neg \text{Parent} \sqcup \neg \text{Femme}$
 $\exists \text{Enfant} \sqcup \neg \text{Parent}$
 $\text{Personne} \sqcup \neg \text{Parent}$
 $\text{Femme} \sqcup \neg \text{Mère}$
 $\text{Parent} \sqcup \neg \text{Mère}$
 Eve:Mère
 Femme

$\neg \text{Femme} \sqcup \text{Personne}$
 $\text{Parent} \sqcup \forall \neg \text{Enfant} \sqcup \neg \text{Personne}$
 $\text{Mère} \sqcup \neg \text{Parent} \sqcup \neg \text{Femme}$
 $\exists \text{Enfant} \sqcup \neg \text{Parent}$
 $\text{Personne} \sqcup \neg \text{Parent}$
 $\text{Femme} \sqcup \neg \text{Mère}$
 $\text{Parent} \sqcup \neg \text{Mère}$
 Eve:Mère
 Femme
 Eve:Femme

Autre exemple: Tbox « définitoire »

- **Tbox**

Woman \equiv Person \sqcap Female

Man \equiv Person \sqcap \neg Woman

Mother \equiv Woman \sqcap \exists hasChild.Person

Father \equiv Man \sqcap \exists hasChild.Person

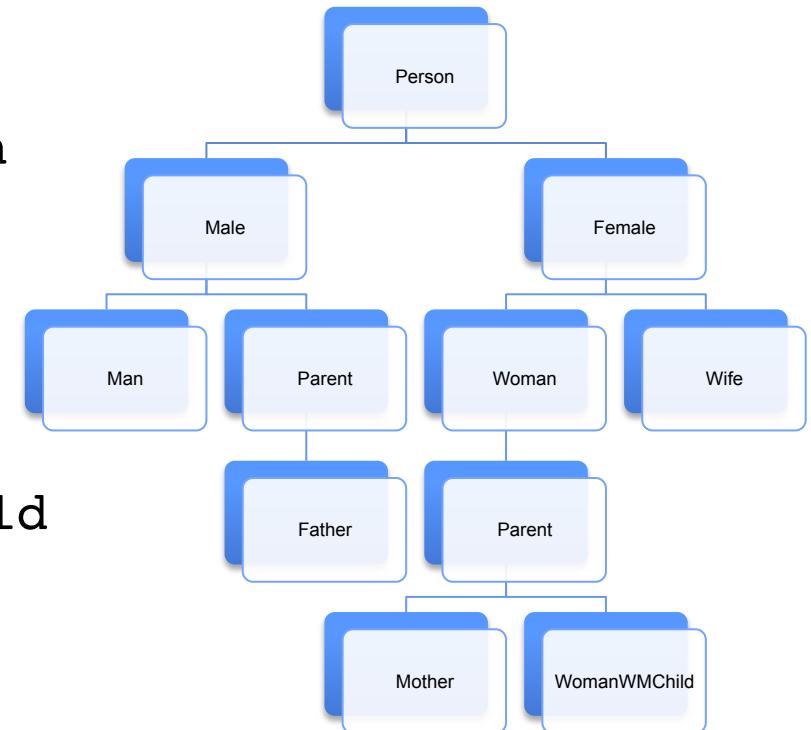
Parent \equiv Father \sqcup Mother

GrandMother \equiv Mother \sqcap
 \exists hasChild.Parent

MotherWMChild \equiv Mother \sqcap ≥ 3 hasChild

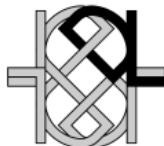
WomanWDaughter \equiv Mother \sqcap
 \forall hasChild. \neg Woman

Wife \equiv Woman \sqcap \exists 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 $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$.
- \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 »

Réécriture de la Abox

- **Tbox**

Woman ≡ Person \sqcap Female

Man ≡ Person \sqcap \neg Woman

Mother ≡ Woman \sqcap \exists hasChild.Person

Father ≡ Man \sqcap \exists hasChild.Person

Parent ≡ Father \sqcup Mother

GrandMother ≡ Mother \sqcap
 \exists hasChild.Parent

MotherWMChild ≡ Mother \sqcap ≥ 3 hasChild

WomanWDaughter ≡ Mother \sqcap
 \forall hasChild. \neg Woman

Wife ≡ Woman \sqcap \exists 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 \equiv Person \sqcap Female

Man \equiv Person $\sqcap \neg(\text{Person} \sqcap \text{Female})$

Mother $\equiv (\text{Person} \sqcap \text{Female}) \sqcap \exists \text{hasChild}.\text{Person}$

Father $\equiv (\text{Person} \sqcap \neg(\text{Person} \sqcap \text{Female})) \sqcap \exists \text{hasChild}.\text{Person}$

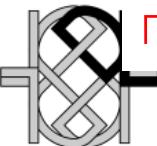
Parent $\equiv ((\text{Person} \sqcap \neg(\text{Person} \sqcap \text{Female})) \sqcap \exists \text{hasChild}.\text{Person}) \sqcup ((\text{Person} \sqcap \text{Female}) \sqcap \exists \text{hasChild}.\text{Person})$

GrandMother $\equiv (\text{Person} \sqcap \text{Female}) \sqcap \exists \text{hasChild}.\text{Person} \sqcap \exists \text{hasChild}.(\text{Person} \sqcap \exists \text{hasChild}.\text{Person})$

MotherWMChild $\equiv (\text{Person} \sqcap \text{Female}) \sqcap \exists \text{hasChild}.\text{Person} \sqcap \geq 3 \text{ hasChild}$

WomanWDaughter $\equiv (\text{Person} \sqcap \text{Female}) \sqcap \exists \text{hasChild}.\text{Person} \sqcap \forall \text{hasChild}. \neg(\text{Person} \sqcap \text{Female})$

Wife $\equiv (\text{Person} \sqcap \text{Female}) \sqcap \exists \text{hasHusband}.(\text{Person} \sqcap \neg(\text{Person} \sqcap \text{Female}))$



Transformation en NNF

- Tbox

Woman \equiv Person \sqcap Female

Man \equiv Person \sqcap \neg Female

Mother \equiv Person \sqcap Female \sqcap \exists hasChild.Person

Father \equiv Person \sqcap \neg Female \sqcap \exists hasChild.Person

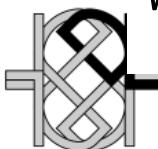
Parent \equiv Person \sqcap \exists hasChild.Person

GrandMother \equiv (Person \sqcap Female) \sqcap \exists hasChild.Person \sqcap
 \exists hasChild.(Person \sqcap \exists hasChild.Person)

MotherWMChild \equiv Person \sqcap Female \sqcap ≥ 3 hasChild

WomanWDaughter \equiv (Person \sqcap Female) \sqcap
 \forall hasChild. (\neg Person \sqcup \neg Female)

Wife \equiv Person \sqcap Female \sqcap \exists hasHusband. (Person \sqcap \neg Female)



Réécriture de la Abox avec la Tbox

- **Abox**

$\neg\text{GrandMother}(\text{MARY})$

$\neg\text{Person}(\text{MARY}) \sqcup \neg\text{Female}(\text{MARY}) \sqcup$

$\neg\exists\text{hasChild}.\text{Person}(\text{MARY}) \sqcup$

$\neg\exists\text{hasChild}.(\text{Person} \sqcap \exists\text{hasChild}.\text{Person})(\text{MARY})$

- **Tbox**

Woman \equiv Person \sqcap Female

Man \equiv Person \sqcap \neg Female

Mother \equiv Person \sqcap Female \sqcap $\exists\text{hasChild}.\text{Person}$

Father \equiv Person \sqcap \neg Female \sqcap $\exists\text{hasChild}.\text{Person}$

Parent \equiv Person \sqcap $\exists\text{hasChild}.\text{Person}$

GrandMother \equiv Person \sqcap Female \sqcap $\exists\text{hasChild}.\text{Person} \sqcap$

$\exists\text{hasChild}.(\text{Person} \sqcap \exists\text{hasChild}.\text{Person})$

MotherWMChild \equiv Person \sqcap Female \sqcap ≥ 3 hasChild

WomanWDaughter \equiv (Person \sqcap Female) \sqcap $\forall\text{hasChild}.(\neg\text{Person} \sqcup \neg\text{Female})$

Wife \equiv Person \sqcap Female \sqcap $\exists\text{hasHusband}.\text{(Person} \sqcap \neg\text{Female})$

- **Abox**

$\text{MotherWDaughter}(\text{MARY})$

$\text{Father}(\text{PETER})$

$\text{hasChild}(\text{MARY}, \text{PETER})$

$\text{hasChild}(\text{MARY}, \text{PAUL})$

$\text{hasChild}(\text{PETER}, \text{HARRY})$

$\text{Person}(\text{PAUL})$

$\text{Parson}(\text{HARRY})$

- **Réécritures Abox**

$\text{Person}(\text{MARY})$

$\text{Female}(\text{MARY})$

$\forall\text{hasChild}.(\neg\text{Person} \sqcup \neg\text{Female})(\text{MARY})$

$\text{Person}(\text{PETER})$

$\neg\text{Female}(\text{PETER})$

$\exists\text{hasChild}.\text{Person}(\text{PETER})$



Réécriture de la Abox avec la Tbox

- **Abox**

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Person(HARRY)

$\neg \text{Person}(\text{MARY}) \sqcup \neg \text{Female}(\text{MARY}) \sqcup$

$\neg \exists \text{hasChild}.\text{Person}(\text{MARY}) \sqcup$

$\neg \exists \text{hasChild}.(\text{Person} \sqcap \exists \text{hasChild}.\text{Person})(\text{MARY})$

Person(MARY)

Female(MARY)

$\forall \text{hasChild}.(\neg \text{Person} \sqcup \neg \text{Female})(\text{MARY})$

Person(PETER)

$\neg \text{Female}(\text{PETER})$

$\exists \text{hasChild}.\text{Person}(\text{PETER})$

- **Abox: terme réécrits**

MotherWDaughter(MARY)

Father(PETER)

$\neg \text{GrandMother}(\text{MARY})$



Tableaux 1

- **Abox**

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

$\neg \text{Person}(\text{MARY}) \sqcup \neg \text{Female}(\text{MARY}) \sqcup \neg \exists \text{hasChild}.\text{Person}(\text{MARY}) \sqcup$

$\neg \exists \text{hasChild}.(\text{Person} \sqcap \exists \text{hasChild}.\text{Person})(\text{MARY})$

Person(MARY)

Female(MARY)

$\forall \text{hasChild}.(\neg \text{Person} \sqcup \neg \text{Female})(\text{MARY})$

Person(PETER)

$\neg \text{Female}(\text{PETER})$

$\exists \text{hasChild}.\text{Person}(\text{PETER})$

Clash

...

$\neg \text{Person}(\text{MARY})$

Person(MARY)

...

$R \sqcup$

2



Tableau 2

- **Abox**

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

$\neg \text{Female}(\text{MARY}) \sqcup \neg \exists \text{hasChild}.\text{Person}(\text{MARY}) \sqcup$

$\neg \exists \text{hasChild}.(\text{Person} \sqcap \exists \text{hasChild}.\text{Person})(\text{MARY})$

Person(MARY)

Female(MARY)

$\forall \text{hasChild}.(\neg \text{Person} \sqcup \neg \text{Female})(\text{MARY})$

Person(PETER)

$\neg \text{Female}(\text{PETER})$

$\exists \text{hasChild}.\text{Person}(\text{PETER})$

Clash

...

$\neg \text{Female}(\text{MARY})$

Female(MARY)

...

$R \sqcup$

3



Tableau 3

- **Abox**

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

$\neg \exists \text{hasChild}.\text{Person}(\text{MARY}) \sqcup$

$\neg \exists \text{hasChild}.(\text{Person} \sqcap \exists \text{hasChild}.\text{Person})(\text{MARY})$

Person(MARY)

Female(MARY)

$\forall \text{hasChild}.(\neg \text{Person} \sqcup \neg \text{Female})(\text{MARY})$

Person(PETER)

$\neg \text{Female}(\text{PETER})$

$\exists \text{hasChild}.\text{Person}(\text{PETER})$

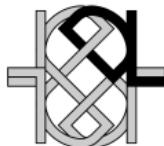
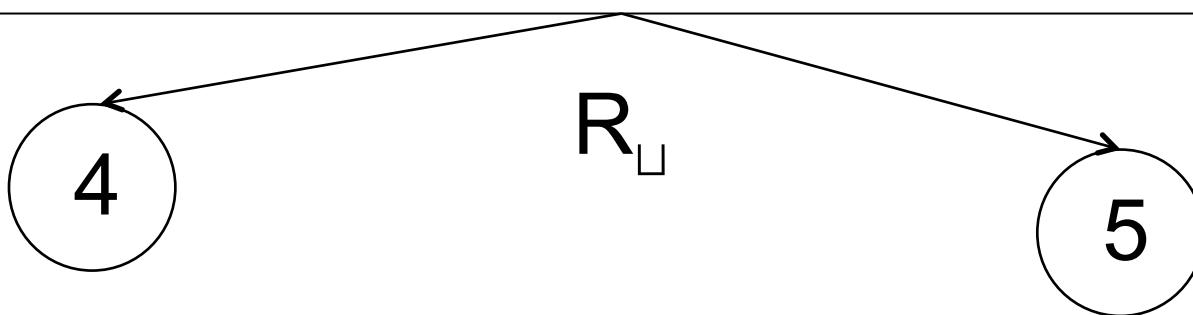


Tableau 4

- **Abox**

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

$\neg \exists \text{hasChild}.\text{Person}(\text{MARY})$

$\rightarrow \forall \text{hasChild}. \neg \text{Person}(\text{MARY})$

Person(MARY)

Female(MARY)

$\forall \text{hasChild}. (\neg \text{Person} \sqcup \neg \text{Female})(\text{MARY})$

Person(PETER)

$\neg \text{Female}(\text{PETER})$

$\exists \text{hasChild}.\text{Person}(\text{PETER})$

Clash



Tableau 5

- **Abox**

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Parson(HARRY)

$\neg \exists \text{hasChild.} (\text{Person} \sqcap \exists \text{hasChild.} \text{Person}) (\text{MARY})$

$\forall \text{hasChild.} \neg \text{Person} (\text{MARY}) \sqcup \forall \text{hasChild.} \forall \text{hasChild.} \neg \text{Person} (\text{MARY})$

Person(MARY)

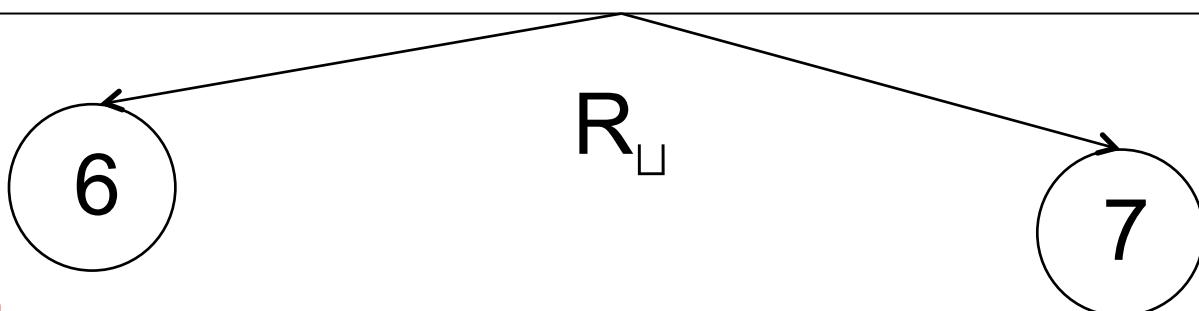
Female(MARY)

$\forall \text{hasChild.} (\neg \text{Person} \sqcup \neg \text{Female}) (\text{MARY})$

Person(PETER)

$\neg \text{Female} (\text{PETER})$

$\exists \text{hasChild.} \text{Person} (\text{PETER})$



Clash



Tableau 7

- **Abox**

hasChild(MARY, PETER)

hasChild(MARY, PAUL)

hasChild(PETER, HARRY)

Person(PAUL)

Person(HARRY)

$\forall \text{hasChild}.\forall \text{hasChild}.\neg \text{Person}(\text{MARY})$

$\forall \text{hasChild}.\neg \text{Person}(\text{PETER})$

$\neg \text{Person}(\text{HARRY})$

Person(MARY)

Female(MARY)

$\forall \text{hasChild}.(\neg \text{Person} \sqcup \neg \text{Female})(\text{MARY})$

Person(PETER)

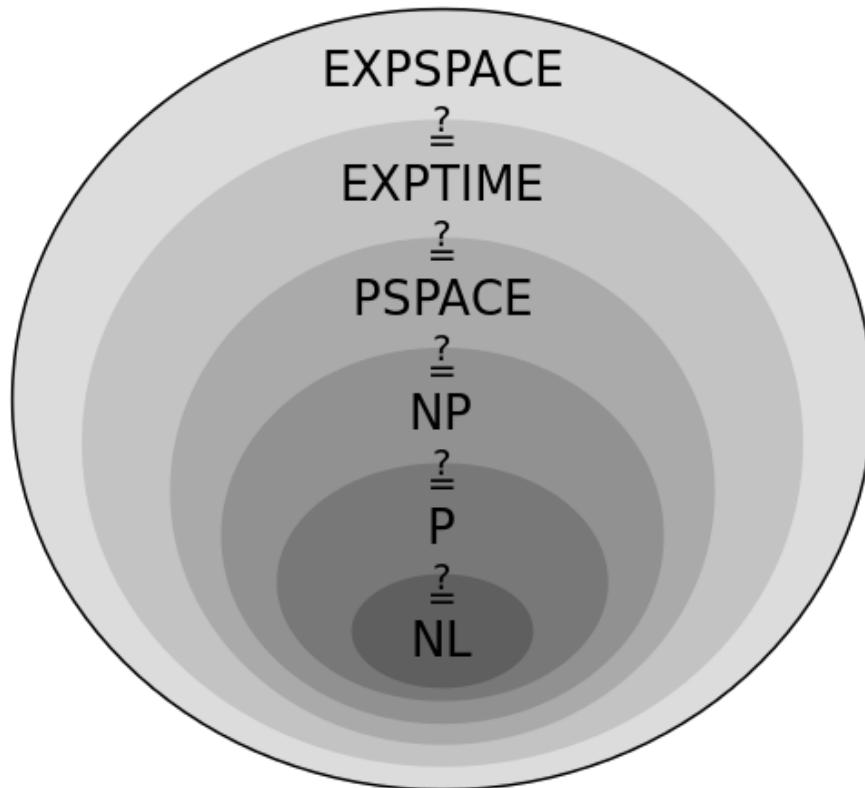
$\neg \text{Female}(\text{PETER})$

$\exists \text{hasChild}.\text{Person}(\text{PETER})$

Clash



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**)
- **EXPSPACE**: 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 $C \sqcap D$
 - Restriction $\forall R.C$
- **Mise sous forme normale dans \mathcal{FL}_0**
$$C \equiv A_1 \sqcap \dots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \dots \sqcap \forall R_n.C_n$$



\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>
```



Algorithme de subsomption structurelle dans \mathcal{FL}_0

$C \sqsubseteq D$

1. Normalisation

$$C \equiv A_1 \sqcap \dots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \dots \sqcap \forall R_n.C_n$$

$$D \equiv B_1 \sqcap \dots \sqcap B_k \sqcap \forall S_1.D_1 \sqcap \dots \sqcap \forall S_l.D_l$$

2. Vérifier récursivement:

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

$$\forall i \leq l \exists j \leq n \text{ tel que } S_i = R_j, C_i \sqsubseteq D_j$$



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

$C \sqsubseteq D$

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 $\forall r_1 \dots \forall r_m.A$ avec $m \geq 0$, r_1, \dots, r_m étant des noms de rôles et A un nom de concept atomique.

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

$$C \equiv \forall U_1.A_1 \sqcap \dots \sqcap \forall U_k.A_k \text{ et } D \equiv \forall V_1.A_1 \sqcap \dots \sqcap \forall V_k.A_k$$

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



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

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

$$C \equiv \forall U_1.A_1 \sqcap \dots \sqcap \forall U_k.A_k \text{ et } D \equiv \forall V_1.A_1 \sqcap \dots \sqcap \forall V_k.A_k$$

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



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

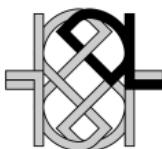
Syntaxe

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 définitoire
- 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 $C \sqcap D$
 - Restriction $\forall R.C$
- **Langage:** \mathcal{EL}
 - Conjonction $C \sqcap D$
 - Restriction $\exists R.C$
- **Langage:** \mathcal{ALN}
 - $\mathcal{AL}(C \sqcap D, \forall R.C, T, \perp, \neg A, \exists R.T)$
 - Restrictions sur cardinalités ($\geq nR, \leq nR$)

**Subsomption
structurelle**

- **Langage:** \mathcal{ALCN}
 - $\mathcal{ALN}(C \sqcap D, \forall R.C, T, \perp, \neg A, \exists R.T, \geq nR, \leq nR)$
 - Négation sans restriction

Tableaux



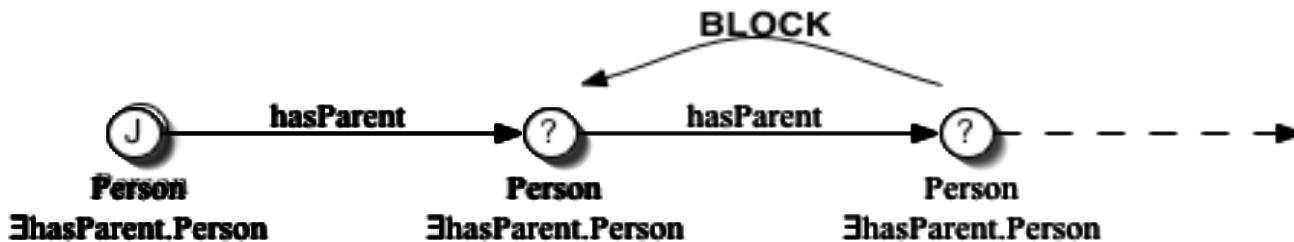
Méthode des tableaux

- 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) \rightarrow 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** (e.g., \sqcup , \sqsubseteq , \exists)
 - En pratique, cela signifie une **recherche**
- Vérification de cycle (**blocage**) souvent requis pour assurer la terminaison
 - exemple:
 $\{ \text{Person} \sqsubseteq \exists \text{hasParent}. \text{Person},$
 $\text{John:Person} \}$



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/*SHOIN* (jusqu' à maintenant),
- Des algorithmes ***SHOIQ*** résolvent ce (très embarrassant) problème
 - Mais les règles introduisent une nouvelle forme de non déterminisme

