

Intelligenza Artificiale II

Description Logics

Marco Piastra

Origini del Semantic Web

- Il Web è stato “inventato” da Tim Berners-Lee (et al.), un fisico del CERN di Ginevra
- La sua visione del Web era molto più ambiziosa di quanto poi trasformatosi in realtà: (syntactic) Web:



“... a plan for achieving a set of connected applications for data on the Web in such a way as to form a consistent logical web of data ...”

“... an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation ...”

- Questa visione è ora nota come **Semantic Web**

Scientific American, May 2001:



THE SEMANTIC WEB

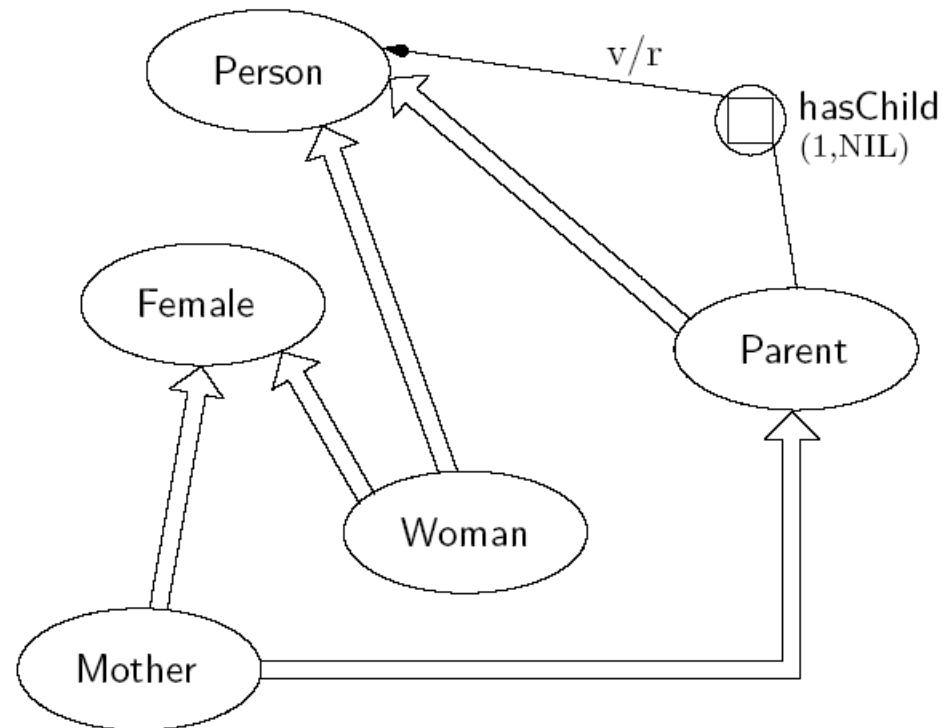
A new form of Web content
that is meaningful to computers
will unleash a revolution of new abilities

by
TIM BERNERS-LEE,
JAMES HENDLER and
ORA LASSILA

- La realizzazione di questa visione è (probabilmente) impossibile, per ora
- Si può cominciare aggiungendo **annotazioni semantiche** (semantic annotations) alle risorse web
- Alcune applicazioni in ambito e-Science ed e-Learning già esistono

Ontologie (per il *Semantic Web*)

- Annotazioni semantiche
in grado di rappresentare **concetti**, **relazioni** e **vincoli**



Uso delle ontologie

- Realizzazione di **strumenti** e servizi:

Progettare ed aggiornare **ontologie** di qualità:

Consistenti: tutti concetti possono avere istanze

Espressive: catturano le intuizioni degli esperti di dominio

Ridondanza minima: assenza di sinonimi non voluti

Adeguatamente assiomatizzate: descrizioni sufficientemente dettagliate

Basi di conoscenza: vaste ontologie

Semantic Annotations per il web

Rispondere a **query** su concetti ed istanze

Trovare concetti più specifici o più generali (*subsumption*)

Identificare pagine o risorse che corrispondono ad una descrizione (semantica)

Integrare ed allineare varie ontologie

Description Logics

Una famiglia di logiche, non una logica sola

- **Obiettivi**

 - Definizione di ontologie

 - Ragionamento automatico

- **Caratteristiche**

 - Un sottoinsieme della logica del primo ordine

 - Si usano solo predicati unari (concetti) e binari (relazioni)

 - Solo forme limitate di funzioni (attributi)

 - Vincoli sull'uso dei connettivi, possibilità di introdurre vincoli numerici

- **Famiglia di logiche**

 - Diminuendo i vincoli, si accresce il potere espressivo (e la complessità)

- **Vantaggi**

 - Sono in generale decidibili

 - Trattamento automatico (abbastanza) efficace

Architettura generale

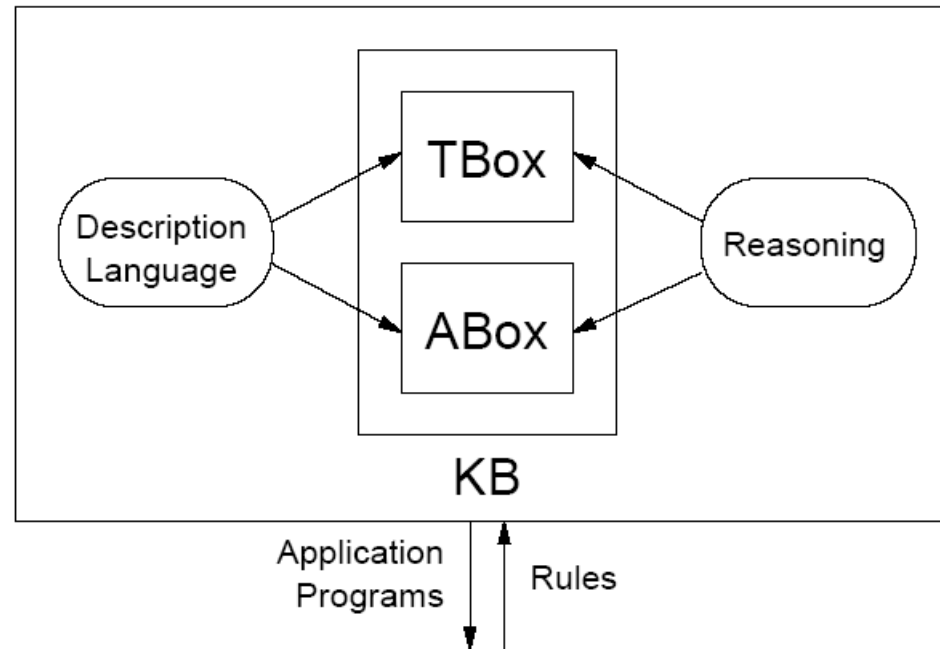


Fig. 2.1. Architecture of a knowledge representation system based on Description Logics.

- TBox e ABox sono insiemi di assiomi (di una teoria=ontologia)

TBox (anche T) Terminology: descrizioni concettuali

ABox (anche A) Assertion: descrizioni riguardanti istanze concrete

Linguaggio

- La più semplice DL: \mathcal{AL} (= *Attributive Language*)

Concetti atomici: A, B, C, \dots

Ruoli atomici: R, S, \dots

Concetto universale: \top

Concetto vuoto (*bottom concept*): \perp

Intersezione: \sqcap

Negazione atomica: \neg

Quantificatori di ruolo: \forall, \exists

- Formule ben formate (fbf)

A	(atomic concept)
\top	(universal concept)
\perp	(bottom concept)
$\neg A$	(atomic negation)
$C \sqcap D$	(intersection)
$\forall R.C$	(value restriction)
$\exists R.\top$	(limited existential quantification).

Esempio 2

$Person \sqcap Female$

Il concetto le cui istanze sono *Person* e *Female* (concetti)

Tipicamente usato in una definizione:

$Woman \equiv Person \sqcap Female$

$Person \sqcap \neg Female$

Il concetto le cui istanze sono *Person* e non *Female*

$Person \sqcap \exists hasChild.\top$

Il concetto le cui istanze sono *Person* ed hanno almeno un figlio

Tipicamente usato in una definizione:

$Parent \equiv Person \sqcap \exists hasChild.\top$

$Person \sqcap \forall hasChild.Female$

Il concetto le cui istanze sono *Person* ed hanno solo figli *Female*

Famglia delle logiche AL

- **Union** [\sqcup]

Si introduce il simbolo \sqcup
Amesse fbf del tipo $A \sqcup B$

- **Full existential quantifier** [\exists]

Ammesse fbf del tipo: $\exists R.C$
Esempio: $Person \sqcap \exists hasChild.Female$

- **Number restriction** [\succcurlyeq]

Si introducono i numeri naturali n ed i simboli \geq, \leq
Ammesse fbf del tipo $\geq n R$ e del tipo $\leq n R$
Esempio: $Person \sqcap (\geq 1 hasChild \sqcup (\leq 3 hasChild \sqcap \exists hasChild.Female))$

- **Complement** [\complement]

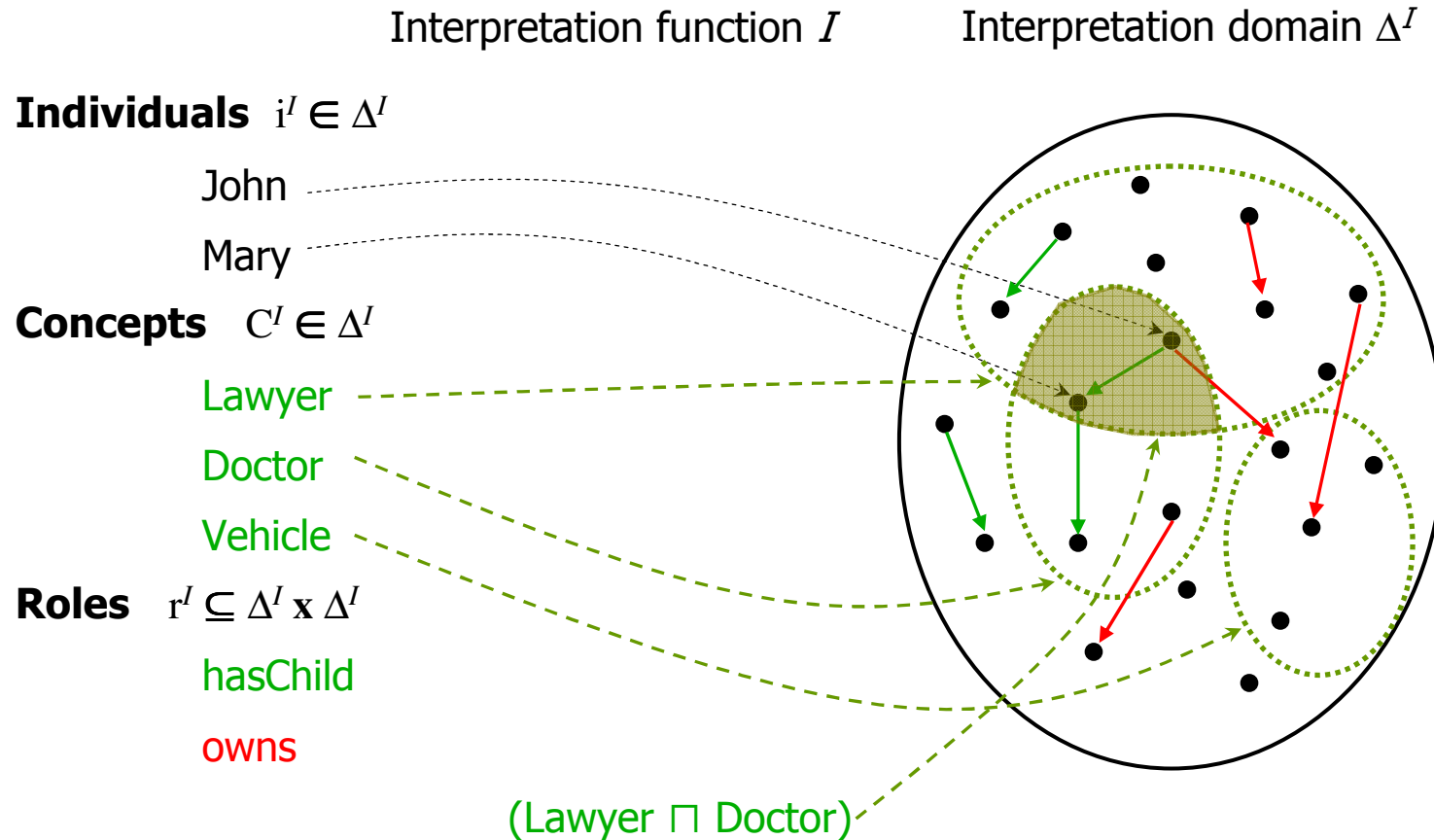
Ammesse fbf del tipo $\neg C$ (C concetto qualsiasi)
Esempio: $Person \sqcap \neg(Female \sqcup \exists hasChild.Female)$

- **Famiglia AL**

Una qualsiasi combinazione (esempio: ALC)

Abbreviazioni convenzionali: $ALC (= ALUEC)$, $ALCN (= ALUENC)$

Semantica intuitiva (da I. Horrocks)



Regole semantiche

▪ Interpretazione / linguaggio \mathcal{AL} e sue estensioni

in relazione ad un universo $\Delta^I (= U \text{ in } L_{PO})$

Interpretazione base: $C^I \subseteq \Delta^I, R^I \subseteq \Delta^I \times \Delta^I$

\mathcal{AL}

$$\top^I = \Delta^I$$

$$\perp^I = \emptyset$$

$$(\neg A)^I = \Delta^I \setminus A^I$$

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

$$(\forall R.C)^I = \{a \in \Delta^I \mid \forall b. (a, b) \in R^I \rightarrow b \in C^I\}$$

$$(\exists R.\top)^I = \{a \in \Delta^I \mid \exists b. (a, b) \in R^I\}.$$

\mathcal{U}

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

\mathcal{E}

$$(\exists R.C)^I = \{a \in \Delta^I \mid \exists b. (a, b) \in R^I \wedge b \in C^I\}.$$

\mathcal{N}

$$(\geq n R)^I = \left\{ a \in \Delta^I \mid |\{b \mid (a, b) \in R^I\}| \geq n \right\},$$

\mathcal{C}

$$(\leq n R)^I = \left\{ a \in \Delta^I \mid |\{b \mid (a, b) \in R^I\}| \leq n \right\},$$

Description Logics e L_{PO}

■ Regole di traduzione

$$\begin{aligned}\pi_x(A) &= A(x), \\ \pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D), \\ \pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D), \\ \pi_x(\exists R.C) &= \exists y.R(x, y) \wedge \pi_y(C), \\ \pi_x(\forall R.C) &= \forall y.R(x, y) \supset \pi_y(C), \\ \pi_x(\geq n R) &= \exists^{\geq n} y.R(x, y) = \exists y_1, \dots, y_n. \bigwedge_{i \neq j} y_i \neq y_j \wedge \bigwedge_i R(x, y_i) \\ \pi_x(\leq n R) &= \exists^{\leq n} y.R(x, y) = \forall y_1, \dots, y_{n+1}. \bigwedge_{i \neq j} y_i \neq y_j \supset \bigvee_i \neg R(x, y_i)\end{aligned}$$

Description Logics e L_{MP}

- Le *Description Logics* sono traducibili in logiche (multi)modali

Si vede confrontando le regole ST di L_{MP}

- Regole di traduzione

La funzione $f()$ definisce la traduzione

$$f(A) = A,$$

$$f(C \sqcap D) = f(C) \wedge f(D),$$

$$f(C \sqcup D) = f(C) \vee f(D),$$

$$f(\neg(C)) = \neg(f(C)),$$

$$f(\forall R_i.C) = \Box_i(f(C)),$$

$$f(\exists R_i.C) = \Diamond_i(f(C)).$$

Definizioni

- Una TBox contiene **definizioni**

Assiomi di una teoria espressi in un linguaggio della famiglia \mathcal{AL}

Si introduce il simbolo \equiv

Una definizione è una fbf del tipo $A \equiv \varphi$ (φ fbf qualsiasi)

Woman	\equiv	$\text{Person} \sqcap \text{Female}$
Man	\equiv	$\text{Person} \sqcap \neg \text{Woman}$
Mother	\equiv	$\text{Woman} \sqcap \exists \text{hasChild}.\text{Person}$
Father	\equiv	$\text{Man} \sqcap \exists \text{hasChild}.\text{Person}$
Parent	\equiv	$\text{Father} \sqcup \text{Mother}$
Grandmother	\equiv	$\text{Mother} \sqcap \exists \text{hasChild}.\text{Parent}$
MotherWithManyChildren	\equiv	$\text{Mother} \sqcap \geq 3 \text{ hasChild}$
MotherWithoutDaughter	\equiv	$\text{Mother} \sqcap \forall \text{hasChild}.\neg \text{Woman}$
Wife	\equiv	$\text{Woman} \sqcap \exists \text{hasHusband}.\text{Man}$

Fig. 2.2. A terminology (TBox) with concepts about family relationships.

Terminologie

- **Nomi simbolici** (*name symbols*)

In una definizione $A \equiv \varphi$ (φ fbf qualsiasi)

il simbolo A viene detto **nome simbolico** (di un concetto composto)

- **Terminologia T** (o TBox)

Un'insieme di definizioni in cui a ciascun nome simbolico corrisponde una sola definizione

- **Simboli base** (*base symbols*)

In una terminologia, l'insieme dei ruoli e dei concetti che occorrono solo nelle parti destre delle definizioni

- **Terminologia definitoria** (*definitory*)

Una T la cui interpretazione **estesa** (= completa) può essere costruita univocamente a partire dall'interpretazione dei simboli base

Non sono definitorie alcune T *cicliche*

$Human \equiv Animal \sqcap \forall hasParent.Human$

Ma alcune T cicliche sono definitorie ...

Qualsiasi T definitoria può essere tradotta in una T aciclica

Espansioni

▪ Espansione di una TBox

Sostituzione esaustiva dei nomi simbolici nelle parti destre

Woman	≡	Person \sqcap Female
Man	≡	Person \sqcap \neg (Person \sqcap Female)
Mother	≡	(Person \sqcap Female) \sqcap \exists hasChild.Person
Father	≡	(Person \sqcap \neg (Person \sqcap Female)) \sqcap \exists hasChild.Person
Parent	≡	((Person \sqcap \neg (Person \sqcap Female)) \sqcap \exists hasChild.Person) \sqcup ((Person \sqcap Female) \sqcap \exists hasChild.Person)
Grandmother	≡	((Person \sqcap Female) \sqcap \exists hasChild.Person) \sqcap \exists hasChild.(((Person \sqcap \neg (Person \sqcap Female)) \sqcap \exists hasChild.Person) \sqcup ((Person \sqcap Female) \sqcap \exists hasChild.Person))
MotherWithManyChildren	≡	((Person \sqcap Female) \sqcap \exists hasChild.Person) \sqcap ≥ 3 hasChild
MotherWithoutDaughter	≡	((Person \sqcap Female) \sqcap \exists hasChild.Person) \sqcap \forall hasChild. \neg (Person \sqcap Female)
Wife	≡	(Person \sqcap Female) \sqcap \exists hasHusband.(Person \sqcap \neg (Person \sqcap Female))

Fig. 2.3. The expansion of the Family TBox in Figure 2.2.

Terminologie generalizzate

■ Uso di assiomi di **inclusione**

Al posto di forme come

$$Woman \equiv Person \sqcap Female$$

è possibile usare una forma di **specializzazione**

$$Woman \sqsubseteq Person$$

Il concetto *Woman* è incluso in (= è sottoclasse) *Person*

La relazione classe/sottoclasse (p.es. UML) si traduce con \sqsubseteq

■ **Normalizzazione**

Una terminologia T **generalizzata** (= che contiene assiomi di inclusione) può essere tradotta in una forma \underline{T} **normalizzata**

sostituendo le forme $Woman \sqsubseteq Person$ con

$$\underline{Woman} \equiv \underline{Woman} \sqcap Person$$

dove \underline{Woman} è un nuovo simbolo base (che denota qualità specifiche)

In termini di espressività, le due forme sono equivalenti

Ogni modello di \underline{T} è anche modello di T

Qualsiasi interpretazione base di un modello di T è anche interpretazione base di un modello di \underline{T}

Asserzioni

- Una ABox (o A) contiene solo **asserzioni**

Fatti specifici in termini di istanze, concetti e ruoli e/o loro proprietà

Si introducono costanti individuali: a, b, c, ...

Esempi: $C(a)$, $R(b,c)$

MotherWithoutDaughter(MARY)	Father(PETER)
hasChild(MARY, PETER)	hasChild(PETER, HARRY)
hasChild(MARY, PAUL)	

Fig. 2.4. A world description (ABox).

Ragionamento: TBox

■ Quattro tipologie di problemi

In relazione ad una terminologia

- **Satisfiability** A concept C is satisfiable with respect to \mathcal{T} if there exists a model \mathcal{I} of \mathcal{T} such that $C^{\mathcal{I}}$ is nonempty. In this case \mathcal{I} is a model of C .
- **Subsumption** A concept C is subsumed by a concept D with respect to \mathcal{T} if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model \mathcal{I} of \mathcal{T} . Subsumption is denoted by $\mathcal{T} \models C \sqsubseteq D$.
- **Equivalence** Two concepts C and D are equivalent with respect to \mathcal{T} if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model \mathcal{I} of \mathcal{T} . Subsumption is denoted by $\mathcal{T} \models C = D$.
- **Disjointness** Two concepts C and D are disjoint with respect to \mathcal{T} if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model \mathcal{I} of \mathcal{T} .

Ragionamento: ABox

■ Quattro tipologie di problemi

In relazione ad un insieme di asserzioni

- **Consistency** An ABox \mathcal{A} is consistent with respect to a TBox \mathcal{T} , if there is an interpretation that is a model of both \mathcal{T} and \mathcal{A} .
- **Instance Checking** An individual is an instance of a concept C with respect to an ABox \mathcal{A} , if it is a member of the concept set of C . Instance checking is denoted by $\mathcal{A} \models C(a)$.
- **Retrieval**: retrieve all individuals that are instances of a given concept description
- **Realization problem**: find the most specific concept an individual belongs to

Riducibilità dei problemi

- In generale, nella famiglia \mathcal{AL}

Proposition 2.12 (Reduction to Subsumption) *For concepts C, D we have*

- (i) C is unsatisfiable $\Leftrightarrow C$ is subsumed by \perp ;
- (ii) C and D are equivalent $\Leftrightarrow C$ is subsumed by D and D is subsumed by C ;
- (iii) C and D are disjoint $\Leftrightarrow C \sqcap D$ is subsumed by \perp .

The statements also hold with respect to a $TBox$.

Proposition 2.13 (Reduction to Unsatisfiability) *For concepts C, D we have*

- (i) C is subsumed by D $\Leftrightarrow C \sqcap \neg D$ is unsatisfiable;
- (ii) C and D are equivalent \Leftrightarrow both $(C \sqcap \neg D)$ and $(\neg C \sqcap D)$ are unsatisfiable;
- (iii) C and D are disjoint $\Leftrightarrow C \sqcap D$ is unsatisfiable.

The statements also hold with respect to a $TBox$.

Metodi a tableau

- Le logiche della famiglia \mathcal{AL} (viste qui) sono decidibili

Il che significa che il problema della soddisfacibilità è decidibile

Inoltre:

Le logiche della famiglia \mathcal{AL} hanno la proprietà dei modelli finiti

La soddisfacibilità implica l'esistenza di un modello finito

Nel caso peggiore (*number restriction*) la complessità è esponenziale

- Metodi a tableau

(sono completi rispetto alla famiglia \mathcal{AL})

Hanno come obiettivo la dimostrazione di insoddisfacibilità

Quindi sono applicabili a tutte le quattro tipologie di problema, previa eventuale traduzione del problema originario

Tableaux per le Description Logics

■ Caratteristiche generali

A partire da una terminologia **T** (una TBox) ed un problema **P**

L'algoritmo cerca di costruire una **A** (una ABox) che contiene una contraddizione

Condizioni:

La **T** deve essere espansa

In **T** e **P** tutti i nomi simbolici sono stati sostituiti dalle parti destre in **T** (eliminazione dei nomi simbolici)

Le fbf in **T** e **P** devono essere trasformate in modo da contenere solo negazioni atomiche

Ciascun nodo del tableau contiene una **A**

Il nodo radice (la **A** iniziale) contiene **P** (il problema, cioè un concetto) applicato ad un individuo x_0

$A = \{\varphi_0(x_0)\}$, dove φ_0 è la fbf che descrive il problema

Regole per le Description Logics

- Ci si limita al caso $ALC (= ALU\mathcal{E}C)$

(vale a dire senza *number restriction*)

The \rightarrow_{\sqcap} -rule

Condition: \mathcal{A} contains $(C_1 \sqcap C_2)(x)$, but it does not contain both $C_1(x)$ and $C_2(x)$.

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

The \rightarrow_{\sqcup} -rule

Condition: \mathcal{A} contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$.

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

The \rightarrow_{\exists} -rule

Condition: \mathcal{A} contains $(\exists R.C)(x)$, but there is no individual name z such that $C(z)$ and $R(x, z)$ are in \mathcal{A} .

Action: $\mathcal{A}' = \mathcal{A} \cup \{C(y), R(x, y)\}$ where y is an individual name not occurring in \mathcal{A} .

The \rightarrow_{\forall} -rule

Condition: \mathcal{A} contains $(\forall R.C)(x)$ and $R(x, y)$, but it does not contain $C(y)$.

Action: $\mathcal{A}' = \mathcal{A} \cup \{C(y)\}$.

Terminazione

- La terminazione è garantita (decidibilità)

Quando A contiene un conflitto (*clash*) in tutti i suoi rami (successo)

- (i) $\{\perp(x)\} \subseteq \mathcal{A}$ for some individual name x ;
- (ii) $\{A(x), \neg A(x)\} \subseteq \mathcal{A}$ for some individual name x and some concept name A ;

Quando almeno un ramo di A è completo (= non si possono più applicare regole) e non ci sono conflitti (ciò implica fallimento)

Esempio 3

▪ Dimostrazione di *subsumption*

Given an unfoldable terminology \mathcal{T} containing the definition of **vegan** from Example 2.1:

$$\text{vegan} \doteq \text{person} \sqcap \forall \text{eats}.\text{plant}$$

and the following definition of **vegetarian**:

$$\text{vegetarian} \doteq \text{person} \sqcap \forall \text{eats}.\text{(plant} \sqcup \text{dairy)}$$

the algorithm can be used to show that $\text{vegan} \sqsubseteq_{\mathcal{T}} \text{vegetarian}$ by demonstrating that $(\text{vegan} \sqcap \neg \text{vegetarian})_{u\mathcal{T}}$ (i.e., $\text{vegan} \sqcap \neg \text{vegetarian}$ unfolded w.r.t. \mathcal{T}) is not satisfiable:

1. Unfold and normalise $\text{vegan} \sqcap \neg \text{vegetarian}$ to give:

$$\text{person} \sqcap \forall \text{eats}.\text{plant} \sqcap (\neg \text{person} \sqcup \exists \text{eats}.\text{(}\neg \text{plant} \sqcap \neg \text{dairy)})$$

2. Initialise \mathbf{T} to contain a single node x labelled:

$$\mathcal{L}(x) = \{\text{person} \sqcap \forall \text{eats}.\text{plant} \sqcap (\neg \text{person} \sqcup \exists \text{eats}.\text{(}\neg \text{plant} \sqcap \neg \text{dairy)})\}$$

Esempio 3

3. Apply the \sqcap -rule to $\text{person} \sqcap \forall \text{eats}.\text{plant} \sqcap (\neg \text{person} \sqcup \exists \text{eats}.\text{plant} \sqcap \neg \text{dairy}) \in \mathcal{L}(x)$:

$$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{\text{person}, \forall \text{eats}.\text{plant}, \neg \text{person} \sqcup \exists \text{eats}.\text{plant} \sqcap \neg \text{dairy}\}$$

4. Apply the \sqcup -rule to $\neg \text{person} \sqcup \exists \text{eats}.\text{plant} \sqcap \neg \text{dairy} \in \mathcal{L}(x)$:

- (a) Save \mathbf{T} and try:

$$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{\neg \text{person}\}$$

This is an obvious contradiction as $\{\text{person}, \neg \text{person}\} \subseteq \mathcal{L}(x)$.

- (b) Restore \mathbf{T} and try:

$$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{\exists \text{eats}.\text{plant} \sqcap \neg \text{dairy}\}$$

Esempio 3

5. Apply the \exists -rule to $\exists \mathit{eats} . (\neg \mathit{plant} \sqcap \neg \mathit{dairy}) \in \mathcal{L}(x)$:
create a new node y and a new edge $\langle x, y \rangle$
 $\mathcal{L}(y) = \{ \neg \mathit{plant} \sqcap \neg \mathit{dairy} \}$
 $\mathcal{L}(\langle x, y \rangle) = \mathit{eats}$
6. Apply the \forall -rule to $\forall \mathit{eats} . \mathit{plant} \in \mathcal{L}(x)$ and $\mathcal{L}(\langle x, y \rangle) = \mathit{eats}$:
 $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{ \mathit{plant} \}$
7. Apply the \sqcap -rule to $\neg \mathit{plant} \sqcap \neg \mathit{dairy} \in \mathcal{L}(y)$:
 $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{ \neg \mathit{plant}, \neg \mathit{dairy} \}$
This is an obvious contradiction as $\{ \mathit{plant}, \neg \mathit{plant} \} \subseteq \mathcal{L}(y)$.

As all possible applications of the \sqcup -rule (step 4) have now been shown to lead to a contradiction, it can be concluded that, with respect to the definitions in \mathcal{T} , $\mathit{vegan} \sqcap \neg \mathit{vegetarian}$ is unsatisfiable and thus that $\mathit{vegetarian}$ subsumes vegan .



3

OWL

La logica SHIQ

Famiglia [S]

ALC con aggiunta dei ruoli transitivi

$$R \in \text{Trn} \Rightarrow (\langle a,b \rangle \in R^I, \langle b,c \rangle \in R^I \Rightarrow \langle a,c \rangle \in R^I)$$

Role hierarchy [H]

Ammesse fbf del tipo: $R_2 \sqsubseteq R_1$ (Esempio: *hasDaughter* \sqsubseteq *hasChild*)

Inverse Roles [I]

Ammesse fbf del tipo: $R_1 \equiv R_2^-$ (Esempio: *hasChild* \equiv *isChildOf*⁻)

Nominals/singleton [O]

(Esempio: {*Italy*})

Qualified Number Restriction [Q]

Ammesse fbf del tipo $\geq n R.C$ e $\leq n R.C$ (Esempio: ≥ 3 *hasChild.Female*)

Functional Restriction [F] (una forma ristretta di \mathcal{N})

Ammesse fbf del tipo $\leq 1 R$ e $\geq 2 R$ (Esempio: ≤ 1 *hasTutor*)

■ SHIQ e OWL

OWL Lite: SHIF – Decidibile e di complessità limitata

OWL DL: SHOIQ – Decidibile ma di complessità elevata

OWL Full: forma particolare (equivale a RDF) - Non decidibile

Formato delle ontologie OWL (XML)

- Esempio

Student = Person \sqcap ≥ 1 enrolledIn

```
<owl:Class rdf:ID="Student">
  <owl:intersectionOf rdf:parsetype="Collection">
    <owl:Class rdfs:about="Person" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="enrolledIn" />
      <owl:minCardinality rdfs:datatype="&xsd;Integer">
        1
      </owl:minCardinality>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

- Riferimenti

<http://www.w3.org/TR/owl-features/>

Linguaggio OWL

Abstract Syntax	DL Syntax	Semantics
Descriptions (C)		
A (URI reference)	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
<code>owl:Thing</code>	\top	$\text{owl:Thing}^{\mathcal{I}} = \Delta^{\mathcal{I}}$
<code>owl:Nothing</code>	\perp	$\text{owl:Nothing}^{\mathcal{I}} = \{\}$
<code>intersectionOf($C_1 C_2 \dots$)</code>	$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
<code>unionOf($C_1 C_2 \dots$)</code>	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
<code>complementOf(C)</code>	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
<code>oneOf($o_1 \dots$)</code>	$\{o_1, \dots\}$	$\{o_1, \dots\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, \dots\}$
<code>restriction(R someValuesFrom(C))</code>	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y. \langle x, y \rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
<code>restriction(R allValuesFrom(C))</code>	$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y. \langle x, y \rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
<code>restriction(R hasValue(o))</code>	$R : o$	$(R : o)^{\mathcal{I}} = \{x \mid \langle x, o^{\mathcal{I}} \rangle \in R^{\mathcal{I}}\}$
<code>restriction(R minCardinality(n))</code>	$\geq n R$	$(\geq n R)^{\mathcal{I}} = \{x \mid \#(\{y. \langle x, y \rangle \in R^{\mathcal{I}}\}) \geq n\}$
<code>restriction(R maxCardinality(n))</code>	$\leq n R$	$(\leq n R)^{\mathcal{I}} = \{x \mid \#(\{y. \langle x, y \rangle \in R^{\mathcal{I}}\}) \leq n\}$