Intelligenza Artificiale

Description Logics

Marco Piastra

1

Semantic Web

Origini del Semantic Web

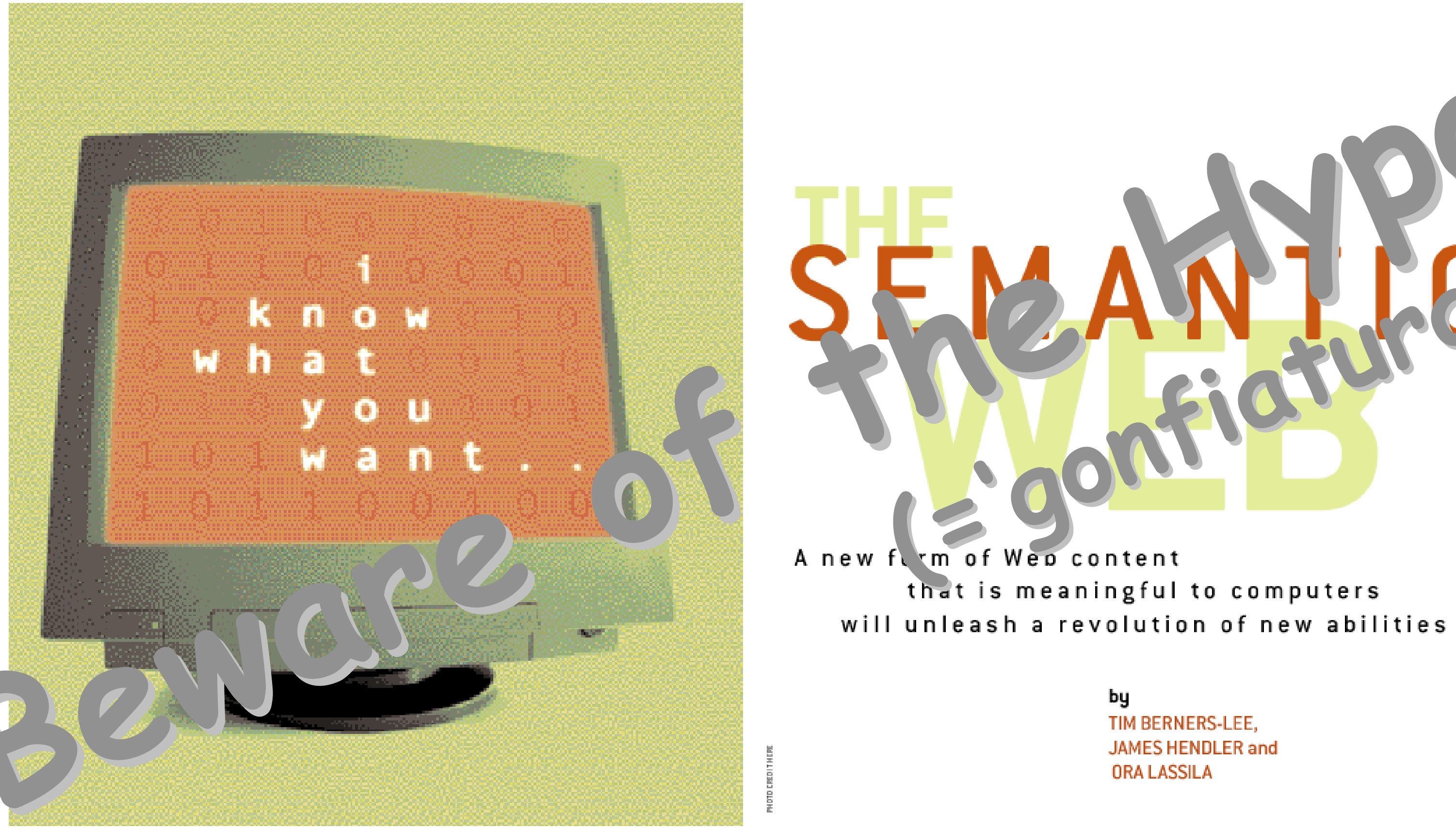
- Il Web è stato “inventato” da Tim Berners-Lee (et al.), un fisico del CERN di Ginevra
- La sua visione del Web era molto più ambiziosa di quanto poi trasformatosi in realtà: (syntactic) Web:

“… 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**

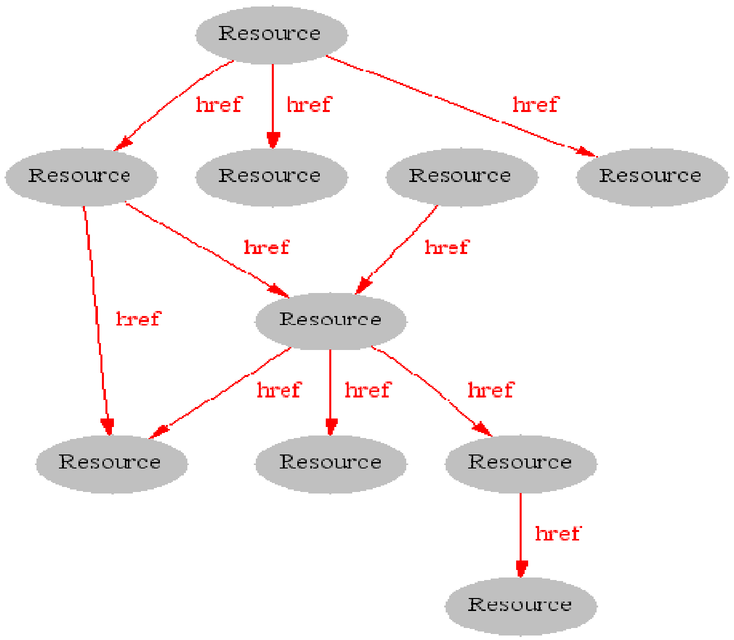
Scientific American, May 2001:



Beware of the Hype! (='gonfiatura')

- 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

Syntactic Web



- Un luogo dove le macchine si incaricano della **presentazione** e le persone del **collegamento** e dell'**interpretazione**
- Perchè non dare maggiori funzioni alle macchine?

Esempio 1: pagina web

- Pagina web di un convegno:

- La presentazione consiste di:
  - information rendering (e.g., font e colore)
  - hyperlink di collegamento
- Il contenuto semantico è interpretato dalle persone ...

Esempio 1: cosa vede una persona

WWW2002

The eleventh international world wide web conference

Sheraton waikiki hotel

Honolulu, hawaii, USA

7-11 may 2002

1 location 5 days learn interact

Registered participants coming from

australia, canada, chile denmark, france, germany, ghana, hong kong, india, ireland, italy, japan, malta, new zealand, the netherlands, norway, singapore, switzerland, the united kingdom, the united states, vietnam, zaire

Register now

On the 7th May Honolulu will provide the backdrop of the eleventh international world wide web conference. This prestigious event ...

Speakers confirmed

Tim berners-lee

Tim is the well known inventor of the Web, ...

Ian Foster

Ian is the pioneer of the Grid, the next generation internet ...

Esempio 1: cosa vede la macchina

⌖⌖⌖📄📁📁📄

❄♒♏ ♏●♏❖♏■⧫♒ ♓■⧫♏❒■♋⧫♓□■♋● ⬥□❒●♎ ⬥♓♎♏ ⬥♏♌ ♍□■♐♏❒♏■♍♏

💧♒♏❒♋⧫□■ ⬥♋♓🙵♓🙵♓ ♒□⧫♏●

☟□■□●◆●◆📪 ♒♋⬥♋♓♓📪 ✞💧✌

🖮📫📂📂 ❍♋⍓ 📄📁📁📄

📂 ●□♍♋⧫♓□■ 🗄 ♎♋⍓⬧ ●♏♋❒■ ♓■⧫♏❒♋♍⧫

☼♏♑♓⬧⧫♏❒♏♎ ❐♋❒⧫♓♍♓❐♋■⧫⬧ ♍□❍♓■♑ ♐❒□❍

♋◆⬧⧫❒♋●♓♋📪 ♍♋■♋♎♋📪 ♍♒♓●♏ ♎♏■❍♋❒🙵📪 ♐❒♋■♍♏📪 ♑♏❒❍♋■⍓📪 ♑♒♋■♋📪 ♒□■♑ 🙵□■♑📪 ♓■♎♓♋📪 ♓❒♏●♋■♎📪 ♓⧫♋●⍓📪 🙰♋❐♋■📪 ❍♋●⧫♋📪 ■♏⬥ ⌘♏♋●♋■♎📪 ⧫♒♏ ■♏⧫♒♏❒●♋■♎⬧📪 ■□❒⬥♋⍓📪 ⬧♓■♑♋❐□❒♏📪 ⬧⬥♓⧫⌘♏❒●♋■♎📪 ⧫♒♏ ◆■♓⧫♏♎ 🙵♓■♑♎□❍📪 ⧫♒♏ ◆■♓⧫♏♎ ⬧⧫♋⧫♏⬧📪 ❖♓♏⧫■♋❍📪 ⌘♋♓❒♏

☼♏♑♓⬧⧫♏❒ ■□⬥

⚐■ ⧫♒♏ 🖮⧫♒ 💣♋⍓ ☟□■□●◆●◆ ⬥♓●● ❐❒□❖♓♎♏ ⧫♒♏ ♌♋♍🙵♎❒□❐ ♐□❒ ⧫♒♏ ♏●♏❖♏■⧫♒ ♓■⧫♏❒■♋⧫♓□■♋● ⬥□❒●♎ ⬥♓♎♏ ⬥♏♌ ♍□■♐♏❒♏■♍♏📬 ❄♒♓⬧ ❐❒♏⬧⧫♓♑♓□◆⬧ ♏❖♏■⧫ ⑤

💧❐♏♋🙵♏❒⬧ ♍□■♐♓❒❍♏♎

❄♓❍ ♌♏❒■♏❒⬧📭●♏♏

❄♓❍ ♓⬧ ⧫♒♏ ⬥♏●● 🙵■□⬥■ ♓■❖♏■⧫□❒ □♐ ⧫♒♏ ⬥♏♌📪 ⑤

✋♋■ ☞□⬧⧫♏❒

✋♋■ ♓⬧ ⧫♒♏ ❐♓□■♏♏❒ □♐ ⧫♒♏ ☝❒♓♎📪 ⧫♒♏ ■♏⌧⧫ ♑♏■♏❒♋⧫♓□■ ♓■⧫♏❒■♏⧫ ⑤

Esempio 1: cosa vede la macchina (XML)

<name>🕈🕈🕈📄📁📁📄 ❄♒♏ ♏●♏❖♏■⧫♒ ♓■⧫♏❒■♋⧫♓□■♋● ⬥□❒●♎ ⬥♓♎♏ ⬥♏♌♍□■</name>

<location>⬥♋♓🙵♓🙵♓ ♒□⧫♏● ☟□■□●◆●◆📪 🕆💧✌</location>

<date>🖮📫📂📂 ❍♋⍓ 📄📁📁📄</date>

<participants>☼♏♑♓⬧⧫♏❒♏♎ ◻♋❒⧫♓♍♓◻♋■⧫⬧ ♍□❍♓■♑ ♐❒□❍ ♋◆⬧⧫❒♋●♓♋📪 ♍♋■♋♎♋📪 ♍♒♓●♏ ♎♏■❍♋❒🙵📪 ♐❒♋■♍♏📪 ♑♏❒❍♋■⍓📪 ♑♒♋■♋📪 ♒□■♑ 🙵□■♑📪 ♓■♎♓♋📪 ♓❒♏●♋■♎📪 ♓⧫♋●⍓📪 🙰♋◻♋■📪 ❍♋●⧫♋📪 ■♏⬥ ⌘♏♋●♋■♎📪 ⧫♒♏ ■♏⧫♒♏❒●♋■♎⬧📪 ■□❒⬥♋⍓📪 ⬧♓■♑♋◻□❒♏📪 ⬧⬥♓⧫⌘♏❒●♋■♎📪 ⧫♒♏ ◆■♓⧫♏♎ 🙵♓■♑♎□❍📪 ⧫♒♏ ◆■♓⧫♏♎ ⬧⧫♋⧫♏⬧📪 ❖♓♏⧫■♋❍📪 ⌘♋♓❒♏</participants>

<introduction>☼♏♑♓⬧⧫♏❒ ■□⬥ 🏳■ ⧫♒♏ 🖮⧫♒ 💣♋⍓ ☟□■□●◆●◆ ⬥♓●● ◻❒□❖♓♎♏ ⧫♒♏ ♌♋♍🙵♎❒□◻ □♐ ⧫♒♏ ♏●♏❖♏■⧫♒ ♓■⧫♏❒■♋⧫♓□■♋● ⬥□❒●♎ ⬥♓♎♏ ⬥♏♌ ♍□■♐♏❒♏■♍♏📬 ❄♒♓⬧ ◻❒♏⬧⧫♓♑♓□◆⬧ ♏❖♏■⧫ ⑤ 💧◻♏♋🙵♏❒⬧ ♍□■♐♓❒❍♏♎</introduction>

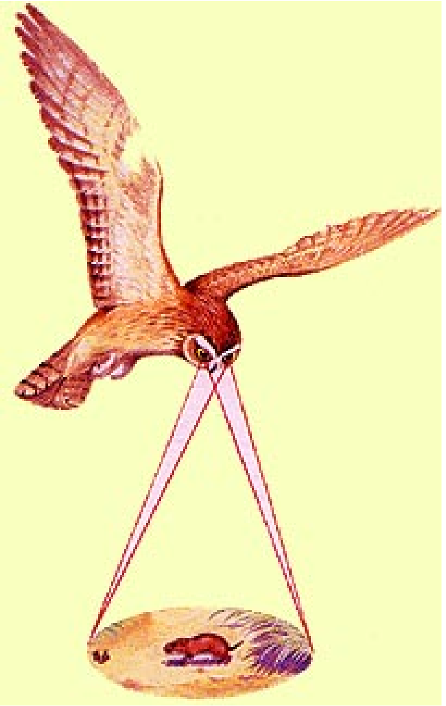
<speaker>❄♓❍ ♌♏❒■♏❒⬧📫●♏♏</speaker>

<bio>❄♓❍ ♓⬧ ⧫♒♏ ⬥♏●● 🙵■□⬥■ ♓■❖♏■⧫□❒ □♐ ⧫♒♏ ⬥♏♌📪</bio>…

Annotazioni semantiche

- Da associare a pagine, siti, aree del web
- Descrizione 'concettuale' dei contenuti
  - In un formato standard
  - Che può essere utilizzato in modo automatico (e.g. query)
- Esempio:
  - "Trova l'animale che usa il sonar
    e che non sia un pippistrello o un delfino"
  - (identificazione molto difficile per via testuale)

Barbagianni (Barn Owl)

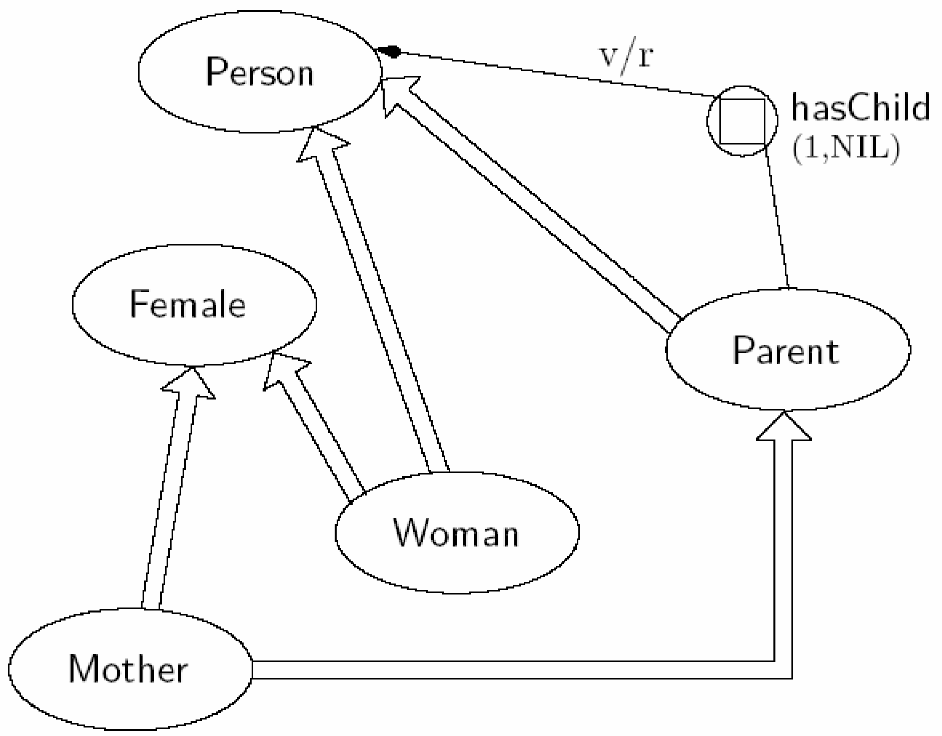


Ontologie

- Attenzione alle confusioni:
  - In filosofia, tradizionalmente, **Ontologia** è “la scienza dell’essere” (in origine una sorta di “metafisica”), di analisi delle entità prime
  - In informatica, recentemente, il termine **ontologia** ha assunto un significato pratico, di studio della “concettualizzazione”, linguistica e cognitiva, a scopo di trattamento automatico
    - “Definire le **ontologie** consente di formalizzare i modi di creazione della conoscenza, regolarne il trasferimento e garantirne la permanenza nel tempo e nello spazio, ovviamente a parità di condizioni: è la mobilità stessa della conoscenza che impone continue e fondamentali revisioni alle strutture formalizzate. È soprattutto uno strumento produttivo per operare il passaggio da sistemi informativi (lineari) a sistemi cognitivi (non routinarî, o sistemi socio-tecnici), che sappiano far fronte a un alto numero di eccezioni comunicative, senza “bloccarsi” dichiarando eccessi di “errore” o di “rumore”.
      [D. Bogliolo, AIDA Informazioni, anno 18, numero 2, aprile-giugno 2000]

Ontologie (per il Semantic Web)

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



Struttura delle ontologie

Le ontologie hanno due componenti principali:

- **Termini** (nomi), per i concetti rilevanti in un determinato dominio:
  - **Elefante** è un concetto le cui istanze sono un tipo di **Animali**
  - **Erbivoro** è un concetto le cui istanze **mangiano** solo **Piante** o parti di esse
  - **Elefante Adulto** è un concetto le cui istanze sono **Elefanti** con un **età** superiore a 20 anni
- **Vincoli** (altre informazioni) sul dominio:
  - Un **Elefante Adulto** pesa almeno 2.000 kg
  - Tutti gli **Elefanti** sono **Elefanti Indiani** o **Elefanti Africani**
  - Nessun **Animale** può essere **Erbivoro** e **Carnivoro**

Formato delle ontologie: OWL (XML)

```
<owl:Ontology rdf:about="">
  <rdfs:comment>Airport</rdfs:comment>
</owl:Ontology>

<rdfs:Class rdf:ID="Airport">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#name"/>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#iataCode"/>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#location"/>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</rdfs:Class>
```

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

1

Description Logics

Logica del primo ordine ed automazione

- Principi generali
- La logica del primo ordine non è decidibile
  - E` solo semi-decidibile
  - Il trattamento automatico è molto complesso e costoso
- Frammenti o forme ristrette della logica del primo ordine
  - I termini dello scambio sono:
    - Minori capacità espressive
    - Miglioramento della trattabilità e delle prestazioni
- Esempi:
  - Forma a clausole di Horn
    - Possibilità di identificare tutte le conseguenze logiche di un Γ
    - Theorem prover, programmazione logica
  - Regole universali implicative, solo istanziazione in base a fatti
    - Notevole efficienza computazionale (e.g. algoritmo Rete)
    - Sistemi esperti (e.g. Jess)

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

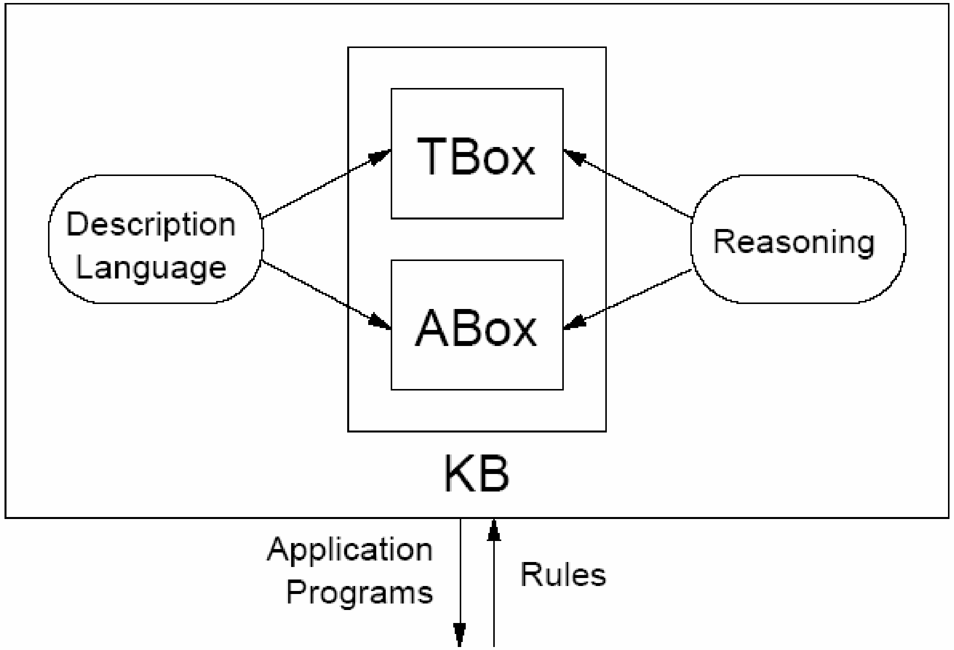


Fig. 2.1. Architecture of a knowledge representation system based on Description Logics.

- **TBox** - Terminology: descrizioni concettuali
- **ABox** – Assertion: descrizioni riguardanti istanze concrete

Linguaggio

- La più semplice DL: $\mathcal{AL}$ (= Attributive Language)
  - Concetti atomici: A, B, C, ...
  - Ruoli atomici: *R*, *S*, ...
  - Concetto universale: $\top$
  - Concetto vuoto (*bottom concept*): $\bot$
  - Intersezione: $\sqcap$
  - Negazione atomica: $\neg$
  - Quantificatori di ruolo: $\forall$, $\exists$
- Formule ben formate (fbf)

A	(atomic concept)
$\top$	(universal concept)
$\bot$	(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 dei linguaggi $\mathcal{AL}$

- *Union* [$\mathcal{U}$]
    - Si introduce il simbolo $\sqcup$
    - Amesse fbf del tipo A $\sqcup$ B
- *Full existential quantifier* [$\mathcal{E}$]
    - Ammesse fbf del tipo: $\exists R$.C
    - Esempio: Person $\sqcap$ $\exists$hasChild.Female
- *Number restriction* [$\mathcal{N}$]
    - 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* [$\mathcal{C}$]
    - Ammesse fbf del tipo $\neg C$ (*C* concetto qualsiasi)
    - Esempio: Person $\sqcap$ $\neg$(Female $\sqcup$ $\exists$hasChild.Female)
- Famiglia $\mathcal{AL}$
    - Una qualsiasi combinazione (esempio: $\mathcal{ALC}$)
    - Abbreviazioni convenzionali: $\mathcal{ALC}$ (= $\mathcal{ALUEC}$), $\mathcal{ALCN}$ (= $\mathcal{ALUENC}$)

Traduzione in logica del primo ordine

Expression	FOPC formula
$\top$	True
$\bot$	False
$C \sqcap D$	$C(x) \wedge D(x)$
$C \sqcup D$	$C(x) \vee D(x)$
$\neg C$	$\neg C(x)$
$\exists R.C$	$\exists y.(R(x,y) \wedge C(y))$
$\forall R.C$	$\forall y.(R(x,y) \rightarrow C(y))$
$\geqslant nR$	$\exists y_1, \ldots, y_n. \left(\bigwedge_{1 \leqslant i \leqslant n} \left(R(x, y_i) \wedge \bigwedge_{i < j \leqslant n} y_i \neq y_j \right) \right)$
$\leqslant nR$	$\forall y_1, \ldots, y_{n+1}. \left(\left(\bigwedge_{1 \leqslant i \leqslant n+1} R(x, y_i) \right) \rightarrow \bigvee_{1 \leqslant i < j \leqslant n+1} y_i = y_j \right)$
$\geqslant nR.C$	$\exists y_1, \ldots, y_n. \left(\bigwedge_{1 \leqslant i \leqslant n} \left(R(x, y_i) \wedge C(y_i) \wedge \bigwedge_{i < j \leqslant n} y_i \neq y_j \right) \right)$
$\leqslant nR.C$	$\forall y_1, \ldots, y_{n+1}. \left(\left(\bigwedge_{1 \leqslant i \leqslant n+1} (R(x, y_i) \wedge C(y_i)) \right) \rightarrow \bigvee_{1 \leqslant i < j \leqslant n+1} y_i = y_j \right)$
$\exists A.C$	$\exists y.((A(x) = y) \wedge C(y))$
$\forall A.C$	$\forall y.((A(x) = y) \rightarrow C(y))$
$A = B$	$\forall y.((A(x) = y) \leftrightarrow (B(x) = y))$
$A \neq B$	$\exists y.\neg((A(x) = y) \leftrightarrow (B(x) = y))$

Semantica

- Interpretazione del $\mathcal{I}$ linguaggio $\mathcal{AL}$ e sue estensioni
  - in relazione ad un universo $\Delta^{\mathcal{I}}$ (indicato come **U** in LPO)
    - Interpretazione base: $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
    - $\mathcal{AL}$

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b.\ (a,b) \in R^{\mathcal{I}} \to b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a,b) \in R^{\mathcal{I}}\}.
\end{aligned}
$$

    - $\mathcal{U}$ $\quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}.$
    - $\mathcal{E}$ $\quad (\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}.$
    - $\mathcal{N}$ $\quad (\geqslant n\, R)^{\mathcal{I}} = \left\{a \in \Delta^{\mathcal{I}} \;\middle|\; |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \geq n\right\},$
    - $\mathcal{C}$ $\quad (\leqslant n\, R)^{\mathcal{I}} = \left\{a \in \Delta^{\mathcal{I}} \;\middle|\; |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \leq n\right\},$

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 $A \equiv C$ (C concetto qualsiasi)

$$
\begin{array}{rcl}
\mathsf{Woman} & \equiv & \mathsf{Person} \sqcap \mathsf{Female} \\
\mathsf{Man} & \equiv & \mathsf{Person} \sqcap \neg\mathsf{Woman} \\
\mathsf{Mother} & \equiv & \mathsf{Woman} \sqcap \exists\mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Father} & \equiv & \mathsf{Man} \sqcap \exists\mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Parent} & \equiv & \mathsf{Father} \sqcup \mathsf{Mother} \\
\mathsf{Grandmother} & \equiv & \mathsf{Mother} \sqcap \exists\mathsf{hasChild}.\mathsf{Parent} \\
\mathsf{MotherWithManyChildren} & \equiv & \mathsf{Mother} \sqcap \geqslant 3\,\mathsf{hasChild} \\
\mathsf{MotherWithoutDaughter} & \equiv & \mathsf{Mother} \sqcap \forall\mathsf{hasChild}.\neg\mathsf{Woman} \\
\mathsf{Wife} & \equiv & \mathsf{Woman} \sqcap \exists\mathsf{hasHusband}.\mathsf{Man}
\end{array}
$$

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

Terminologie

- **Nomi simbolici** (*name symbols*)
    - In una definizione A ≡ C (C concetto qualsiasi)
      il simbolo A viene detto **nome simbolico** (di un concetto composito)
- **Terminologia** $\mathcal{T}$ 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 $\mathcal{T}$ la cui interpretazione **estesa** (= completa) può essere costruita univocamente a partire dall'interpretazione dei simboli base
  - Non sono definitorie alcune $\mathcal{T}$ *cicliche*

    Human ≡ Animal ⊓ ∀hasParent.Human

    - Ma alcune $\mathcal{T}$ cicliche sono definitorie …
  - Qualsiasi $\mathcal{T}$ definitoria può essere tradotta in una $\mathcal{T}$ aciclica

Espansioni

- **Espansione** di una TBox
  - Sostituzione esaustiva dei nomi simbolici nelle parti destre

$$
\begin{array}{rcl}
\mathsf{Woman} & \equiv & \mathsf{Person} \sqcap \mathsf{Female} \\
\mathsf{Man} & \equiv & \mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female}) \\
\mathsf{Mother} & \equiv & (\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Father} & \equiv & (\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Parent} & \equiv & ((\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
 & & \sqcup ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
\mathsf{Grandmother} & \equiv & ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
 & & \sqcap \exists \mathsf{hasChild}.(((\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \\
 & & \qquad \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
 & & \qquad \sqcup ((\mathsf{Person} \sqcap \mathsf{Female}) \\
 & & \qquad\quad \sqcap \exists \mathsf{hasChild}.\mathsf{Person})) \\
\mathsf{MotherWithManyChildren} & \equiv & ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \sqcap \geqslant 3\, \mathsf{hasChild} \\
\mathsf{MotherWithoutDaughter} & \equiv & ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \\
 & & \sqcap \forall \mathsf{hasChild}.(\neg(\mathsf{Person} \sqcap \mathsf{Female})) \\
\mathsf{Wife} & \equiv & (\mathsf{Person} \sqcap \mathsf{Female}) \\
 & & \sqcap \exists \mathsf{hasHusband}.(\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female}))
\end{array}
$$

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 $\mathcal{T}$ **generalizzata** (= che contiene assiomi di inclusione) può essere tradotta in una forma $\overline{\mathcal{T}}$ **normalizzata**
    - sostituendo le forme Woman $\sqsubseteq$ Person con

      Woman $\equiv$ $\overline{\text{Woman}}$ $\sqcap$ Person
    - dove $\overline{\text{Woman}}$ è un nuovo simbolo base (che denota le qualità specifiche)
  - In termini di espressività, le due forme sono equivalenti
    - Ogni modello di $\overline{\mathcal{T}}$ è anche modello di $\mathcal{T}$
    - Qualsiasi interpretazione base di un modello di $\mathcal{T}$ è anche interpretazione base di un modello di $\overline{\mathcal{T}}$

Asserzioni

- Una ABox 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* $\bot$*;*

(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* $\bot$*.*

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
    - Previa eventuale traduzione del problema originario

Tableaux per le Description Logics

- Caratteristiche generali
  - A partire da una terminologia $\mathcal{T}$ (una TBox) ed un problema $\mathcal{P}$
  - L'algoritmo cerca di costruire una $\mathcal{A}$ (una ABox) che contiene una contraddizione
  - Condizioni:
    - La $\mathcal{T}$ deve essere espansa
    - In $\mathcal{T}$ e P tutti i nomi simbolici sono stati sostituiti dalle parti destre in $\mathcal{T}$ (eliminazione dei nomi simbolici)
    - Le fbf in $\mathcal{T}$ e $\mathcal{P}$ devono essere trasformate in modo da contenere solo negazioni atomiche
    - Ciascun nodo del tableau contiene una $\mathcal{A}$
    - Il nodo radice (la $\mathcal{A}$ iniziale) contiene $\mathcal{P}$ (il problema, cioè un concetto) applicato ad un individuo x_o
      $\mathcal{A} = \{C_0(x_0)\}$, dove C_0 è la fbf che descrive il problema

Regole per le Description Logics

- Ci si limita al caso $\mathcal{ALC}$ (= $\mathcal{ALUEC}$)
  - (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 $\mathcal{A}$ contiene un conflitto (*clash*) in tutti i suoi rami (successo)

(i) $\{\bot(x)\} \subseteq \mathcal{A}$ 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 $\mathcal{A}$ è completo
(=non si possono più applicare regole) e non ci sono più conflitti
(fallimento)

Esempio 3

- Dimostrazione di *subsumption*

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

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

and the following definition of vegetarian:

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

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

1. Unfold and normalise $\mathsf{vegan} \sqcap \neg\mathsf{vegetarian}$ to give:
$$\mathsf{person} \sqcap \forall \mathit{eats}.\mathsf{plant} \sqcap (\neg\mathsf{person} \sqcup \exists \mathit{eats}.(\neg\mathsf{plant} \sqcap \neg\mathsf{dairy}))$$

2. Initialise $\mathbf{T}$ to contain a single node x labelled:
$$\mathcal{L}(x) = \{\mathsf{person} \sqcap \forall \mathit{eats}.\mathsf{plant} \sqcap (\neg\mathsf{person} \sqcup \exists \mathit{eats}.(\neg\mathsf{plant} \sqcap \neg\mathsf{dairy}))\}$$

Esempio 3

3. Apply the $\sqcap$-rule to $\mathsf{person} \sqcap \forall \mathit{eats}.\mathsf{plant} \sqcap (\neg\mathsf{person} \sqcup \exists \mathit{eats}.(\neg\mathsf{plant} \sqcap \neg\mathsf{dairy})) \in \mathcal{L}(x)$:
$$\begin{aligned}\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \quad \cup \quad & \{\mathsf{person}, \forall \mathit{eats}.\mathsf{plant},\\ & \neg\mathsf{person} \sqcup \exists \mathit{eats}.(\neg\mathsf{plant} \sqcap \neg\mathsf{dairy})\}\end{aligned}$$

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

   (a) Save $\mathbf{T}$ and try:
   $$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{\neg\mathsf{person}\}$$
   This is an obvious contradiction as $\{\mathsf{person}, \neg\mathsf{person}\} \subseteq \mathcal{L}(x)$.

   (b) Restore $\mathbf{T}$ and try:
   $$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{\exists \mathit{eats}.(\neg\mathsf{plant} \sqcap \neg\mathsf{dairy}))\}$$

Esempio 3

5. Apply the $\exists$-rule to $\exists \textit{eats}.(\neg\mathsf{plant} \sqcap \neg\mathsf{dairy}) \in \mathcal{L}(x)$:

$$\begin{array}{l} \text{create a new node } y \text{ and a new edge } \langle x, y\rangle \\ \mathcal{L}(y) = \{\neg\mathsf{plant} \sqcap \neg\mathsf{dairy}\} \\ \mathcal{L}(\langle x, y\rangle) = \textit{eats} \end{array}$$

6. Apply the $\forall$-rule to $\forall \textit{eats}.\mathsf{plant} \in \mathcal{L}(x)$ and $\mathcal{L}(\langle x, y\rangle) = \textit{eats}$:

$$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\mathsf{plant}\}$$

7. Apply the $\sqcap$-rule to $\neg\mathsf{plant} \sqcap \neg\mathsf{dairy} \in \mathcal{L}(y)$:

$$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \quad \cup \quad \{\neg\mathsf{plant}, \neg\mathsf{dairy}\}$$

   This is an obvious contradiction as $\{\mathsf{plant}, \neg\mathsf{plant}\} \subseteq \mathcal{L}(y)$.

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 $\mathcal{T}$, $\mathsf{vegan} \sqcap \neg\mathsf{vegetarian}$ is unsatisfiable and thus that $\mathsf{vegetarian}$ subsumes vegan.