Slides based on
Jurafsky and Martin
“Speech and Language Processing”

Semantics 1/3
(Intro & representing meanings)

Ing. Roberto Tedesco, PhD
roberto.tedesco@polimi.it

NLP – AA 20-21

Semantics: 3 parts

1 - Representing meanings:
- The meaning of linguistic utterances, represented with formal structures
- Based on ***representation languages***
- Often, a Knowledge Base is used to describe the "world" that represents the "context"

2 - Semantic analysis:
- Semantic Analysis: from linguistic utterances to meaning representations
- Information Extraction
- Both SA and IE do not consider the ambiguity of words

3 - Lexical semantics:
- How to represent meanings of words
- How to **solve ambiguities** (Word Sense Disambiguation)

Representing meanings: Languages

Desiderata for representations

- **Verifiable**: It should be possible to verify that the sentence (i.e. the *meaning* of it) is true or false
  "Does Maharani serve vegetarian food?"
- **Unambiguous**: Ambiguities should be solved
  "I wanna eat someplace that's close to Politecnico"
  (like Godzilla???)
  "I want to eat Italian food" (which kind of italian food?)
- **Canonical form**: Different sentences with the same meaning should produce the same representation
  "Does Maharani serve vegetarian food?"
  "Do they have vegetarian food at Maharani?"

Desiderata for representations

- **Inferences and variables**: Inferences from the representation should be possible
  "Can vegetarians eat at Maharani?"
  → true/false
  "I'd like to find a restaurant where I can get vegetarian food"
  → list of vegeratian-food restaurants…
- **Expressiveness**: representation should be able to represent any natural language utterance
- Ok, let's start… First of all, what is the structure of human languages?

Meaning structure of language

- Assumption: all human languages are based on a form of predicate-arguments arrangement (i.e., subj-verb-complements)
  - The ***grammar*** organizes this structure
  - Computational representations rely on this structure
- Verbs dictate specific constraints on args (***frames***)

"Robert wants Italian food"
"Robert wants to spend less than five dollars"
"Robert wants it to be close by here"

Verbal frames

NP want NP
NP want Inf-VP
NP want NP Inf-VP

"NP[Robert] wants NP[Italian food]"
"NP[Robert] wants Inf-VP[to spend less than five dollars]"
"NP[Robert] wants NP[it] Inf-VP[to be close by here]"

Syntactic analysis

Meaning structure of language: Linking and roles

- In all those examples:
  - Someone wanting → want → something wanted
  - ***Thematic/case roles*** associated with the verb “to want”
- Verbal frames permit to link verbal arguments (the surface structure) with thematic/case roles (the underlying semantic representation)

“wanting[Robert] wants wanted[Italian food]”
“wanting[Robert] wants wanted[to spend less than five dollars]”
“wanting[Robert] wants wanted[it to be close by here]”

Meaning structure of language: Restrictions

- The verb “to want” restricts its arguments:
  - Only certain categories of concepts can play the role of “wanting” (“a *stone* wants…” ???)
  - Only certain categories of concepts can play the role of “wanted”
- ***Selectional restriction***: constraints that verbs pose on their arguments
- More on roles and restrictions in 3. “Lexical semantics”…
- Summing up, verbs define:
  - A surface form → frame
  - An underlying semantic structure → thematic/case roles
  - Restrictions on their arguments

Representation languages

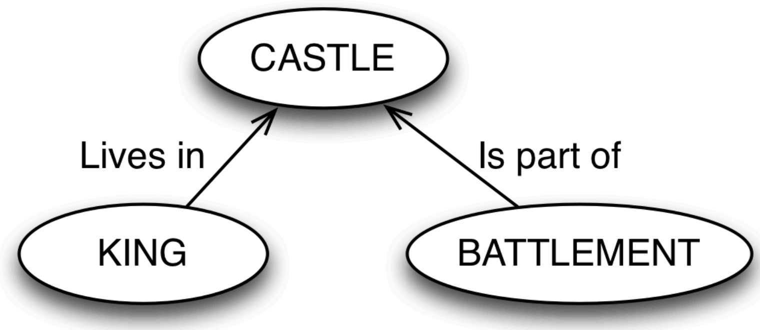
- Once understood the meaning structure of languages, we need a way to represent it
- We present some of them
- Languages representing ***meaning of utterances***:
  - Semantic Networks
  - Logics
  - Frame-based representations
- Languages representing ***meaning of words***:
  - Lexical databases

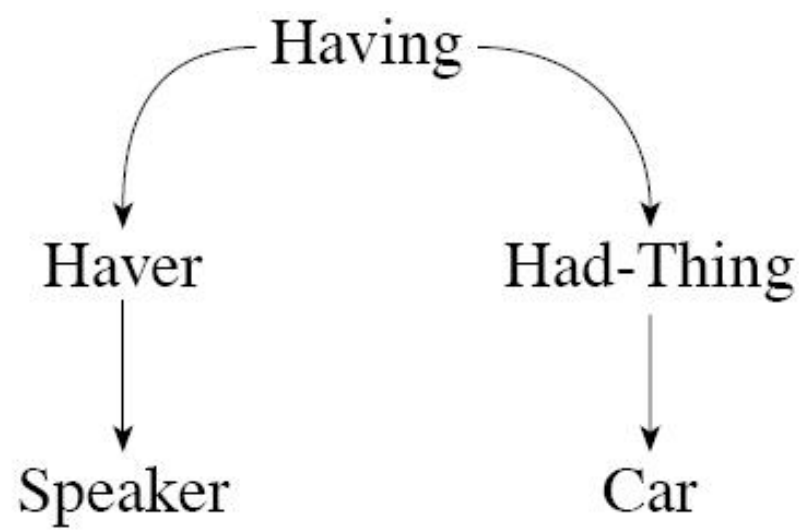
Semantic Networks (SNs)

- Graphic notation introduced to model the organization of human semantic memory, or memory for *word concepts*
  - Node: word concept
  - Arc: relationship between word concepts
- In this approach, *concepts* coincide with *words*
  - Knowledge modeling is language-dependent
- A word *meaning* is defined by the connections to other words

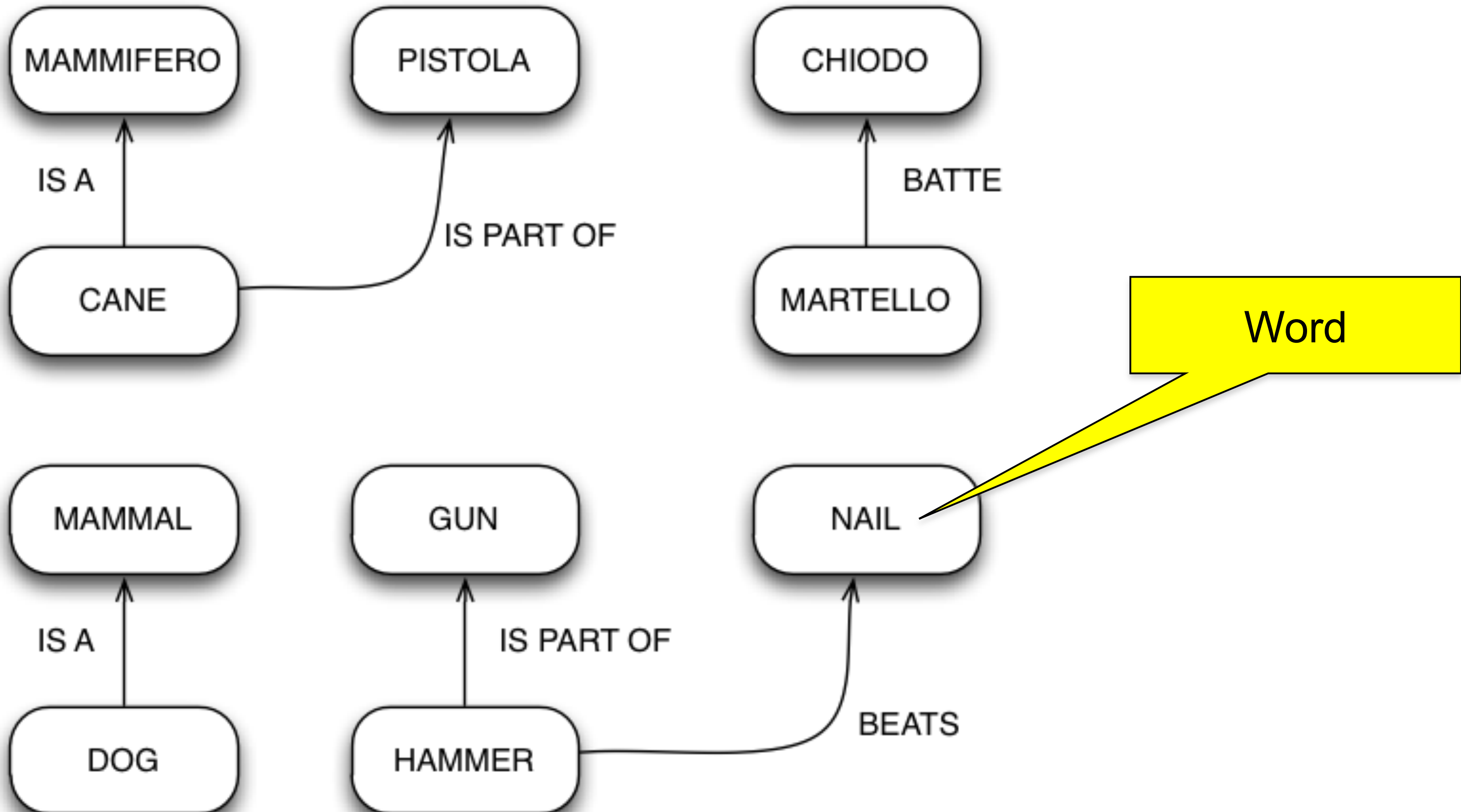
Semantic Networks: An example

- A network representing a domain (a KB)
  - Defines meanings of concepts
  - Description is language-dependent
- A network representing a sentence: “I have a car”
  - Defines the meaning of a sentence



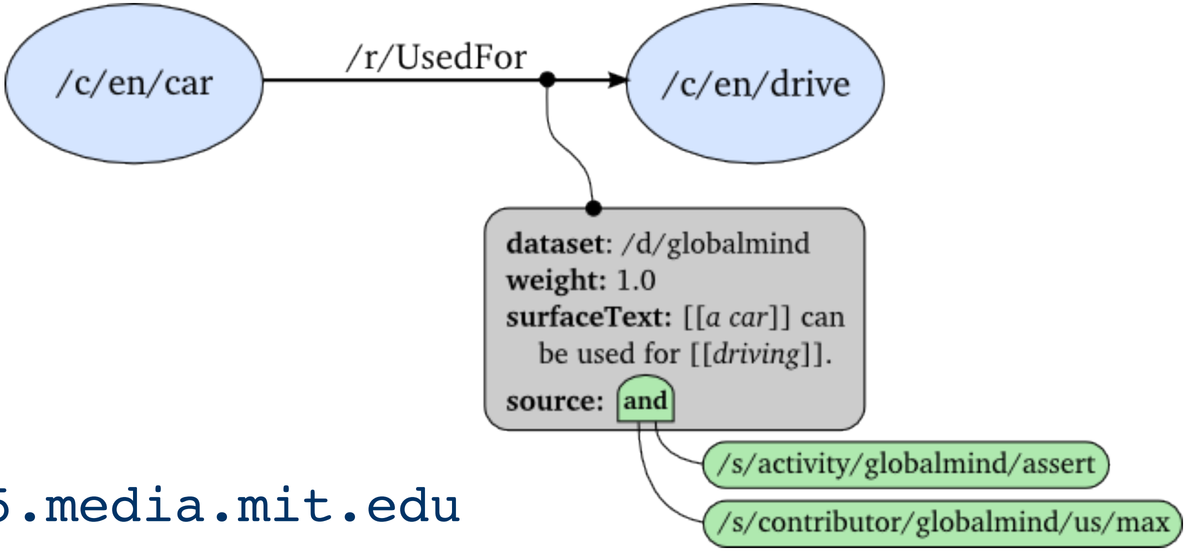


Semantic Networks are language dependent



Semantic Networks: characteristics

- Can describe both utterances and KBs
  - But such KBs are language dependent
- Do not have any formal basis
- ConceptNet is an example of (somehow extended) SN
  - Info on arcs
  - Relationship → surface text



http://conceptnet5.media.mit.edu

Logics

- Languages:
  - First-Order Logics (FOL)
  - Description Logic (DL)
  - Temporal logics
- Reasoning
- Implementations:
  - Production-rule systems
  - PROLOG
  - Ontologies

First Order Predicate Calculus (FOPC) aka First Order Logic (FOL)

$$
\begin{array}{rcl}
\mathit{Formula} & \rightarrow & \mathit{AtomicFormula} \\
& | & \mathit{Formula\ Connective\ Formula} \\
& | & \mathit{Quantifier\ Variable}, \ldots\ \mathit{Formula} \\
& | & \neg\ \mathit{Formula} \\
& | & (\mathit{Formula}) \\
\mathit{AtomicFormula} & \rightarrow & \mathit{Predicate}(\mathit{Term}, \ldots) \\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots) \\
& | & \mathit{Constant} \\
& | & \mathit{Variable} \\
\mathit{Connective} & \rightarrow & \wedge \mid \vee \mid \Rightarrow \\
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid \mathit{VegetarianFood} \mid \mathit{Maharani} \cdots \\
\mathit{Variable} & \rightarrow & x \mid y \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{Serves} \mid \mathit{Near} \mid \cdots \\
\mathit{Function} & \rightarrow & \mathit{LocationOf} \mid \mathit{CuisineOf} \mid \cdots
\end{array}
$$

Logical connectives & quantifiers

- Logical connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
$False$	$False$	$True$	$False$	$False$	$True$
$False$	$True$	$True$	$False$	$True$	$True$
$True$	$False$	$False$	$False$	$True$	$False$
$True$	$True$	$False$	$True$	$True$	$True$

- Quantifiers

“A restaurant that serves Mexican food at Politecnico”

$$\exists x Restaurant(x) \wedge Serves(x, MexicanFood) \wedge Near(LocationOf(x), LocationOf(Politecnico))$$

“All vegetarian restaurants serve vegetarian food”

$$\forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$$

First Order Logic (FOL)

- Can represent utterances:

"A restaurant that serves Mexican food at Politecnico"

$$\exists x Restaurant(x) \wedge Serves(x, MexicanFood) \wedge Near(LocationOf(x), LocationOf(Politecnico))$$

"All vegetarian restaurants serve vegetarian food"

$$\forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$$

"I have a car" ("I" is the speaker)

$$\exists e, y\ Having(e) \wedge Haver(e, Speaker) \wedge HadThing(e, y) \wedge Car(y)$$

- And, as SNs, can represent concepts (a KB)
  - Such description is language-independent
- Formally defined
- Reasoning

Reasoning

- FOL is semidecidable:
  - If a formula is true, there is a mechanical way to determine this result;
  - if a formula is false, however, such procedure may give a negative result or no result at all
- Restrictions of FOL exist that are decidable

Inference: Modus ponens

- Modus ponens: The most important inference method

$$\frac{\begin{array}{l} VegetarianRestaurant(Rudys) \\ \forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood) \end{array}}{Serves(Rudys, VegetarianFood)}$$

- It is used as forward chaining or backward chaining

Forward Chaining reasoning

- A new proposition (a ***fact***) added to the KB:

$$VegetarianRestaurant(Rudys)$$

- A ***rule*** in the KB:

$$\forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$$

- As the new fact is added, the rule fires:
  - because the ***antecedent*** matches the new fact
  - variable x is instantiated: x=$Rudys$
  - then, the ***consequent*** asserts a new fact into the KB:

$$Serves(Rudys, VegetarianFood)$$

Backward Chaining reasoning (1/2)

- Fact in the KB: $VegetarianRestaurant(Rudys)$
- Rule in the KB:

$$\forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$$

- Query: $Serves(Rudys, VegetarianFood)$?
- ***Consequent*** matches the query: $x{=}Rudys$ …
- … then: if antecedent is true, the query is true
  - Searching for: $VegetarianRestaurant(Rudys)$
- Antecedent is true (it is in the KB), then consequent is true, then query is true

Backward Chaining reasoning (2/2)

- Fact in the KB: $VegetarianRestaurant(Rudys)$
- Rule in the KB; consequent matches the query…

$$\forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$$

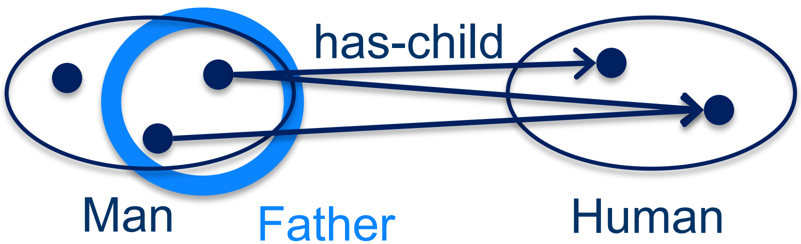
- Query: $Serves(x, VegetarianFood) \quad ?$
- … then: query has answer if antecedent matches
  - Searching for: $VegetarianRestaurant(x)$
- Antecedent matches, the answer is: $x = Rudys$

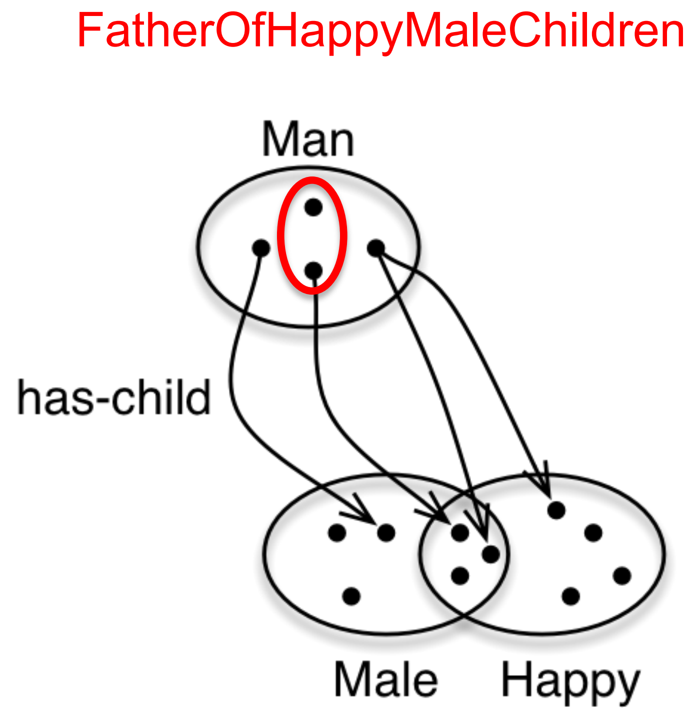
Inferences

- Neither forward nor backward reasoning are complete
  - I.e., there are valid inferences that cannot be found
- Complete reasoning techniques exist, but they are much more time consuming
  - E.g., the Tableau Calculus

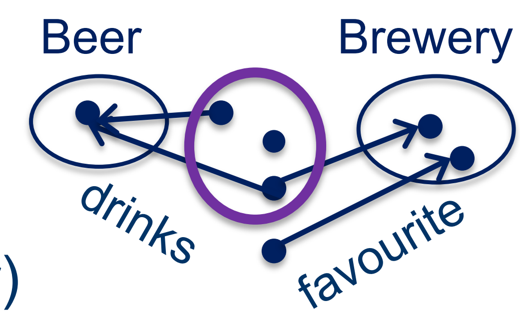
Description Logics

- Subset of FOL
- Decidable
- $Father = Man \sqcap \exists has\text{-}child.Human$





- $FatherOfHappyMaleChildren = Man \sqcap \forall has\text{-}child.(Male \sqcap Happy)$
- $\exists favourite.Brewery \sqsubseteq \exists drinks.Beer$

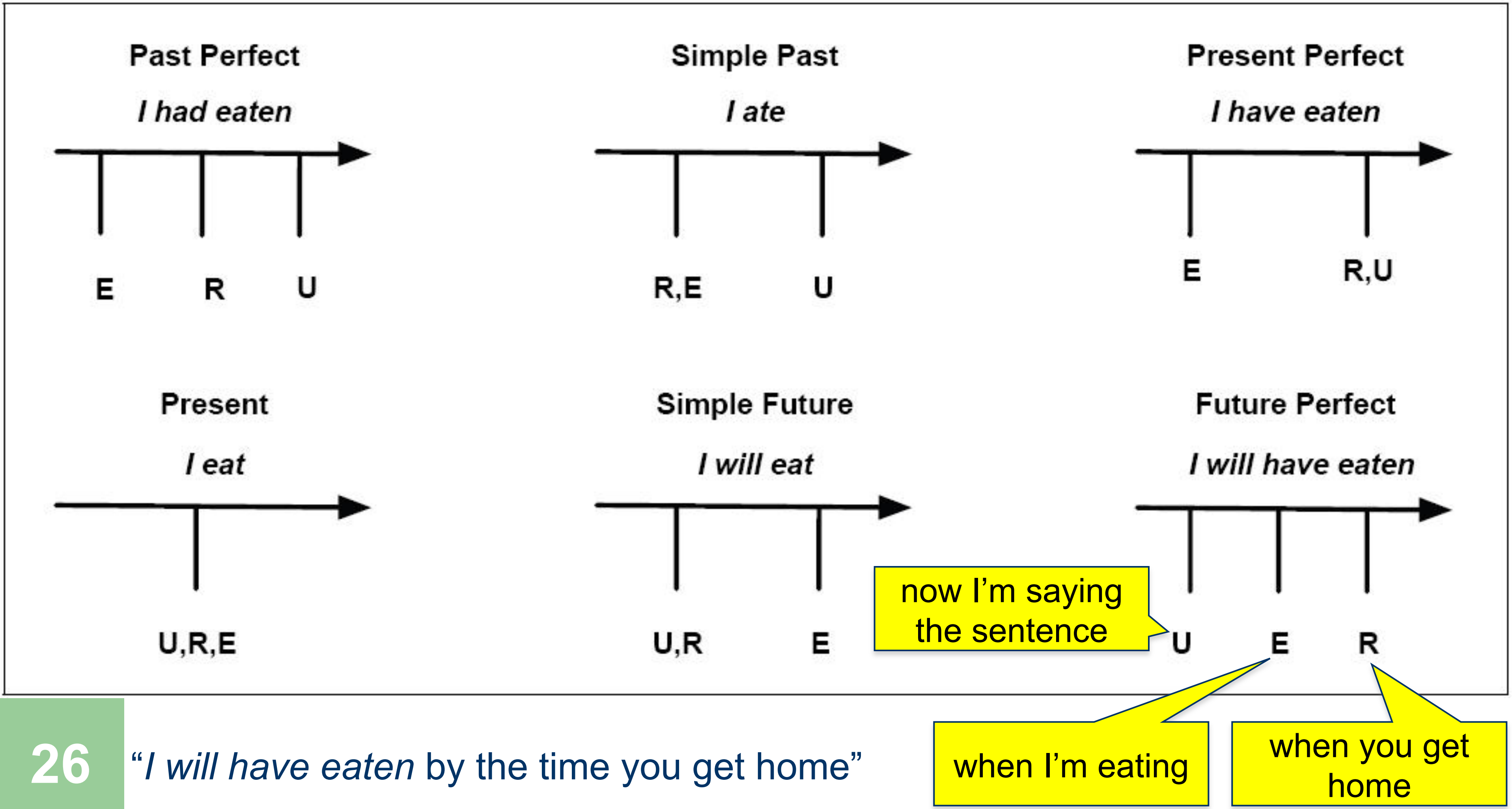


- "I have a car" (→ definiton of the car owner)
  $CarOwner = HumanBeing \sqcap \exists owns\text{-}object.Car$

Temporal logics (tense logic)

- Events are related to the concept of *time*
  - Time points or time intervals
  - Temporal logics try to represent just that
- Time of the event (E)
- Time of the utterance (U)
- Time of the *reference point* (R)
  - "When Mary's flight departed, I had eaten lunch"
  - Lunch occurred before flight departure

English tenses (Reichenbach, 1947)



"*I will have eaten* by the time you get home"

Temporal logics (Linear temporal Logic)

- "If you have passport and ticket, you can board the flight"
- $\Box((\neg passport \land \neg ticket) \Rightarrow \bigcirc \neg board\_flight)$

$\Box$ = is true in all future moments

$\bigcirc$ = is true in the next future moment

Frame-based representation

- Represents objects as feature-structures
  - Features (slots)
  - Values (fillers)
  - Values can, in turn, be a frame
- "I believe Mary ate British food" (NB "I" is the Speaker)

$$
\begin{bmatrix}
\textsc{Believing} & \\
\textsc{Believer} & \textsc{Speaker} \\
\textsc{Believed} & \begin{bmatrix}
\textsc{Eating} & \\
\textsc{Eater} & \textsc{Mary} \\
\textsc{Eaten} & \textsc{BritishFood}
\end{bmatrix}
\end{bmatrix}
$$

Representing meanings: Tools

Implementations

- Production-rule systems
- PROLOG
- Ontologies

- Both production-rule systems and PROLOG implement a subset of FOL (a decidable subset)
- Most common ontologies are based on DL

Production rules systems

- Demo

- Implement forward chaining inference
  - CLIPS, Jess, Drools, …

```
(deffacts startup "Refrigerator Status"
   (refrigerator 1 light on)
   (refrigerator 1 door open)
   (refrigerator 2 door open))
```

Fact list

(refrigerator 1 light on)
(refrigerator 1 door open)
(refrigerator 2 door open)

$$\forall x\ Refrigerator(x, open) \wedge Refrigerator(x, light\ on) \Rightarrow$$
$$Refrigerator(x, food\ spoiled)$$

```
(defrule example-rule "example"
(refrigerator ?x light on)
(refrigerator ?x door open)
=>
(assert (refrigerator ?x food spoiled)))
```

Fact list

(refrigerator 1 light on)
(refrigerator 1 door open)
(refrigerator 2 door open)
(refrigerator 1 food spoiled)

The PROLOG language

- Demo

- Implements backward chaining inference
- ***A lot*** of different implementations exist…

KB

```
mortal(X) :- human(X).
human(socrates).
```

$$\frac{Human(socrates) \qquad \forall x\; Human(x) \Rightarrow Mortal(x)}{Mortal(socrates)}$$

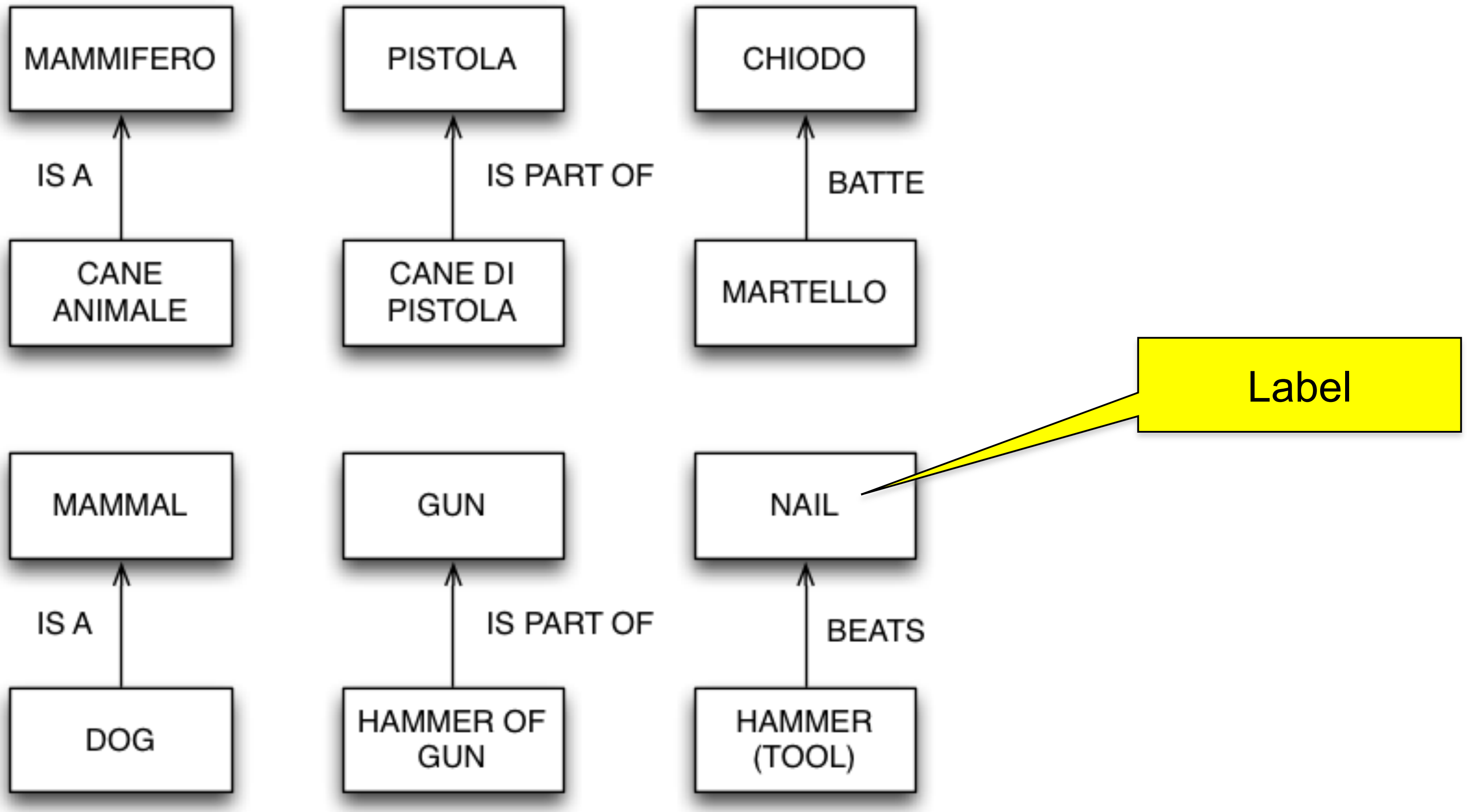
```
?- mortal(socrates).
yes
```

goal: search KB for `mortal(socrates)` → *not found*
matches consequent: `mortal(X=socrates) :- human(X=socrates)`
new goal: search KB for: `human(socrates)` → *found*
answer: `yes`

Ontologies

- *Formal definition of concepts belonging to a domain*
- Abstract definition of concepts: does not depend on the language
- Most ontologies are composed of:
  - Classes (e.g. *Wine*, *Winery*)
  - Individuals (e.g. *champagne*)
  - Attributes (e.g. *price*)
  - Relationships (e.g. Winery *produces* Wine)
- Ontologies described here rely on DL

Ontologies are language independent



Ontologies: characteristics (1/2)

- Class (e.g. **`Wine`**):
  - A set of elements; a concept; a type (i.e., a class)
- Individual (e.g. **`Nero_Avola`**):
  - An element; a concept; an instance (i.e., an object)
- Attribute (e.g. **`price`**):
  - A simple property; a class field
  - Has a primitive data type (e.g. **`string`**)
- Data type (e.g. **`string`**)
  - A primitive data type

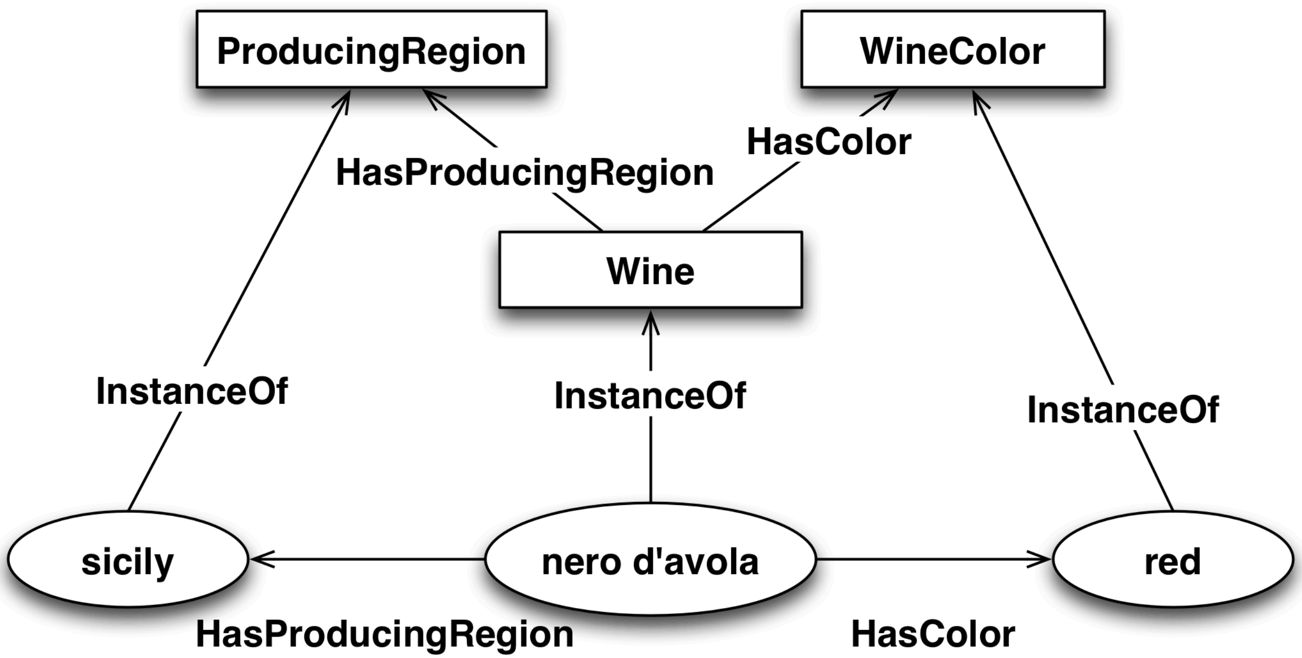
Ontologies: characteristics (2/2)

- Relationship (e.g. **`Winery Produces Wine`**)
  - Semantic characterization of a concept; relationships among classes or among individuals
- Restriction
  - For example, on attribute values (e.g., “>0”)
- Logical rules further specifying the domain

$hasParent(?x1,?x2) \wedge hasBrother(?x2,?x3) \rightarrow hasUncle(?x1,?x3)$

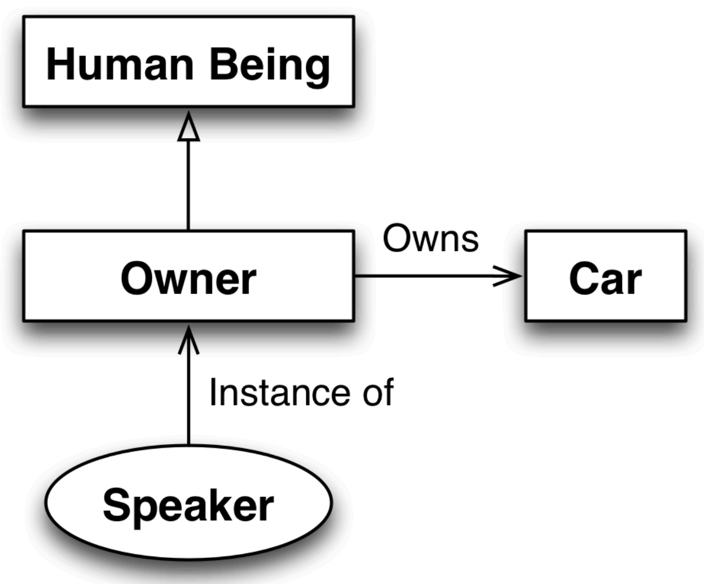
Ontologies as a graph

- Ontologies can be used to represent KB



Ontologies as a graph

- Ontologies can be used to represent sentences
  - “I have a car”
    (“I” is the Speaker)



OWL (Web Ontology Language)

- W3C specification
- Based on RDF (triple: *subject predicate object*)
- Three languages:
  - OWL Lite: taxomonies and simple constraints
  - OWL DL: permits to represent a Description Logic (decidable subset of First-Order Logic)
  - OWL Full: Higher order logics (not decidable)
- Inference rules: SWRL
- Query: SPARQL

Open and closed world

- The “closed world” assumption (e.g. SQL):
  - Any statement that is not known to be true (i.e. a tuple/fact not found in DB) is false (*negation as failure*)
  - The system is assumed to have complete knowledge
  - PROLOG, production rules engines use “closed world”
- The “open world” assumption (FOL, OWL):
  - Any statement that is not known to be true is… unknown
  - The system does not have enough info to decide
  - Pros and cons:

    − limits the kinds of inferences an agent can make

    \+ represents the notion that no single agent has complete knowledge

Open and closed world

- E.g. propositions in KB:
  - “Giovanni is an architect”
  - “Giovanni is not a physicist”
- Query: “Is Giovanni an engineer?”
  - Closed world answer: *no* (proposition not found in KB)
  - Open world answer: *unknown*
- Query: “Is Giovanni a physicist?”
  - Closed world answer: *no* (proposition not found in KB)
  - Open world answer: *no* (negated proposition found)

The OWL language

- A set of *axioms*
- The language:
  - Class
  - Property
    - Object property: relationship
    - Datatype property: attribute
  - *subclass*, *disjoint*, *equivalent* relationships
  - Restrictions on properties (type, cardinality)
  - Characteristics of object properties (transitive, …)
  - Individuals
- Domain $\xrightarrow{\text{Object/Datatype Property}}$ Range

Example (1/3)

```
<owl:Class rdf:ID="ConsumableThing"/>

<owl:Class rdf:ID="PotableLiquid">
  <rdfs:subClassOf rdf:resource="#ConsumableThing" />
  …
</owl:Class>

<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="#PotableLiquid"/>
  <rdfs:label xml:lang="en">wine</rdfs:label>
  <rdfs:label xml:lang="fr">vin</rdfs:label>
  …
</owl:Class>
```

Example (2/3)

```
<owl:Class rdf:ID="Grape">
  …
</owl:Class>

<owl:Class rdf:ID="WineGrape">
  <rdfs:subClassOf rdf:resource="#Grape" />
  …
</owl:Class>

<WineGrape rdf:ID="ChardonnayGrape">
  …
</WineGrape >
```

Example (3/3)

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about=”#year">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdf:range rdf:resource=”&xsd:nonNegativeInteger"/>
</owl:DatatypeProperty>

<Wine rdf:ID="LindemansBin05Chardonnay">
  <madeFromGrape rdf:resource="#ChardonnayGrape" />
  <year rdf:datatype=“&xsd:nonNegativeInteger”>2005</year>
</Wine>
```

Query: SPARQL

- Query language on RDF (W3C standard)
  - Matching on triples

```
PREFIX food: <http://somewhere/example#>
SELECT ?wine
WHERE
{
   ?wine food:madeFromGrape food:ChardonnayGrape .
   ?wine food:year ?year .
   FILTER (?year > 2004)
}
```

```
---------------------------------
| wine                          |
=================================
| LindemansBin05Chardonnay      |
---------------------------------
```

Reasoning: SWRL

- Inference rules: SWRL (W3C standard)

hasParent(?x1,?x2) ^ hasBrother(?x2,?x3) -> hasUncle(?x1,?x3)

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#example1"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom  swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hasBrother">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

Top & domain ontology

- Usually, ontologies describe a specific domain
- Top ontologies: describe basic concepts (e.g. time, space, …)
  - LKIF
- Some ontologies try to represent a vast quantity of fundamental human knowledge
  - CYC

Ontologies and NLP

- Enable searching for concepts
  - Abstract and language independent
  - Search through ontology navigation
  - Search through key-words (how can we map words and concepts?)
  - Reasoning (search for sub-classes, part-of classes, individuals, …, of a given concept)
  - Question answering
- Calculating document similarity (distance)
- Mapping ontologies
- …

Tools for OWL

- JENA: a free and open source Java framework for processing RDF data (OWL is based on RDF)
  - Ontology navigation
  - Query (SPARQL)
  - Reasoning (embedded; third-party engines; e.g., Pellet)
- Protégé: An ontology editor
  - OWL, SPARQL
  - Extensible via plug-ins

REFERENCES

Books

- D. Jurafsky, J. H. Martin, *Speech and Language Processing*, Prentice Hall.
- C. D. Manning, H. Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press.
- S. Bird, E. Klein, and Edward Loper, *Natural Language Processing with Python*, O'Reilly
- S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall.

On semantics

- OWL
  - http://www.w3.org/2004/OWL/
- SPARQL
  - http://www.w3.org/TR/rdf-sparql-query/
- LKIF
  - http://www.estrellaproject.org/lkif-core/
- Cyc
  - http://www.cyc.com/
- Protégé
  - http://protege.stanford.edu/
  - http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library

On semantics

- Inference engine Pellet
  - http://clarkparsia.com/pellet
- Inference rule language SWRL
  - http://www.w3.org/Submission/SWRL/
- Quary language SQWRL:
  - http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL
- OWL Jena
  - http://jena.sourceforge.net/
- Production rule engines:
  - http://www.jessrules.com/
  - http://clipsrules.sourceforge.net/
  - http://www.jboss.org/drools

On semantics

- WordNet
  - http://clarkparsia.com/pellet
- MultiWordNet
  - http://multiwordnet.fbk.eu
- EuroWordNet
  - http://www.illc.uva.nl/EuroWordNet/
- Global WordNet
  - http://globalwordnet.org

General references

- Tools:
  - http://www-nlp.stanford.edu/links/statnlp.html
  - http://www.comp.nus.edu.sg/~rpnlpir/
- General resources:
  - http://www-nlp.stanford.edu/links/linguistics.html
  - http://www.math.tau.ac.il/~shimsh/text_domain.html

NLP in Italy

- ILC-CNR (Pisa, Genova)
  - http://www.ilc.cnr.it/indexflash.html
  - http://www.ge.ilc.cnr.it/
- ISTC-CNR (Roma, Padova, Trento)
  - http://www.istc.cnr.it/
  - http://www.loa-cnr.it/
  - http://www.pd.istc.cnr.it/home.htm
- ITC-IRST (Trento)
  - http://tcc.itc.it/index.html
- Università degli Studi di Torino
  - http://www.di.unito.it/~gull/
  - http://www.di.unito.it/~tutreeb/
- ForumTAL
  - http://www.forumtal.it/