ESSLLI 2012

Ontology-based Interpretation of Natural Language

August 15, 2012

Ontology lexica and automatic grammar generation

Philipp Cimiano · Christina Unger

Semantic Computing Group

CITEC, Bielefeld University

These slides are not the orginal ones produced by prof. Philipp Cimiano.

They have been edited by Manuel Fiorelli (fiorelli@info.uniroma2.it).

Motivation

So far we have:

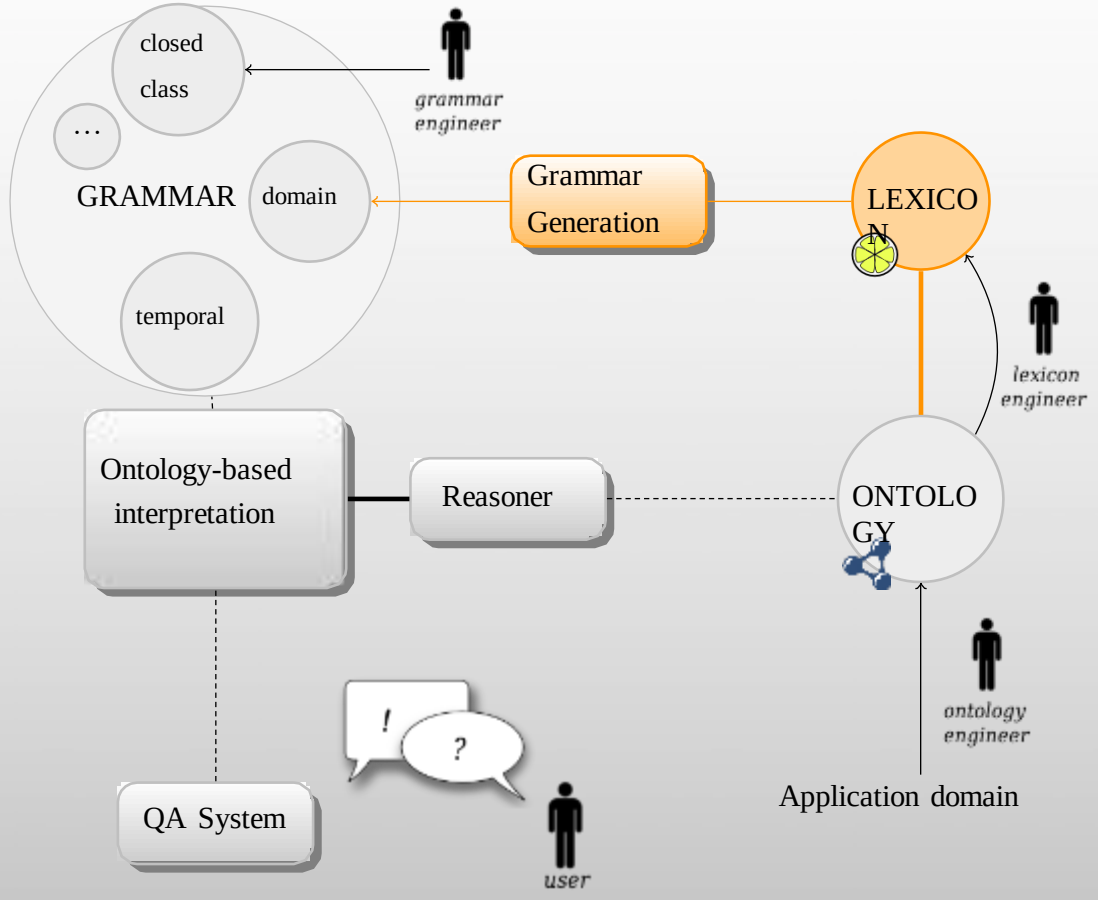
- Ontologies
- LTAG/DUDES grammars aligned to the ontology

But…

- Who is going to write those grammars for new domains?
- What if we want to produce different grammar formats (e.g. CG, HPSG, GF, etc.)?

Goal: automatize the task of grammar generation

Today


closed
class
...
GRAMMAR
domain
temporal
grammar
engineer
Grammar
Generation
LEXICO
N
lexicon
engineer
Ontology-based
interpretation
Reasoner
ONTOLO
GY
ontology
engineer
Application domain
!
?
QA System
user

Universität Bielefeld

Outline

lemon: A model for the lexicon-ontology interface

Automatic grammar generation

- Verbs
- Proper names
- Nouns
- Prepositions
- Adjectives

CITEC

Universität Bielefeld

Outline

lemon: A model for the lexicon-ontology interface

Automatic grammar generation

- Verbs
- Proper names
- Nouns
- Prepositions
- Adjectives

CITEC

The lexicon-ontology interface

In order to generate grammars for a given ontology, we need to enrich the ontology with lexical and linguistic information.

The lexicon-ontology interface

In order to generate grammars for a given ontology, we need to enrich the ontology with lexical and linguistic information.

An ontology lexicon specifies how ontology concepts correspond to natural language expressions. This can support:

- Interpretation (knowing how natural language expressions should be interpreted with respect to a given ontology)
- Generation (knowing how ontology concepts can be verbalized)

Semantics by reference

Semantics by reference (McCrae et al. 2012)

Lexicon and ontology are clearly separated. The meaning of lexical entries is specified by pointing to elements in the ontology.

- That is, the ontology can live without the lexicon, while the lexicon depends on a given ontology.
- One ontology can be connected to several lexica (e.g. for different languages).

lemon

lemon (**Le**xicon **m**odel for **on**tologies) is a meta-model for describing lexica with RDF.

Most importantly, lemon is agnostic w.r.t linguistic categories.

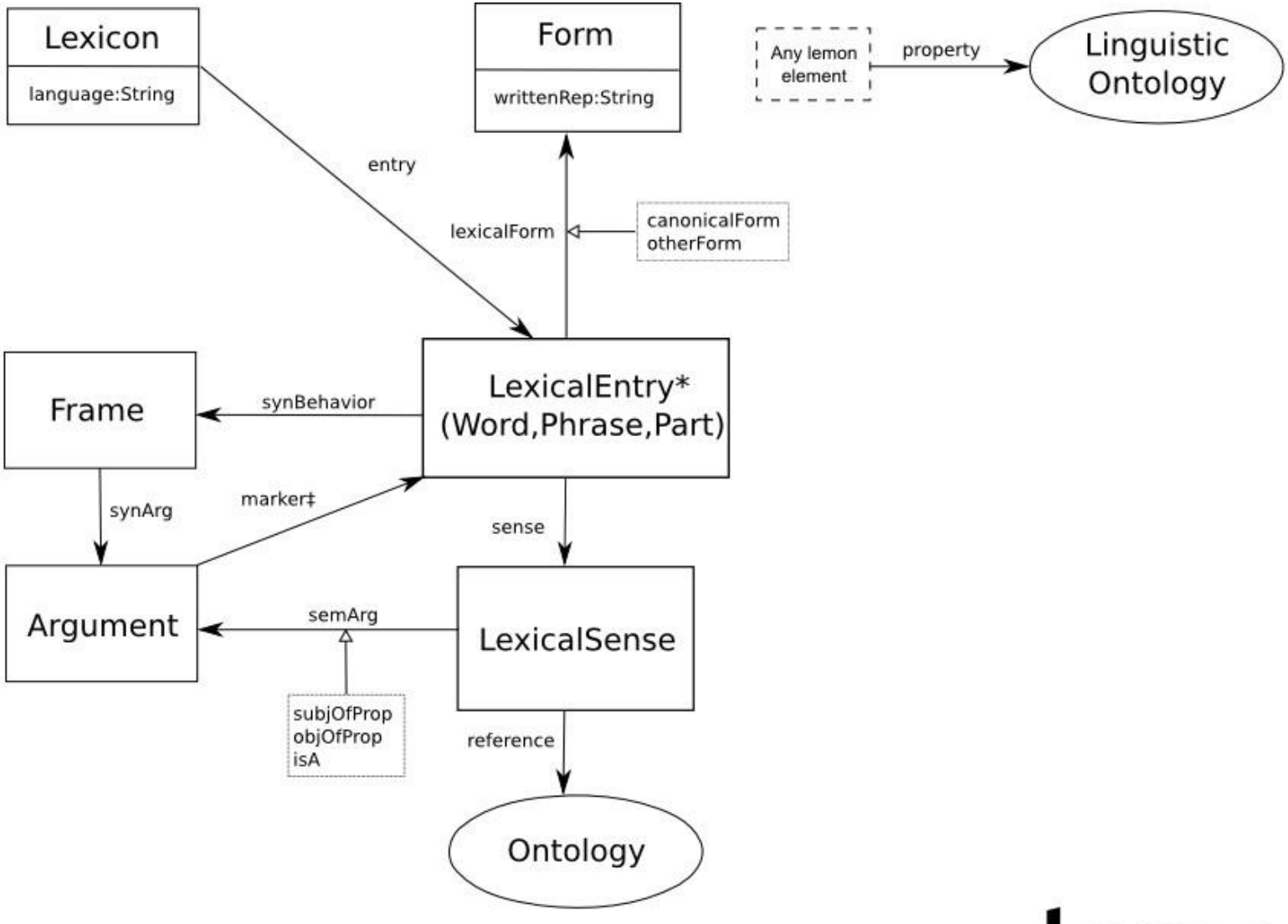
- It has general concepts like `lexical entry'.
- It does not have specific concepts like `noun', 'intransitive verb', `singular', etc.

lemon is developed in collaboration between CITEC (Bielefeld), DERI (Galway), UPM (Madrid), DFKI (Saarbrücken), and TU Delft.

Universität Bielefeld

The lemon model

CITEC



* LexicalEntry has three subclasses: Word, Phrase, Part
† PhraseTerminals are Arguments or Components
‡ marker can also refer to linguistic ontology

lemon

Lemon core concepts

- **Lexicon**

  The lexicon is itself a resource and has one associated language.

- **Lexical Entry**

  Each entry is a resource and is split into three subclasses: Word, Phrase, Part.

- **Form**

  Each entry has a number of forms. Each form has a number of written representations.

- **Lexical Sense**

  Each entry has a number of senses. Each sense has a reference pointing to an ontological symbol (entity, class or relation).

Lexica and entries

Entries are indicated with the entry property.

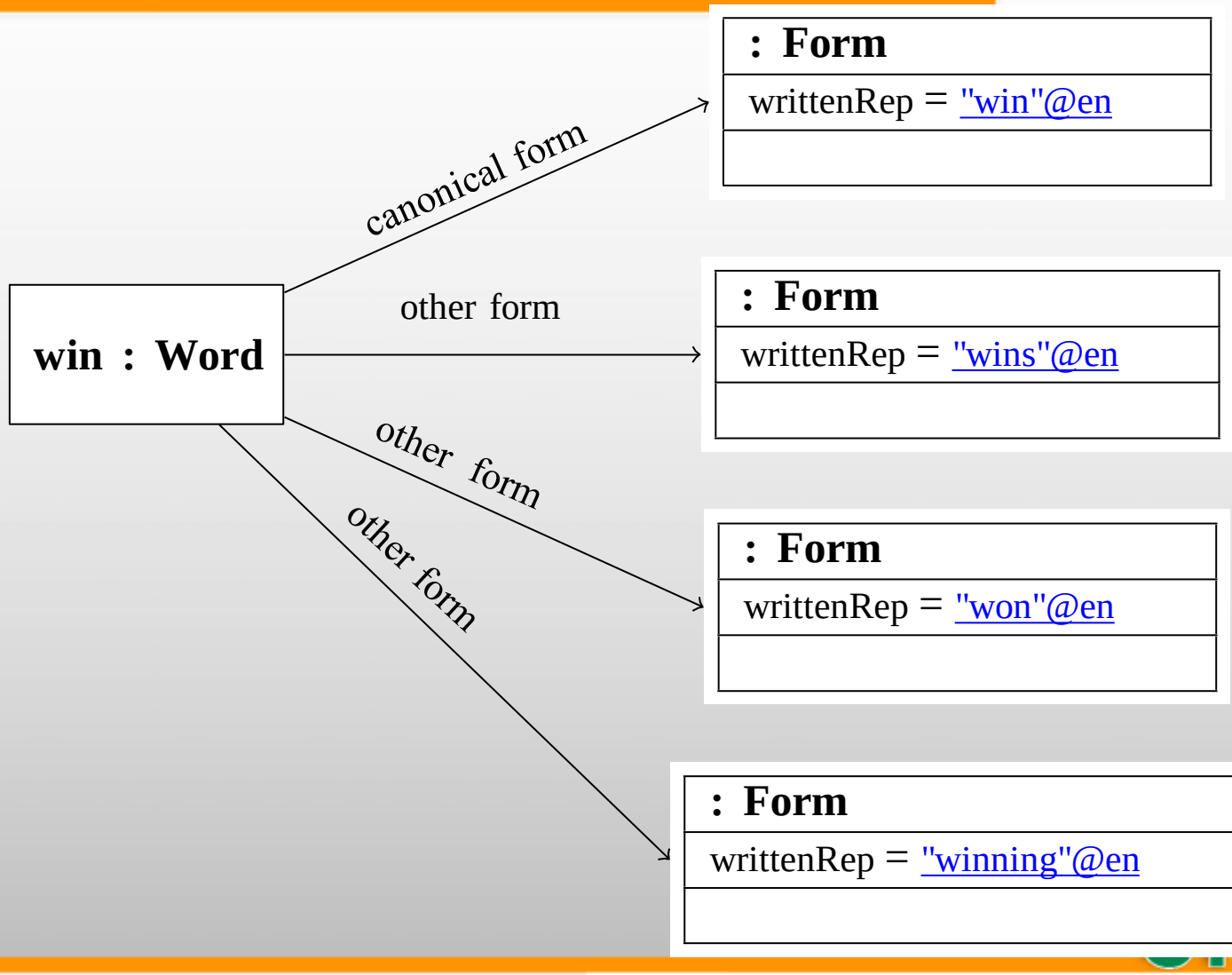
```
1  @prefix : <http://www.example.org/lexicon/> .
2  @prefix lemon: <http://lemon-model.net/lemon#> .
3
4  :lexicon a lemon:Lexicon ;
5              lemon:language "en" ;
6              lemon:entry :team,
7                          :match,
8                          :goal,
9                          :win,
10                           ... .
```

An ISO 639 language code

Forms

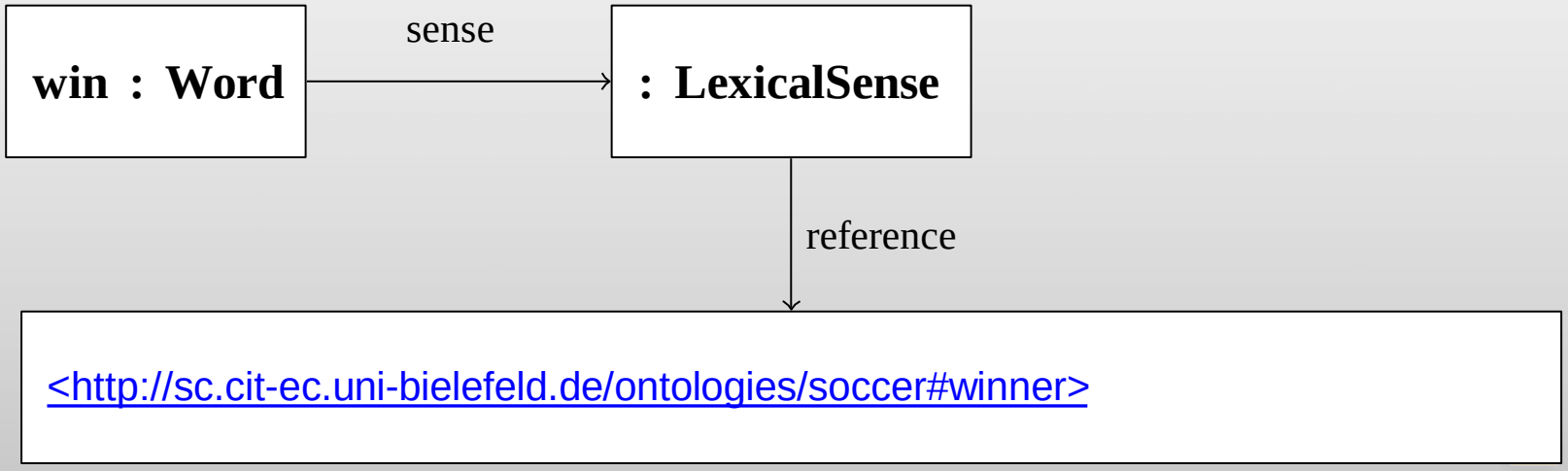
```
@prefix : <http://www.example.org/lexicon/> .
@prefix lemon: <http://lemon-model.net/lemon#> .

:team a lemon:Word;
        lemon:canonicalForm [ lemon:writtenRep "team"@en ] ;
        lemon:otherForm     [ lemon:writtenRep "teams"@en ] .

:win    a lemon:Word ;
        lemon:canonicalForm [ lemon:writtenRep "win"@en ] ;
        lemon:otherForm     [ lemon:writtenRep "wins"@en ] ;
        lemon:otherForm     [ lemon:writtenRep "won"@en ] ;
        lemon:otherForm     [ lemon:writtenRep "winning"@en ] .
```

Forms


win : Word
canonical form
other form
other form
other form
: Form
writtenRep = "win"@en
: Form
writtenRep = "wins"@en
: Form
writtenRep = "won"@en
: Form
writtenRep = "winning"@en

Senses and references

```
1  :team lemon:sense [ lemon:reference
2        <http://sc.cit-ec.uni-bielefeld.de/ontologies/soccer#Team> ].
3
4  :win lemon:sense [ lemon:reference
5        <http://sc.cit-ec.uni-bielefeld.de/ontologies/soccer#winner> ].
```



Frames

Syntactic arguments: Each word has a subcategorization frame.

- intransitive: X won (X = subject)
- transitive: X won Y (X = subject, Y = direct object)

Semantic arguments: Each ontological relation has a number of arguments.

- <subject> soccer:winner <object> .

Frames

Syntactic arguments: Each word has a subcategorization frame.

- intransitive: X won (X = subject)
- transitive: X won Y (X = subject, Y = direct object)

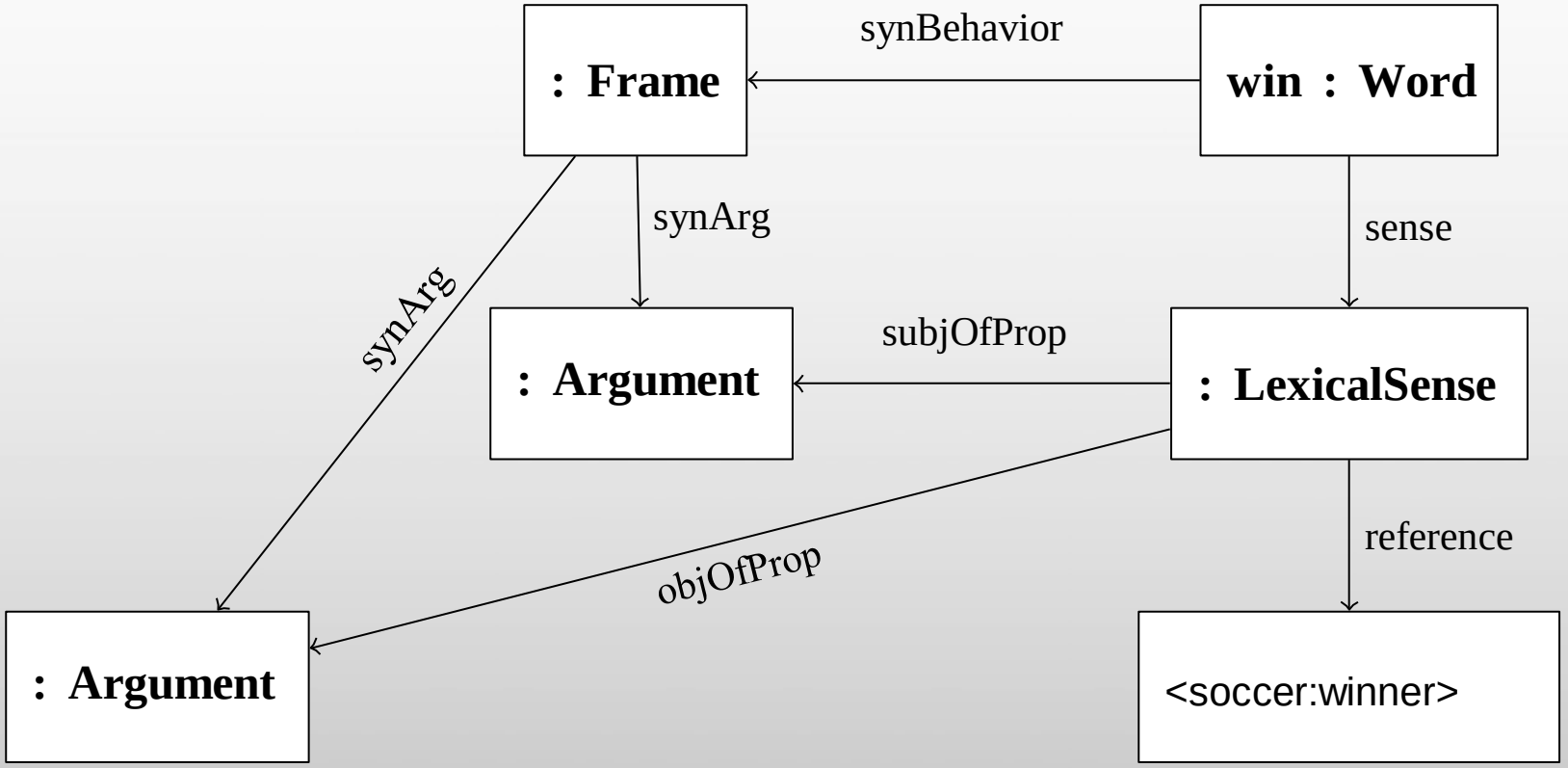
Semantic arguments: Each ontological relation has a number of arguments.

- <subject> soccer:winner <object> .

We also need to specify the **correspondence** between syntactic and semantics arguments.

- Some team won some game.
- <soccer:Match> soccer:winner <soccer:Team> .

Linguistic ontology



Linguistic ontology

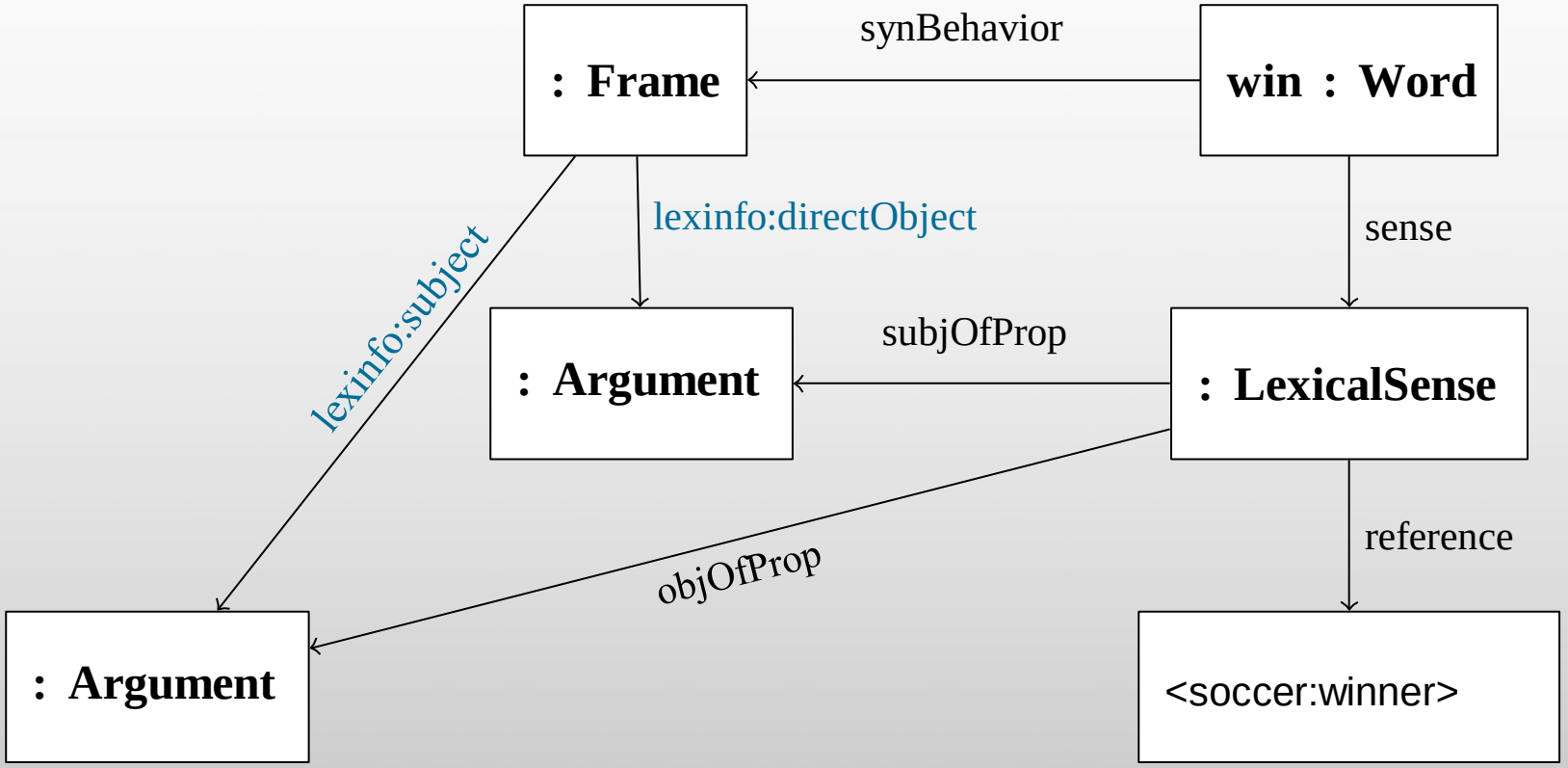
lemon provides a general model but stays agnostic w.r.t. linguistic theories. Hence there is no way to indicate the syntactic roles of the arguments (subject, direct object, indirect object, etc.).

For specifics, lemon thus needs to be extended with a linguistic ontology.

- LexInfo
- ISOcat
- …

We need version 2.0:
http://lexinfo.net/ontology/2.0/lexinfo

Linguistic ontology

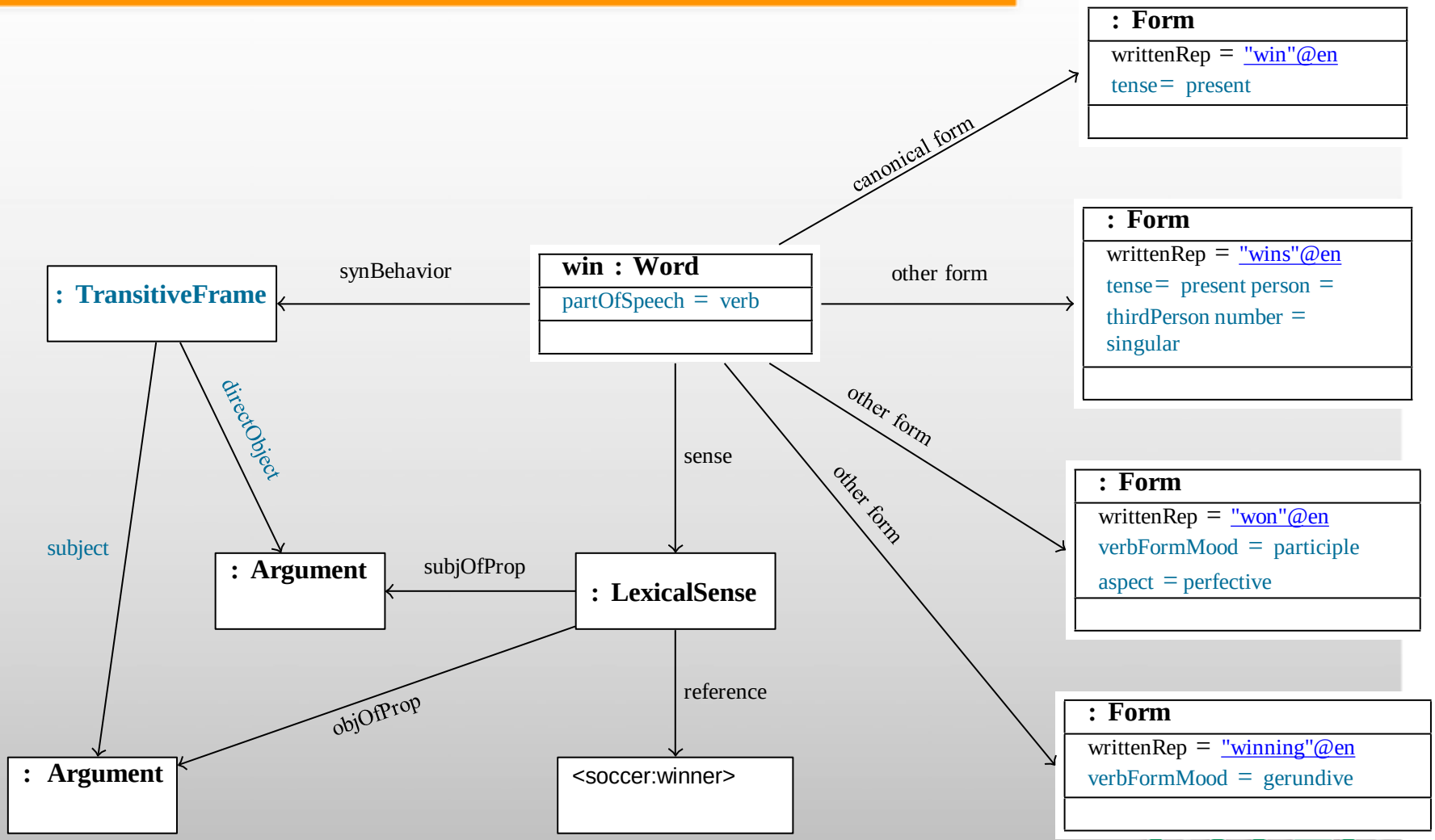


Lexical properties

All LexInfo's lexical properties are subproperties of lemon's property.

- lexinfo:partOfSpeech
  lexinfo:noun, lexinfo:verb, lexinfo:adjective,…
- lexinfo:number
  lexinfo:singular, lexinfo:plural
- lexinfo:person
  lexinfo:firstPerson, lexinfo:secondPerson,
  lexinfo:thirdPerson
- lexinfo:tense lexinfo:present,
  lexinfo:past
- …

A simple lexical entry



Universität Bielefeld

A simple lexical entry

```
:win a lemon:LexicalEntry ;

    lexinfo:partOfSpeech lexinfo:verb ;
    lemon:synBehavior [ rdf:type lexinfo:TransitiveFrame ;
                        lexinfo:subject _:arg1 ;
                        lexinfo:directObject _:arg2 ] ;

    lemon:sense [ lemon:reference
        <http://sc.cit-ec.uni-bielefeld.de/ontologies/soccer#winner> ;
                  lemon:subjOfProp _:arg2 ;
                  lemon:objOfProp _:arg1 ] ;

    lemon:canonicalForm [ lemon:writtenRep "win"@en ;
                          lexinfo:tense lexinfo:present] ;
    lemon:otherForm [ lemon:writtenRep "wins"@en ; ... ] .
```

Most common frames (verbs)

- Intransitive
  - Arguments: subject
  - Example: win
- Transitive
  - Arguments: subject, direct object
  - Example: win (smth)
- Intransitive PP
  - Arguments: subject, prepositional object
  - Example: win against (so)
- Transitive PP
  - Arguments: subject, direct object, prepositional object
  - Example: win (smth) against (so)

Creating lexica with lemon

- lemon source: http://monnetproject.deri.ie/lemonsource/
- Java API: http://www.lemon-model.net/api.html

Outline

lemon: A model for the lexicon-ontology interface

Automatic grammar generation

- Verbs
- Proper names
- Nouns
- Prepositions
- Adjectives

Generating grammars from lexica

For every lexicon entry:

Based on the part of speech and syntactic frame…

- intransitive verb (with PP)
- transitive verb (with PP)
- noun (with PP)
- adjective

…extract relevant properties (word forms, required arguments, semantic restrictions, etc.) and build corresponding grammar entries.

Universität Bielefeld

Outline

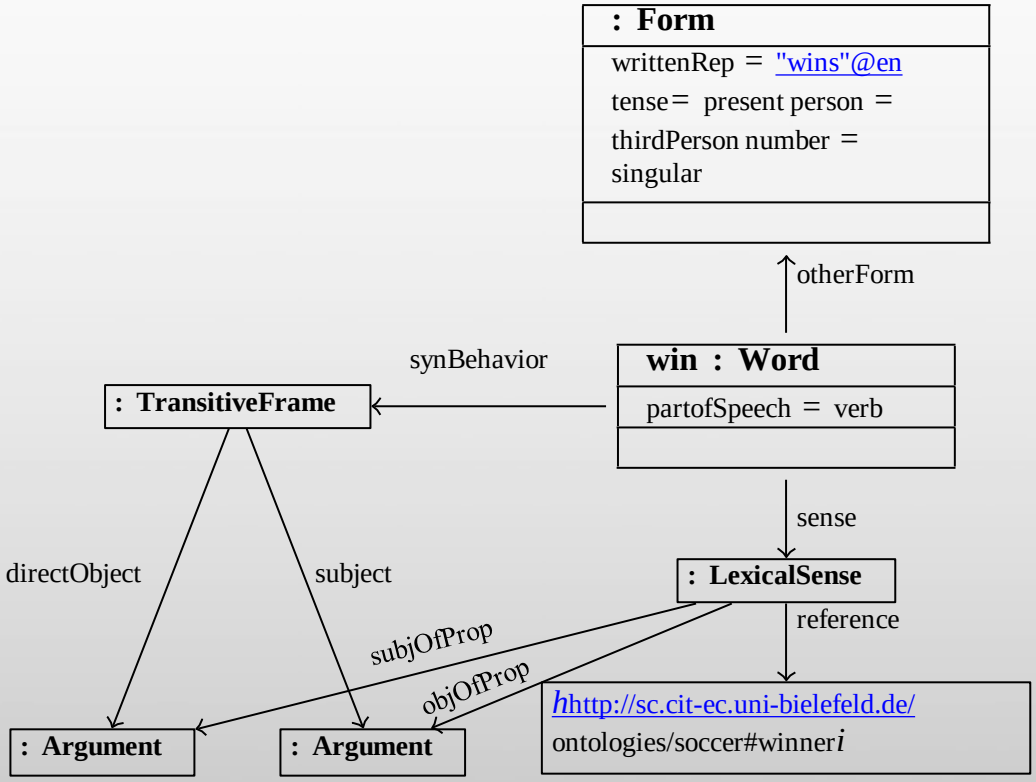
lemon: A model for the lexicon-ontology interface

Automatic grammar generation

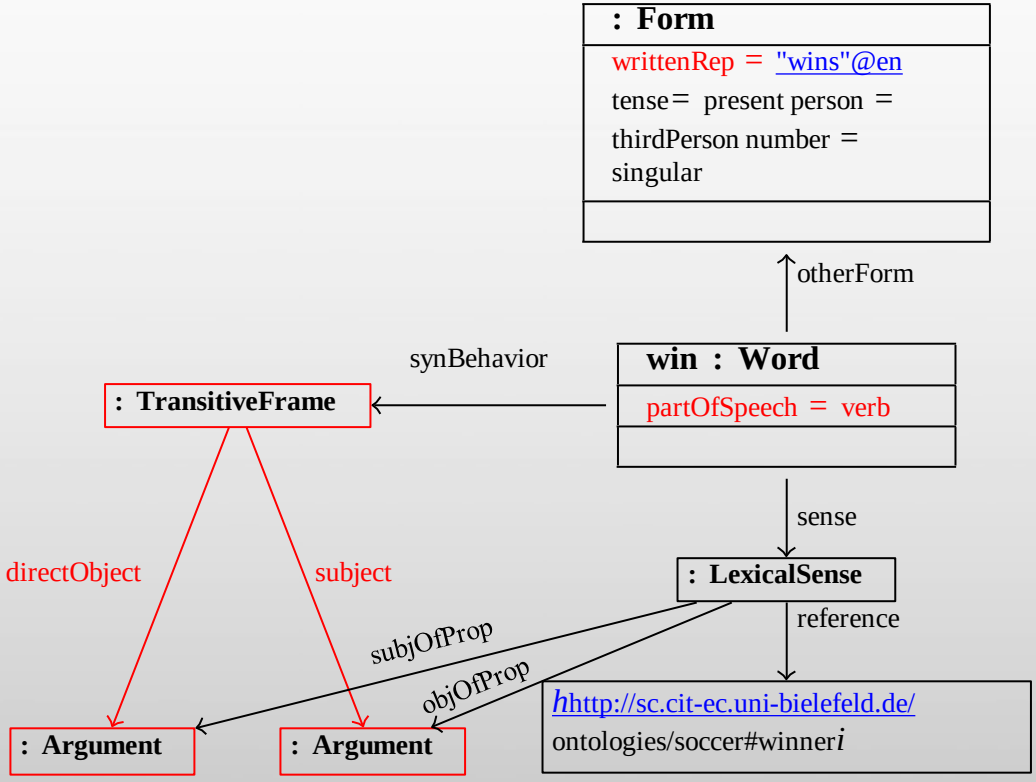
- Verbs
- Proper names
- Nouns
- Prepositions
- Adjectives

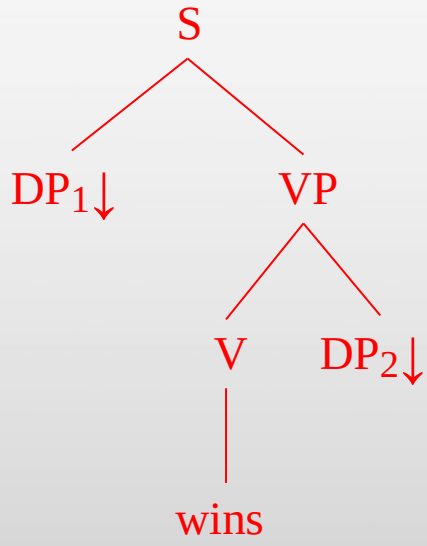
CITEC

Example: to win

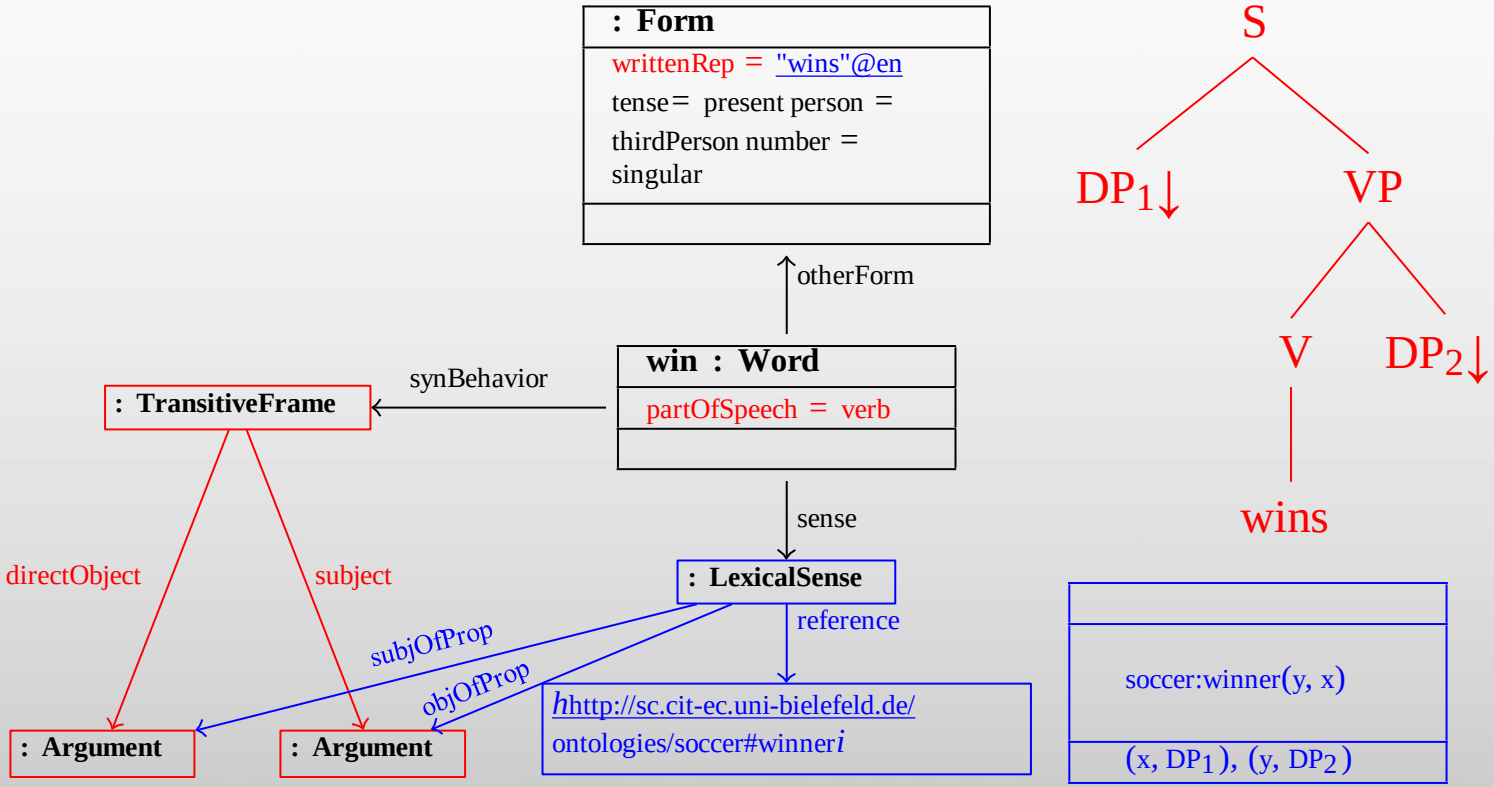


Example: to win



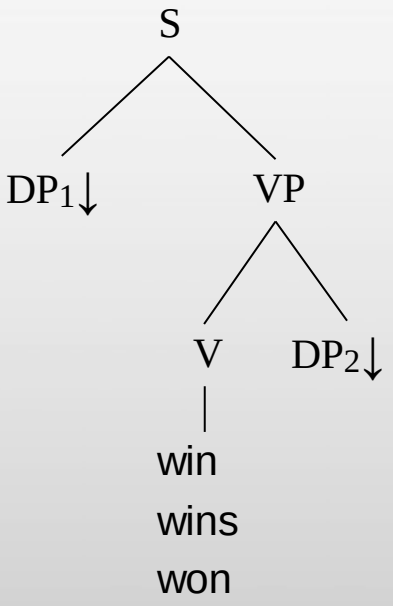


Example: to win



Grammar entries for transitive verbs

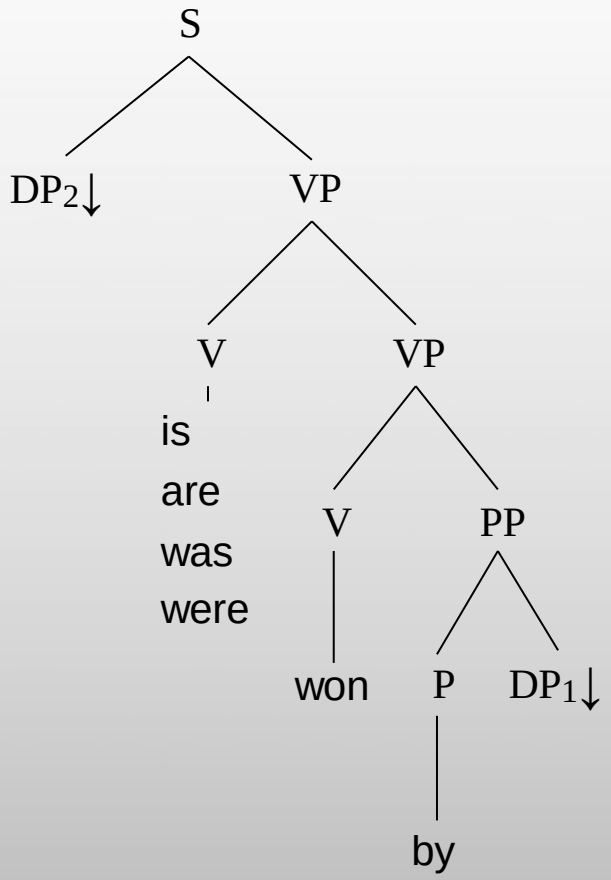
Present (3P singular, others) and past:



soccer:winner(y, x)
(x, DP_1), (y, DP_2)

Grammar entries for transitive verbs

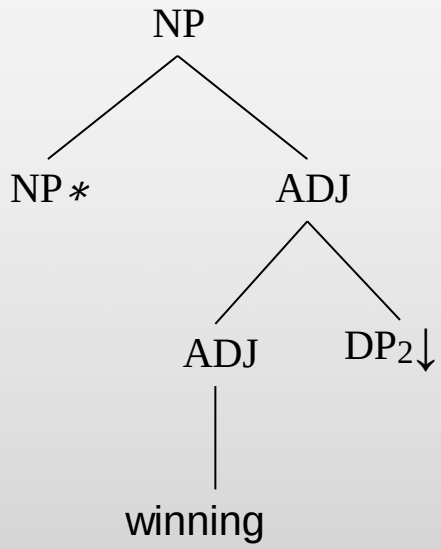
Passive (using past participle):



soccer:winner(y, x)
(x, DP_1), (y, DP_2)

Grammar entries for transitive verbs

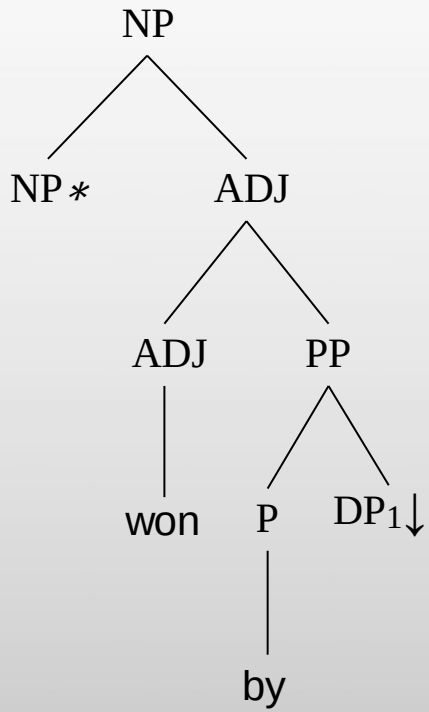
Gerundive:



x	
soccer:winner(y, x)	
(y, DP_2)	

Grammar entries for transitive verbs

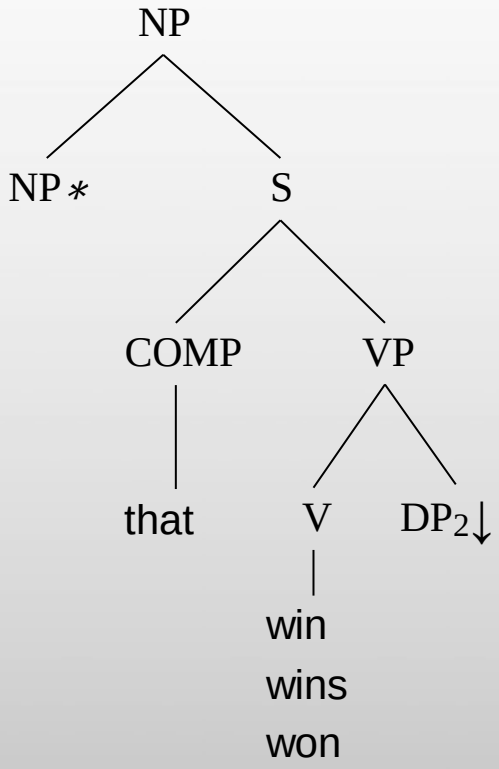
Gerundive:



y
soccer:winner(y, x)
(x, DP_1)

Grammar entries for transitive verbs

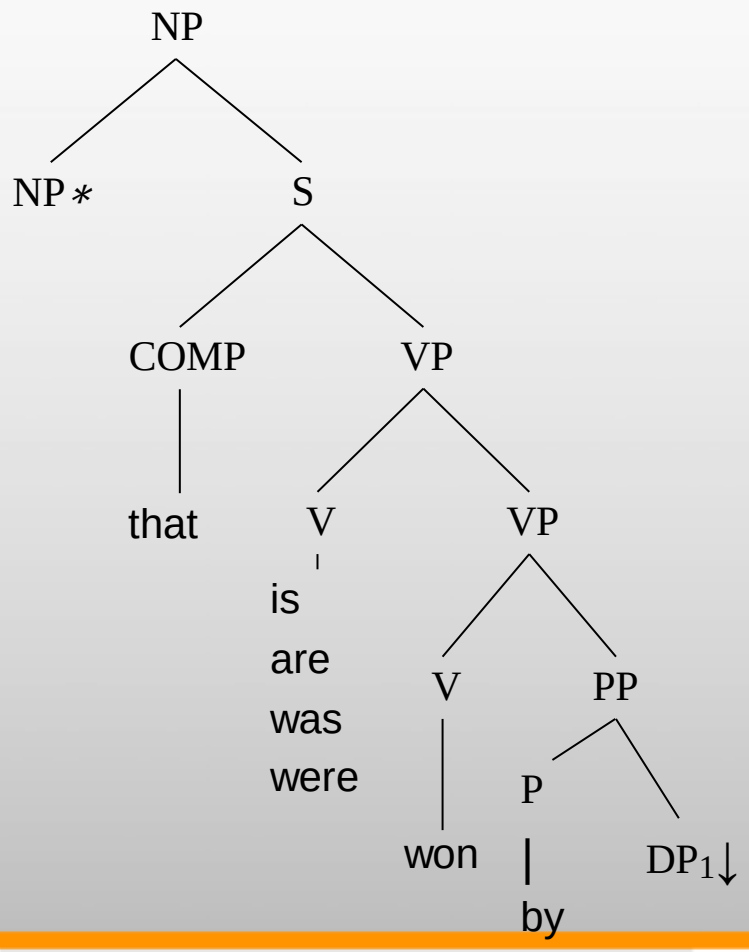
Relative clauses:



x
soccer:winner(y, x)
(y, DP_2)

Grammar entries for transitive verbs

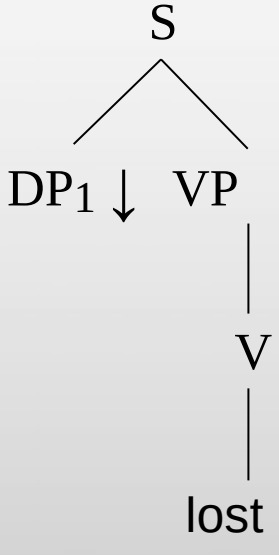
Relative clauses:



y
soccer:winner(y, x)
(x, DP_1)

Intransitive verbs

Example:



y, t
soccer:loser(y, x) time : hasEnd(y, t) t < now
(x, DP1)

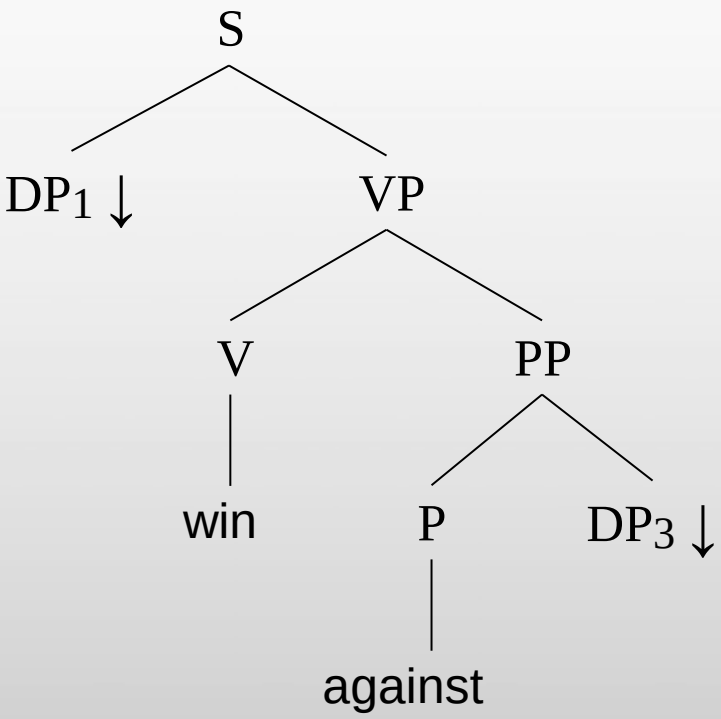
Intransitive and transitive verbs with PP

Examples:

- win against (so.)
- score from (smth.)
- win (smth.) against (so.)
- substitute (so.) for (so.)

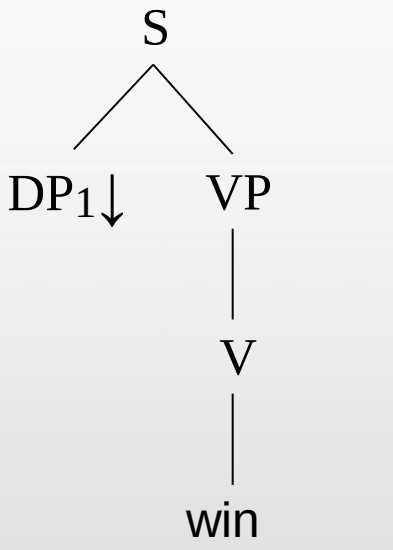
Are prepositional arguments subcategorized or do they adjoin?

Intransitive PP (subcategorized)

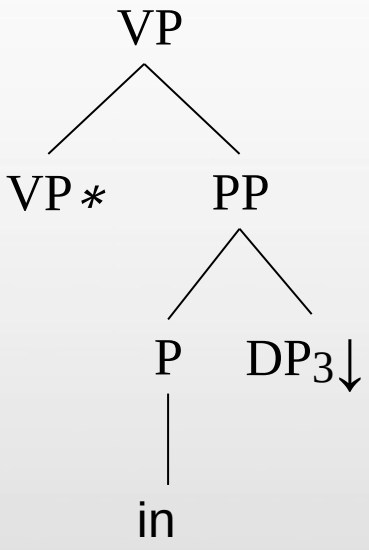


z
soccer:winner(z, x) soccer:loser(z, y)
(x, DP_1), (y, DP_3)

Intransitive PP (adjoining)



z	z
soccer:winner(z, x)	
(x, DP_1)	



z	s
soccer:stadium(z, s) soccer:city(s, y)	
(y, DP_3)	

Universität Bielefeld

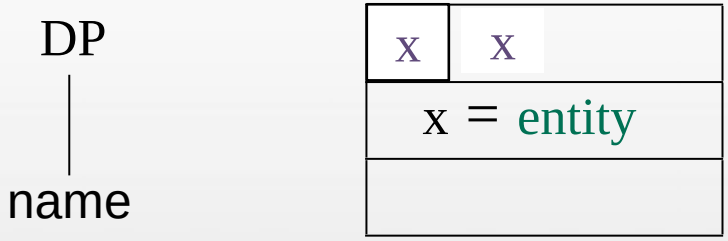
Outline

lemon: A model for the lexicon-ontology interface

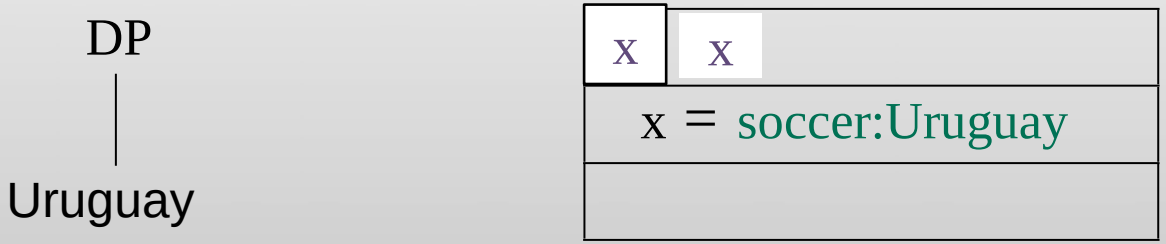
Automatic grammar generation

- Verbs
- Proper names
- Nouns
- Prepositions
- Adjectives

Proper names



In our case, entities are represented by URIs. For example:



Universität Bielefeld

Outline

lemon: A model for the lexicon-ontology interface

Automatic grammar generation

- Verbs
- Proper names
- Nouns
- Prepositions
- Adjectives

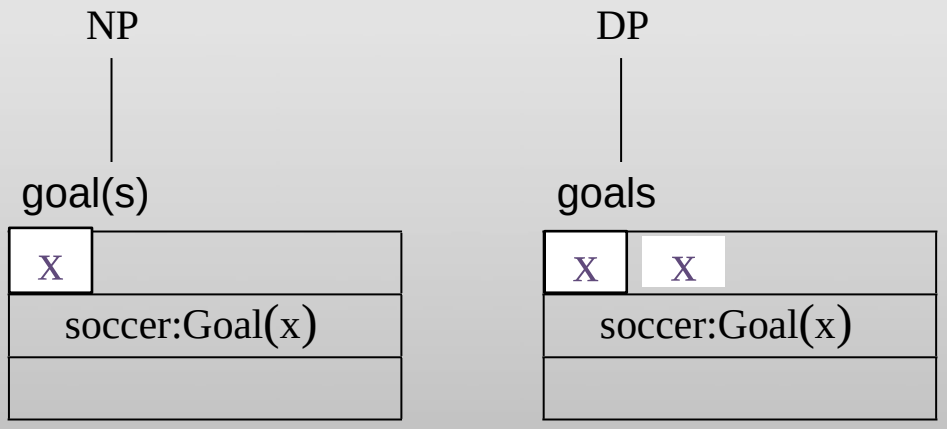
CITEC

Nouns

- semantically simple nouns (usually expressing atomic classes)
  - goal, match, team
- semantically complex nouns (not expressing atomic classes)
  - opener, player, coach
- relational nouns (usually expressing a property)
  - winner, wife, distance, capital

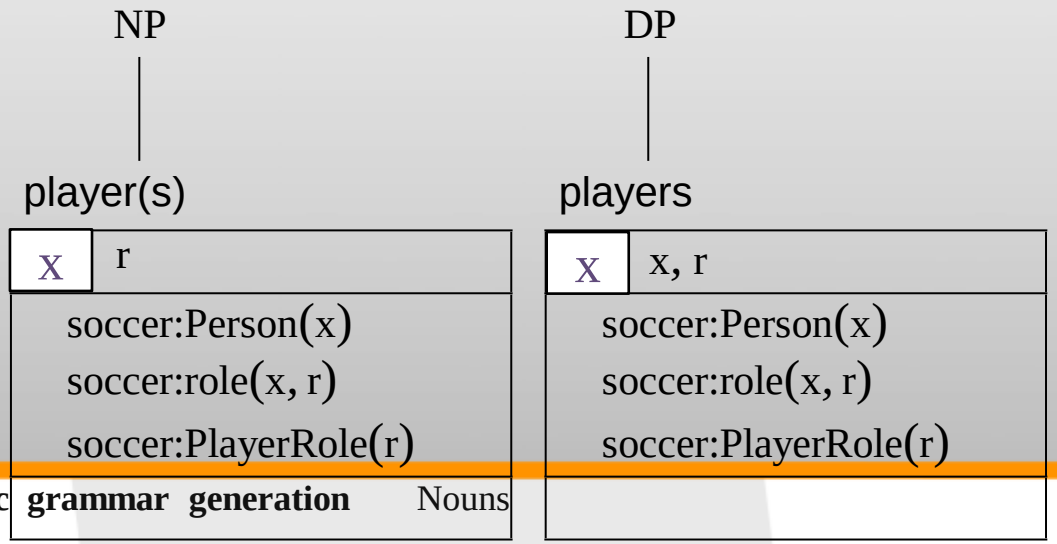
Semantically simple nouns

```
:goal  a lemon:LexicalEntry ;
   lexinfo:partOfSpeech lexinfo:noun ;
   lemon:canonicalForm  [ lemon:writtenRep  "goal"@en ;
                                  lexinfo:number  lexinfo:singular] ;
   lemon:otherForm  [ lemon:writtenRep "goals"@en ;
                               lexinfo:number  lexinfo:plural] ;
   lemon:sense [ lemon:reference soccer:Goal  ] .
```



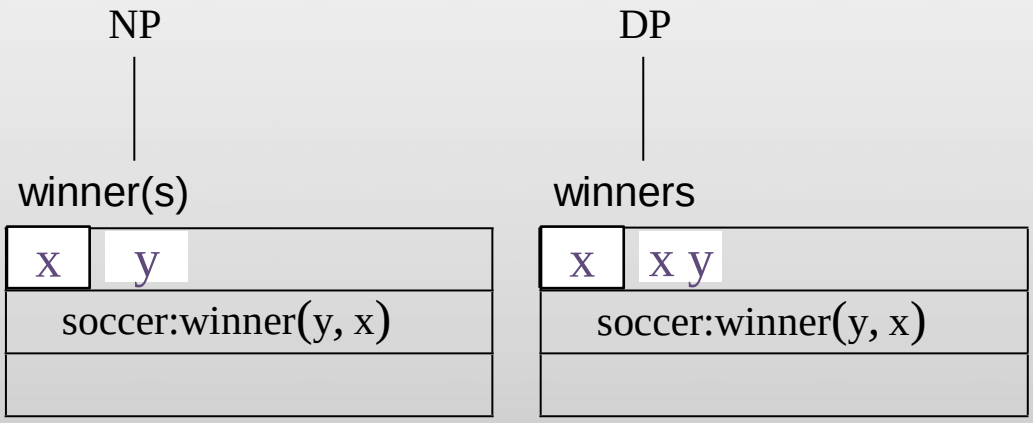
Semantically complex nouns

```
1  :player  a lemon:LexicalEntry ;
2    lexinfo:partOfSpeech lexinfo:noun ;
3    ...
4    lemon:sense
5       [ lemon:subsense [ lemon:reference soccer:Person ] ,
6                          [ lemon:reference soccer:role ;
7                            lemon:propertyRange soccer:PlayerRole ] ] .
```



Relational nouns

```
1 :winner a lemon:LexicalEntry  ;
2    lexinfo:partOfSpeech lexinfo:noun ;
3    ...
4    lemon:sense [ lemon:reference  soccer:winner ] .
```



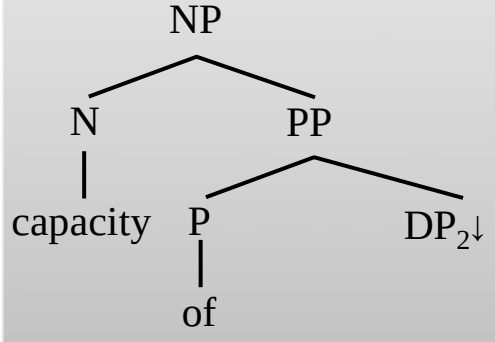
Alternative: Define additional classes

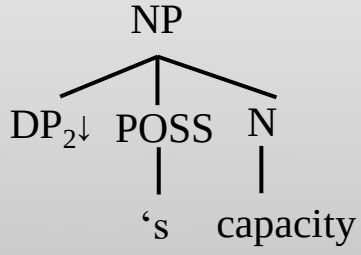
- $\text{Winner} \equiv \exists \text{winner}^{-1}.\top$
- $\text{Player} \equiv \text{Person} \sqcap \exists \text{role}.\text{PlayerRole}$

```
<Declaration><Class IRI="#Player"/></Declaration>
<SubClassOf>
        <Class IRI="#Player"/>
        <Class IRI="#Person"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Player"/>
        <ObjectSomeValuesFrom>
                <ObjectProperty IRI="#role"/>
                <Class IRI="#PlayerRole"/>
        </ObjectSomeValuesFrom>
</SubClassOf>
```

Relational nouns (2)

```
:capacity a lemon:LexicalEntry  ;
          lexinfo:partOfSpeech lexinfo:noun ;
          lemon:canonicalForm [lemon:writtenRep «capacity»@en ;
                               lexinfo:number lexinfo:singular ];
          lemon:synBehavior [ a lexinfo:NounPossessiveFrame ;
                              lexinfo:copulativeArg :y ;
                              lexinfo:possessiveAdjunct :x ] ;
          lemon:sense [ lemon:reference soccer:capacity ;
                        lemon:subjOfProp :x ;
                        lemon:objOfProp :y] ].
```





y
soccer:capacity(x, y)
(x, DP_2)

Universität Bielefeld

Outline

lemon: A model for the lexicon-ontology interface

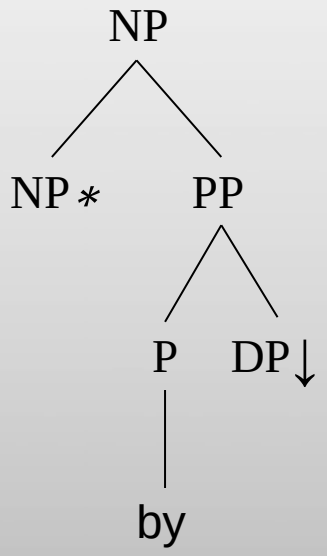
Automatic grammar generation

- Verbs
- Proper names
- Nouns
- Prepositions
- Adjectives

CITEC

Prepositions

```
:by  a lemon:LexicalEntry ;
     lexinfo:partOfSpeech lexinfo:preposition ;
     lemon:canonicalForm  [ lemon:writtenRep  "by"@en ] ;
     lemon:synBehavior  [ lexinfo:complement :y ] ;
     lemon:sense  [ lemon:reference  <soccer:byPlayer> ;
                          lemon:subjOfProp :x  ;
                          lemon:objOfProp        :y ] .
```



x
soccer:byPlayer(x, y)
(y, DP)

Universität Bielefeld

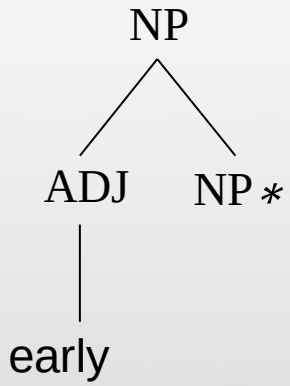
Outline

lemon: A model for the lexicon-ontology interface

Automatic grammar generation

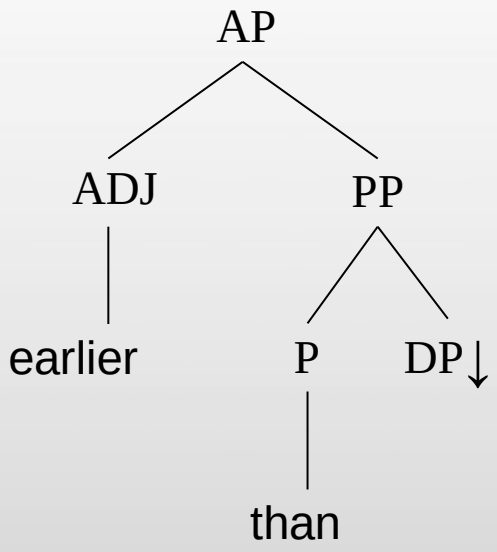
- Verbs
- Proper names
- Nouns
- Prepositions
- Adjectives

Adjectives



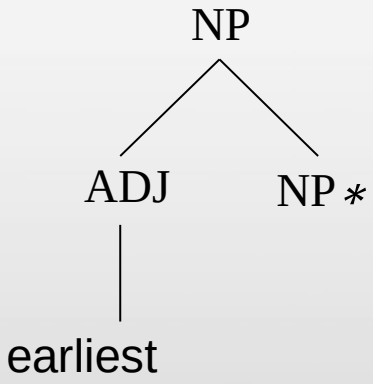
x	m
soccer:atMinute(x, m) m < 10	

Adjectives



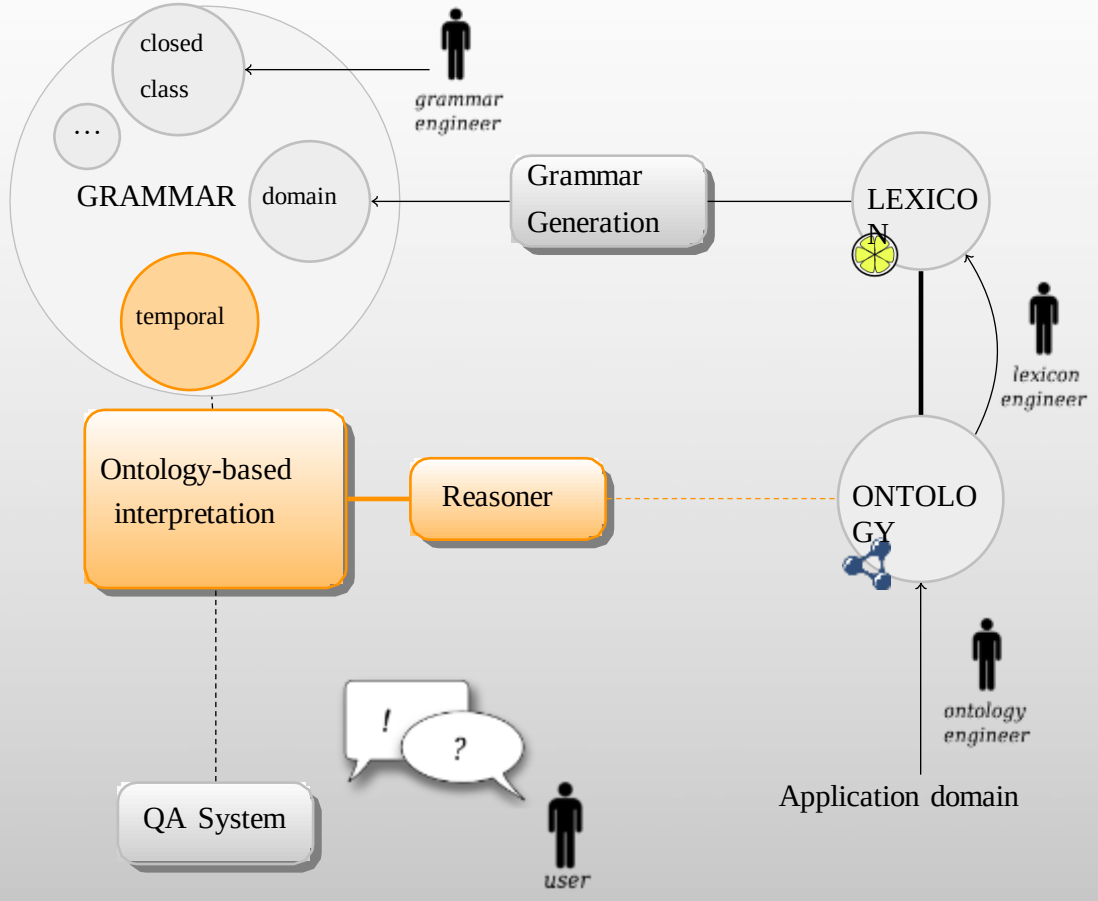
x	m n
soccer:atMinute(x, m) soccer:atMinute(y, n) m < n	
(y, DP)	

Adjectives



x	m
soccer:atMinute(x, m) min(m)	

Tomorrow


closed
class
...
GRAMMAR
domain
temporal
grammar
engineer
Grammar
Generation
LEXICO
N
lexicon
engineer
Ontology-based
interpretation
Reasoner
ONTOLO
GY
ontology
engineer
Application domain
!
?
QA System
user

Useful References

- The Web site of the (Original) Lemon Model (http://lemon-model.net/)
- The lemon Cookbook (http://lemon-model.net/learn/cookbook.html)
- The Web site of LexInfo (http://lexinfo.net/index.html)

The W3C Ontology Lexicon Community Group (Ontolex) (https://www.w3.org/community/ontolex/) has developed a new Lexicon Model for Ontologies (https://www.w3.org/2016/05/ontolex/), which is influenced by the original Lemon Model we have discussed.