

# Prolog & NLP



German Rigau i Claramunt

<http://adimen.si.ehu.es/~rigau>

IXA group

Departamento de Lenguajes y Sistemas Informáticos

UPV/EHU

# Prolog and NLP

- Definite Clause Grammar: DCGs.
  - Introduction.
  - DCG syntax.
  - Examples.

# Introduction

- **Logic Grammars** group a number of language description formalisms, both syntactic and semantic, which cover a very broad spectrum of linguistic phenomena.
- Main characteristics:
  - the use of **unification** as a basic mechanism of composition among constituents.
  - the use of **syntagmatic approach** to grammatical description.

# Introduction

- Logic formalisms apply automatic deduction methods using a set of rules (a **grammar**) and a theorem to be proved (a **sentence**).
- A sentence is list of words that eventually belongs to the language generated by the grammar.
- Thus, a demonstration (a proof) can be seen as a complete analysis of the sentence by the grammar.

# Introduction

- A **definite clause grammar** (DCG) is a way of expressing grammar in a logic programming language such as Prolog.
- They are called definite clause grammars because they represent a grammar as a set of definite clauses in first-order logic.
- DCGs perform our recognition and parsing using top-down depth-first search --- this is the way the Prolog search strategy works.

# Example Prolog

```
sentence(S1,S3) :- noun_phrase(S1,S2), verb_phrase(S2,S3).
```

```
noun_phrase(S1,S3) :- det(S1,S2), noun(S2,S3).
```

```
verb_phrase(S1,S3) :- verb(S1,S2), noun_phrase(S2,S3).
```

```
det([the|X], X).
```

```
det([a|X], X).
```

```
noun([cat|X], X).
```

```
noun([fish|X], X).
```

```
verb([eats|X], X).
```

```
?- sentence([the, cat, eats, a, fish], []).
```

```
?- sentence(X, []).
```

# DCG example

sentence --> noun\_phrase, verb\_phrase.

noun\_phrase --> det, noun.

verb\_phrase --> verb, noun\_phrase.

det --> [the].

det --> [a].

noun --> [cat].

noun --> [fish].

verb --> [eats].

?- sentence([the, cat, eats, a, fish], []).

?- sentence(X, []).

# DCG syntax

- DCGs implements a logic formalism using Prolog.
- A DCG **grammar** consists of one or more DCG rules.
- A DCG **rule** is of the form:
  - `<head> --> <body>`

where

- `<head>` is a non terminal symbol.
  - `<body>` is a list of comma separated **elements**
- An **element** can be:
  - a non terminal symbol.
  - a list of constants or variables between “[” and “]”
  - A list of prolog predicats between “{” and “}”

# DCG syntax

- Example:
  - `verb_rule --> lverb, [Word], {verb(Word)}, rverb.`
  - `verb_rule`, `lverb` and `rverb` are non-terminals of the grammar
  - `[Word]` consumes a word from the input and its value is unified with the variable `Word`.
  - `{verb(Word)}` is a prolog predicate that verifies that `Word` is a verb.

# DCG example

sentence --> noun\_phrase, verb\_phrase .

noun\_phrase --> determiner, noun .

noun\_phrase --> determiner, noun, prepositional\_phrase .

verb\_phrase --> verb .

verb\_phrase --> verb, noun\_phrase .

verb\_phrase --> verb, noun\_phrase, prepositional\_phrase .

prepositional\_phrase --> preposition, noun\_phrase .

noun --> [student] ; [professor] ; [book] ; [university] ; [lesson] ; [glasses].

determiner --> [a] ; [the] .

verb --> [taught] ; [learned] ; [read] ; [studied] ; [saw].

preposition --> [by] ; [with] ; [about] .

# DCG example

```
?- consult(grammar).
```

```
true.
```

```
?- sentence([the, professor, saw, the, student], []).
```

```
true ;
```

```
false.
```

```
?- sentence([the, professor, saw, the, student, with, the, glasses], []).
```

```
true ;
```

```
true ;
```

```
false.
```

```
?- sentence([the, professor, saw, the, bird], []).
```

```
false.
```

- How many sentences the grammar generates?
- How many are correct?
- How can we correct them?

# Attribute grammars

- It is possible to embed **attributes** into logic grammars.
- We can use attributes for many different purposes.
- For instance, to transport any kind of *information* across different parts of the grammar.
- These attributes can represent linguistic features. For instance, number, semantic preferences, etc.

# Example of attribute grammars

sentence(W) --> noun\_phrase(W1), verb\_phrase(W2), {W is W1 + W2}.

noun\_phrase(2) --> determiner, noun.

noun\_phrase(W) --> determiner, noun, prepositional\_phrase(W1), {W is W1 + 2}.

verb\_phrase(1) --> verb.

verb\_phrase(W) --> verb, noun\_phrase(W1), {W is W1 + 1}.

verb\_phrase(W) --> verb, noun\_phrase(W1), prepositional\_phrase(W2), {W is W1 + W2 + 1}.

prepositional\_phrase(W) --> preposition, noun\_phrase(W1), {W is W1 + 1}.

noun --> [student] ; [professor] ; [book] ; [university] ; [lesson] ; [glasses].

determiner --> [a] ; [the] .

verb --> [taught] ; [learned] ; [read] ; [studied] ; [saw].

preposition --> [by] ; [with] ; [about] .

# Example of attribute grammars

?- consult(grammar).

true.

?- sentence(W, [the, professor, saw, the, student], []).

W = 5 ;

false.

?- sentence(W, [the, professor, saw, the, student, with, the, glasses], []).

W = 8 ;

W = 8 ;

false.

?- sentence(W, [the, professor, saw, the, bird], []).

false.

# Example of attribute grammars

sentence --> pronoun(subject), verb\_phrase.

verb\_phrase --> verb, pronoun(object).

pronoun(subject) --> [he].

pronoun(subject) --> [she].

pronoun(object) --> [him].

pronoun(object) --> [her].

verb --> [likes].

- This grammar allows sentences like "he likes her" and "he likes him"
- But not "her likes he" and "him likes him".

# Example of attribute grammars

sentence(s(NP,VP)) --> noun\_phrase(NP), verb\_phrase(VP).

noun\_phrase(np(D,N)) --> det(D), noun(N).

verb\_phrase(vp(V,NP)) --> verb(V), noun\_phrase(NP).

det(d(the)) --> [the].

det(d(a)) --> [a].

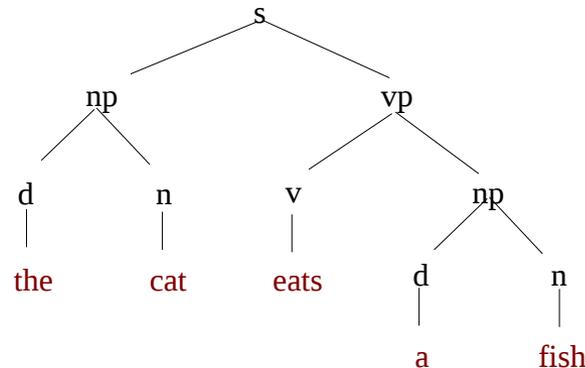
noun(n(fish)) --> [fish].

noun(n(cat)) --> [cat].

verb(v(eats)) --> [eats].

?- sentence(Parse\_tree, [the,cat,eats,a,fish], []).

Parse\_tree = s(np(d(the),n(cat)),vp(v(eats),np(d(a),n(fish)))) ?;



# Exercises

- Integrate the WordNet lexicon in some of the example grammars.
- Obtain the logical formula from a sentence:
  - `sentence(LF, [the,cat,eats,a,fish], [])`.
  - `LF = eat(cat, fish) ?`;
- Answer commonsense questions (sintagmatic relations):
  - `sentence([a,monkey,eats,a,banana], [])`. => yes.
  - `sentence([a,monkey,eats,a,book], [])`. => no.
  - `sentence([a,doctor,reads,a,book], [])`. => yes.
  - `sentence([a,doctor,reads,a,banana], [])`. => no.
  - *Hint: reuse WN lexicographical files*
  - *Hint: create a new file from `wn_sk.pl` with the appropriate info.*
- Answer simple commonsense questions (paradigmatic relations):
  - `sentence([an,apple,is,a,fruit], [])`. => yes.
  - `sentence([an,apple,is,an,animal], [])`. => no.
  - `sentence([an,apple,has,apples], [])`. => yes.
  - `sentence([an,apple,has,fingers], [])`. => no.
  - *Hint: reuse WN hyponymy and part\_of relations*

# Assignments

- Answer more commonsense questions (sintagmatic relations):
  - `sentence([a,monkey,eats,a,banana], [])`. => yes.
  - `sentence([a,monkey,eats,a,book], [])`. => no.
  - `sentence([a,doctor,reads,a,book], [])`. => yes.
  - `sentence([a,doctor,reads,a,banana], [])`. => no.
  - *Hint: reuse WN lexicographical files*
  - *Hint: create a new file from `wn_sk.pl` with the appropriate info.*
  - *Extension: explain why ... (which senses hold the selectional preferences).*
- Answer simple commonsense questions (paradigmatic relations):
  - `sentence([an,apple,is,a,fruit], [])`. => yes.
  - `sentence([an,apple,is,an,animal], [])`. => no.
  - `sentence([an,apple,has,apples], [])`. => yes.
  - `sentence([an,apple,has,fingers], [])`. => no.
  - *Hint: reuse WN hyponymy and part\_of relations*
  - *Extension: explain why (which senses and relations) hold.*
  - *Extension: transitive meronymy.*
  - *Extension: transitive meronymy through hypernymy.*

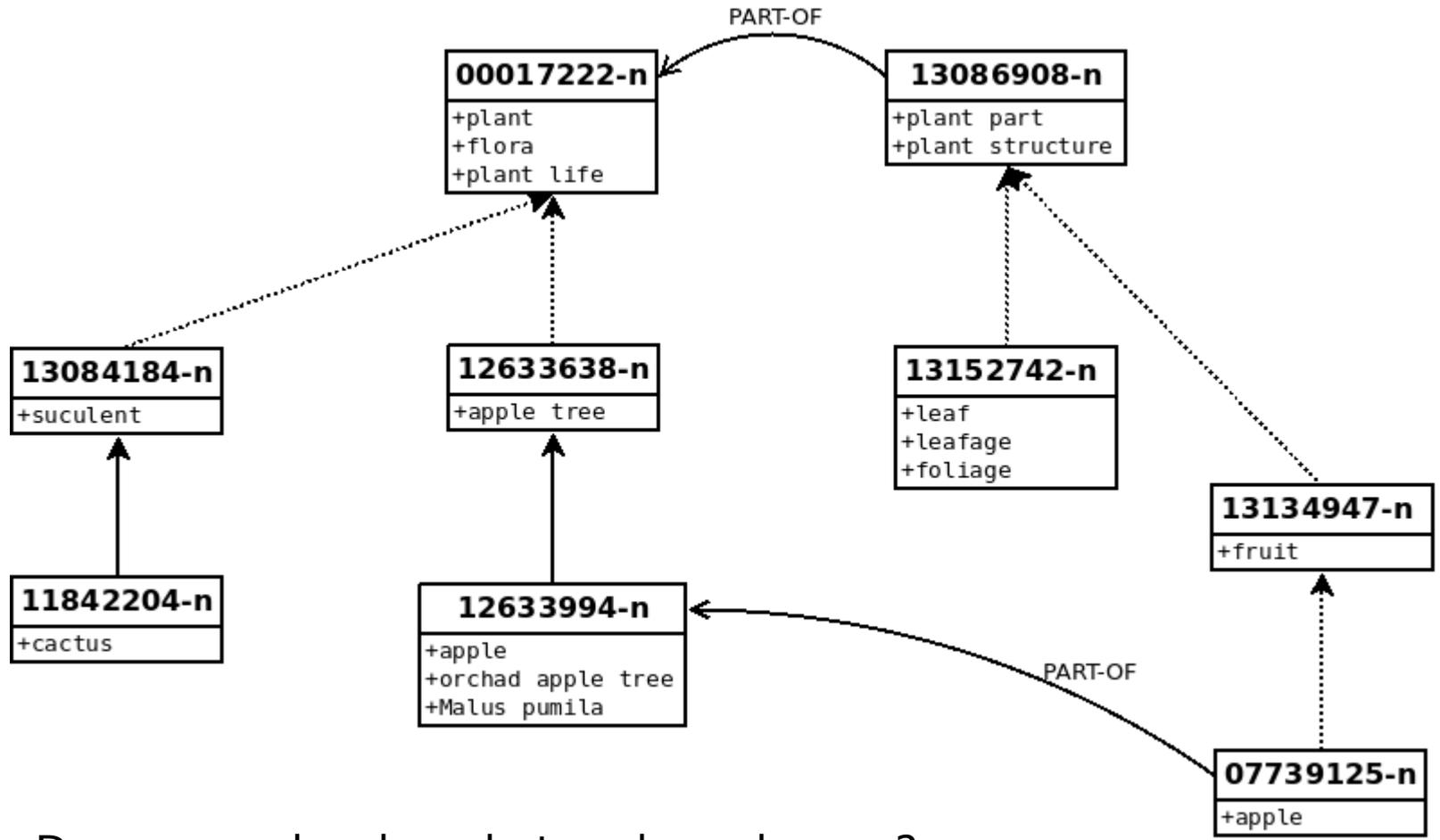
# Assignment 1

- Answer more commonsense questions (sintagmatic relations):
  - `sentence([a,monkey,eats,a,banana], [])`. => yes.
  - `sentence([a,monkey,eats,a,book], [])`. => no.
  - `sentence([a,doctor,reads,a,book], [])`. => yes.
  - `sentence([a,doctor,reads,a,banana], [])`. => no.
  - *Hint: reuse WN Lexicographical files*
  - *Hint: create a new file from `wn_sk.pl` with the appropriate info.*
  - *Extension: explain why ... (which senses hold the selectional preferences).*
- Which is the appropriate info for the new file?
- How can you create that file?
- Which selectional preferences apply?
- How can you implement these selectional restrictions?

# Assignment 2

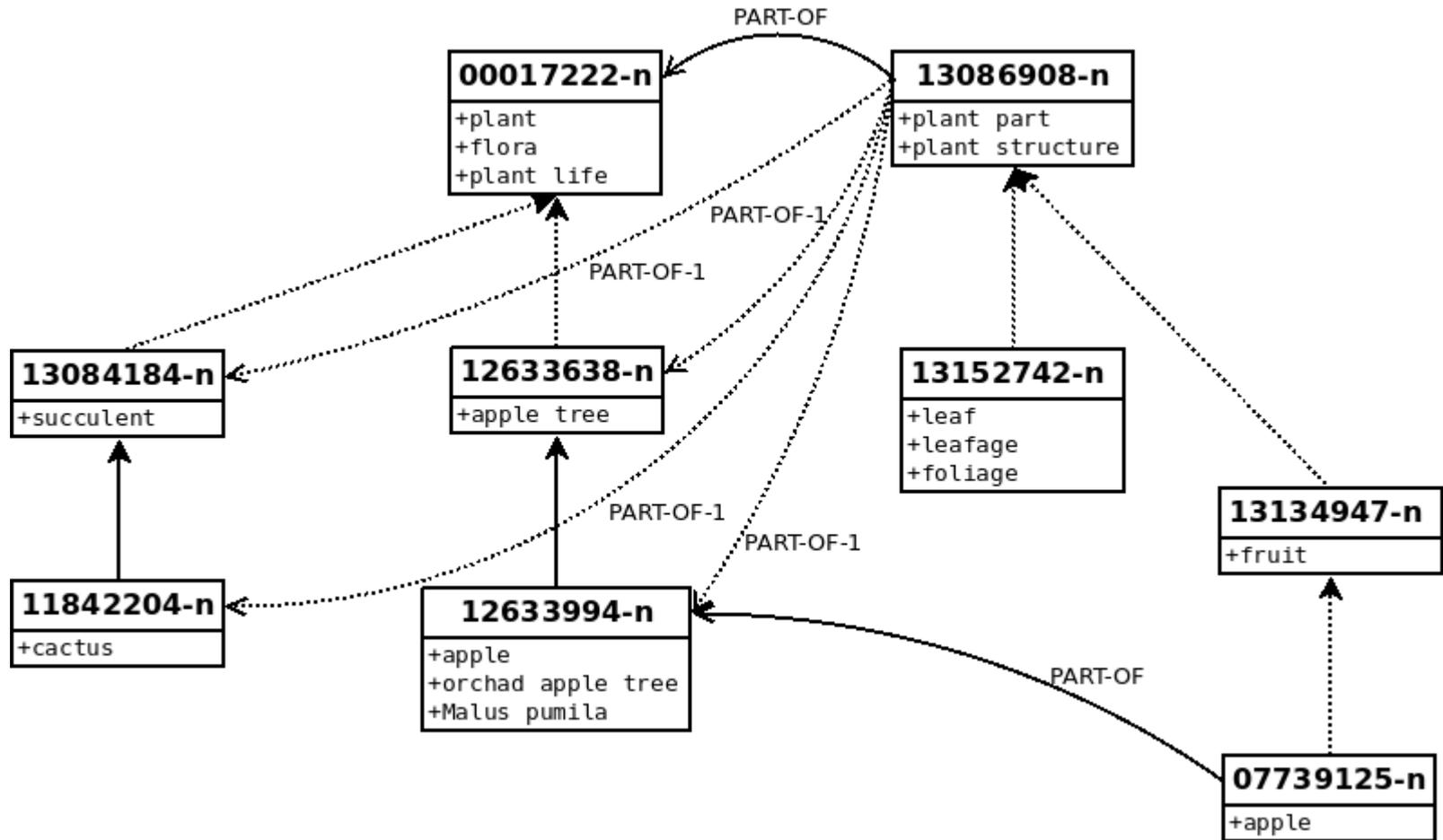
- Answer simple commonsense questions (paradigmatic relations):
  - `sentence([an,apple,is,a,fruit], [])`. => yes.
  - `sentence([an,apple,is,an,animal], [])`. => no.
  - `sentence([an,apple,has,apples], [])`. => yes.
  - `sentence([an,apple,has,fingers], [])`. => no.
  - *Hint: reuse WN hyponymy and part\_of relations*
  - *Extension: explain why (which senses and relations) hold.*
  - *Extension: transitive meronymy.*
  - *Extension: transitive meronymy through hypernymy.*
- Which inheritance mechanisms make sense?
- How can you implement them?
- Also note that there are repeated new inferred relations.

# Exercices



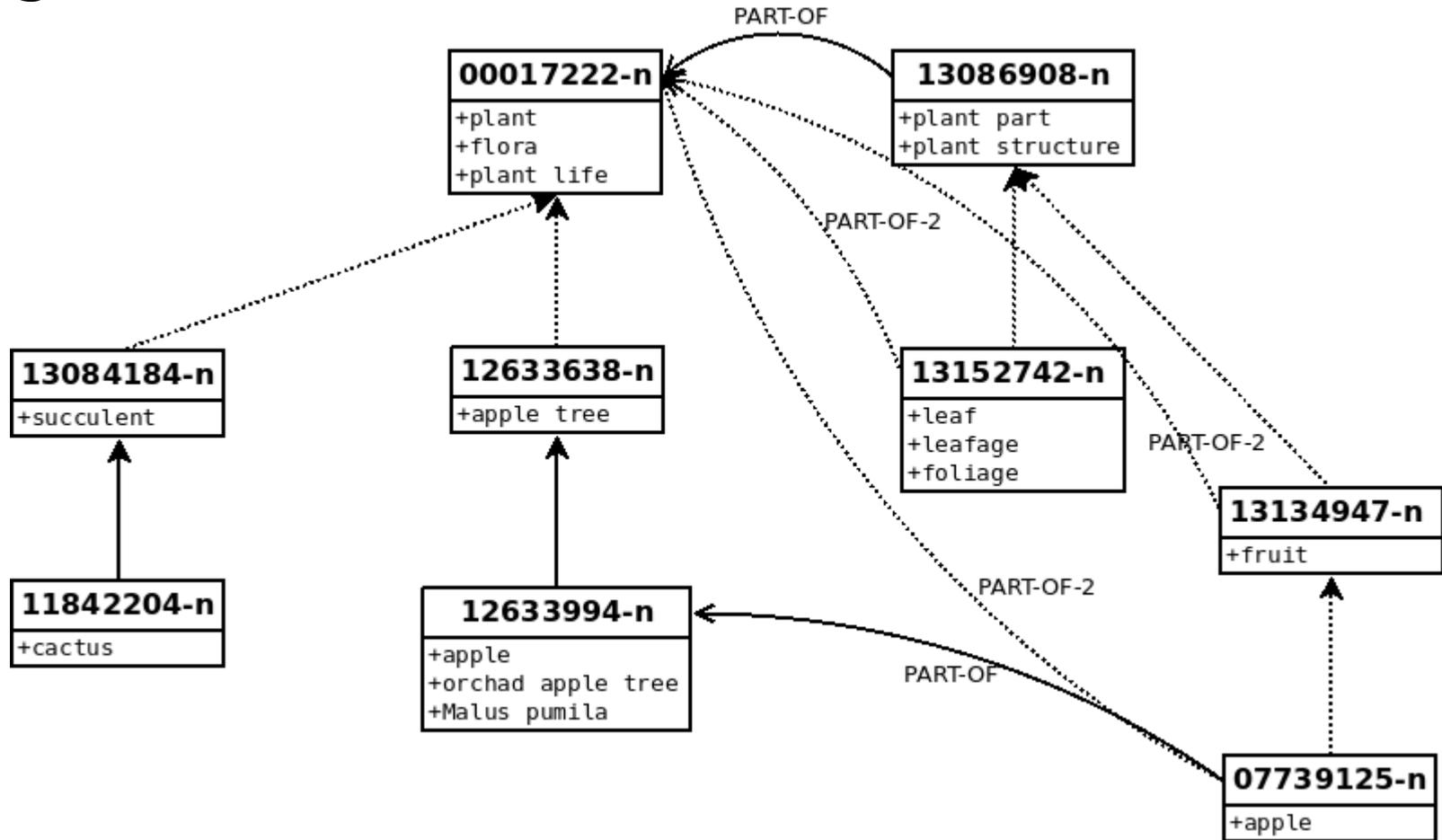
- Does an orchard apple tree have leaves?
- Does an orchard apple tree have fruits?
- Does a cactus have leaves?

# Assignment 2



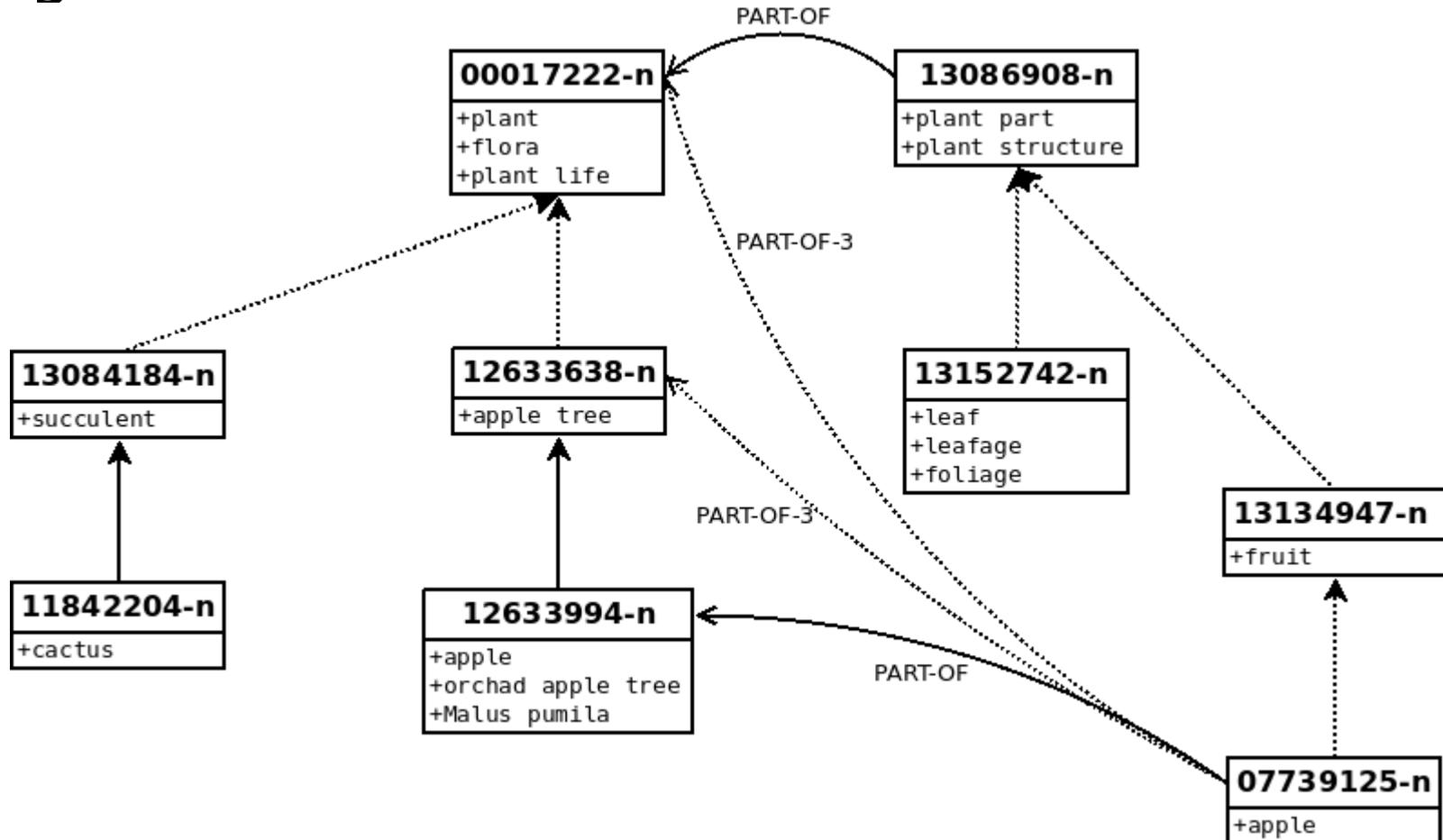
- Inheritance of part-of relations: type 1?
- The PART is inherited through the hyponymy of the WHOLE?

# Assignment 2



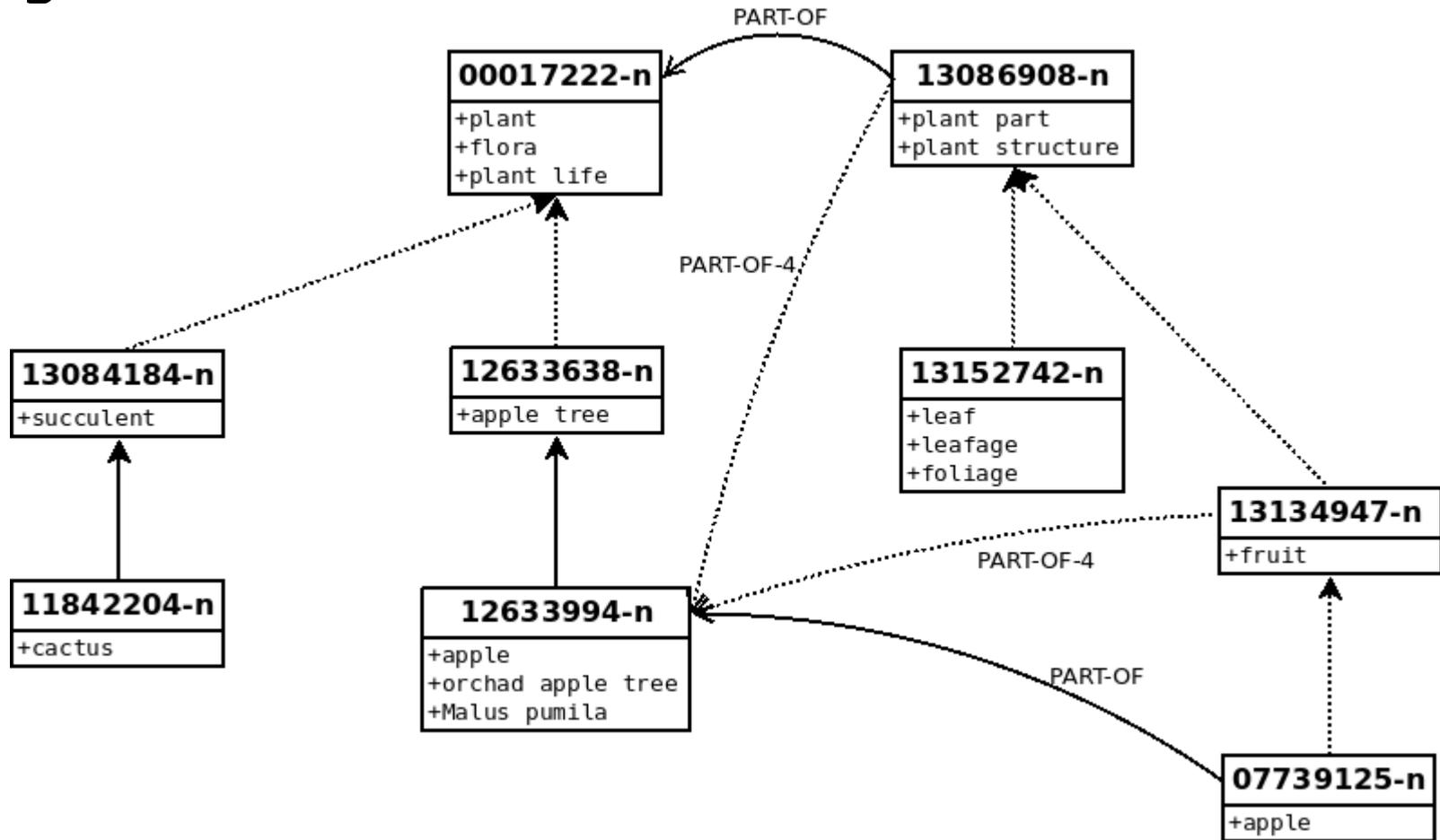
- Inheritance of part-of relations: type 2?
- The WHOLE is inherited through the hyponymy of the PART?

# Assignment 2



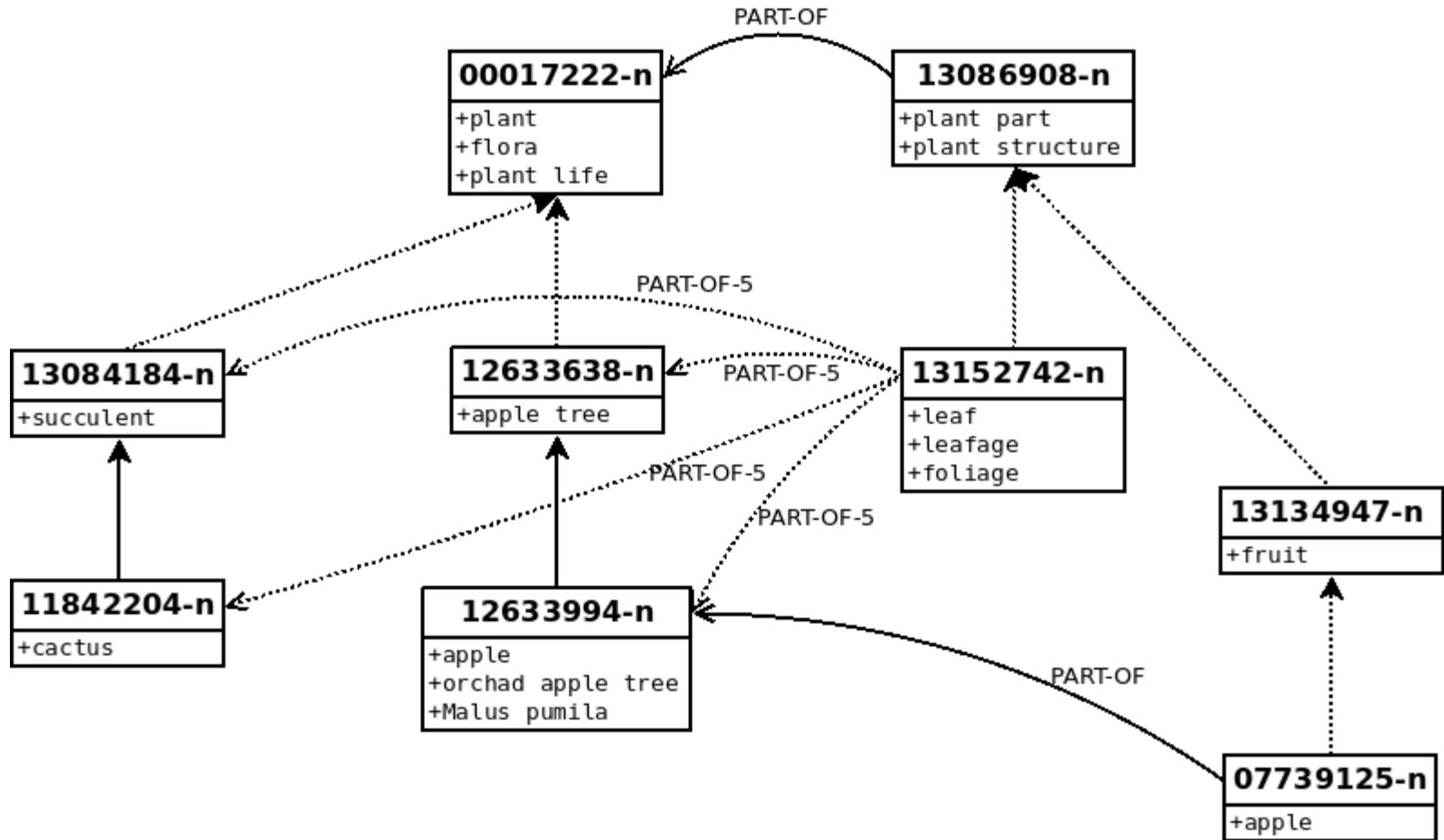
- Inheritance of part-of relations: type 3?
- The PART is transferred through the hypernymy of the WHOLE?

# Assignment 2



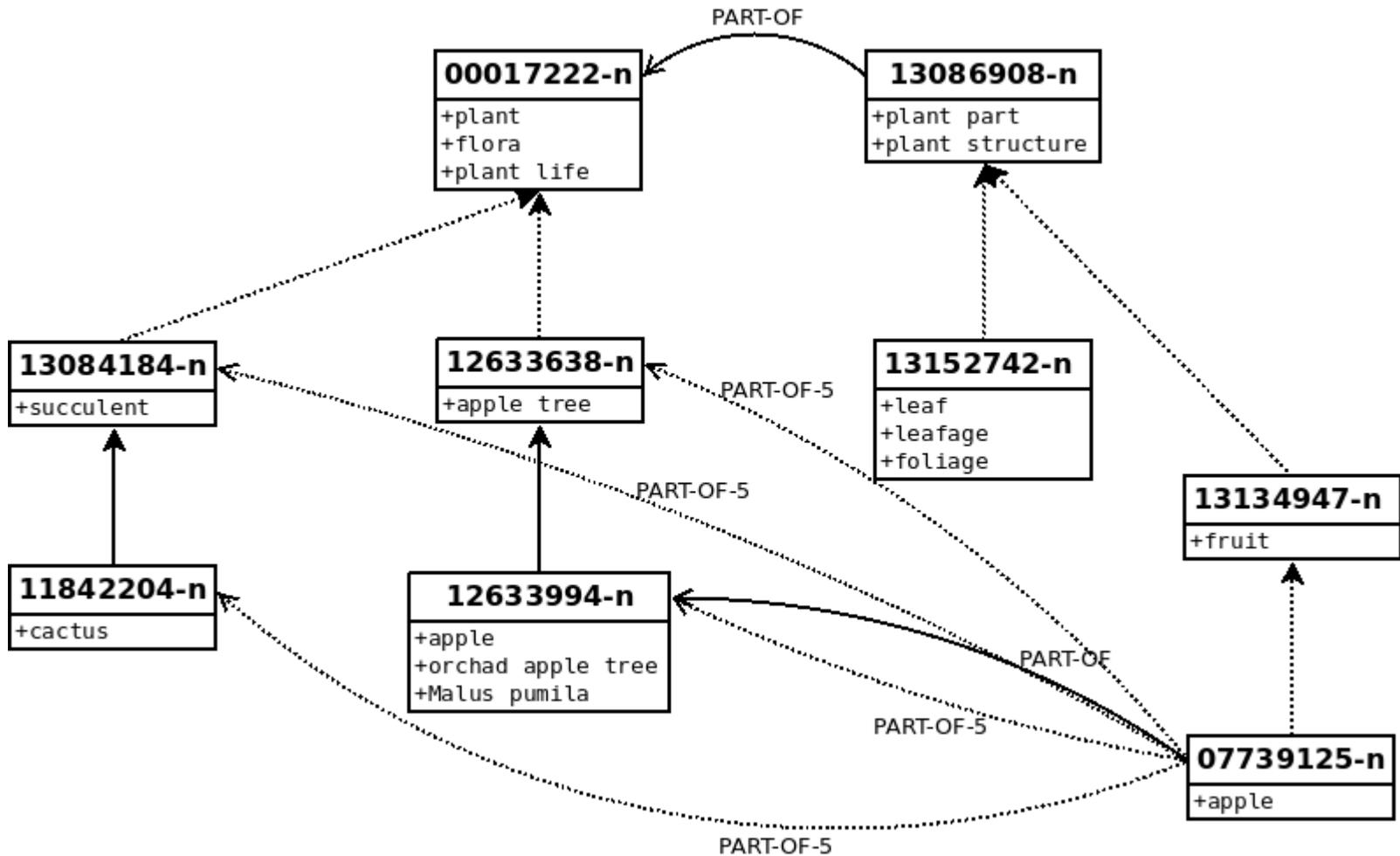
- Inheritance of part-of relations: type 4?
- The WHOLE is transferred through the hypernymy of the PART?

# Assignment 2



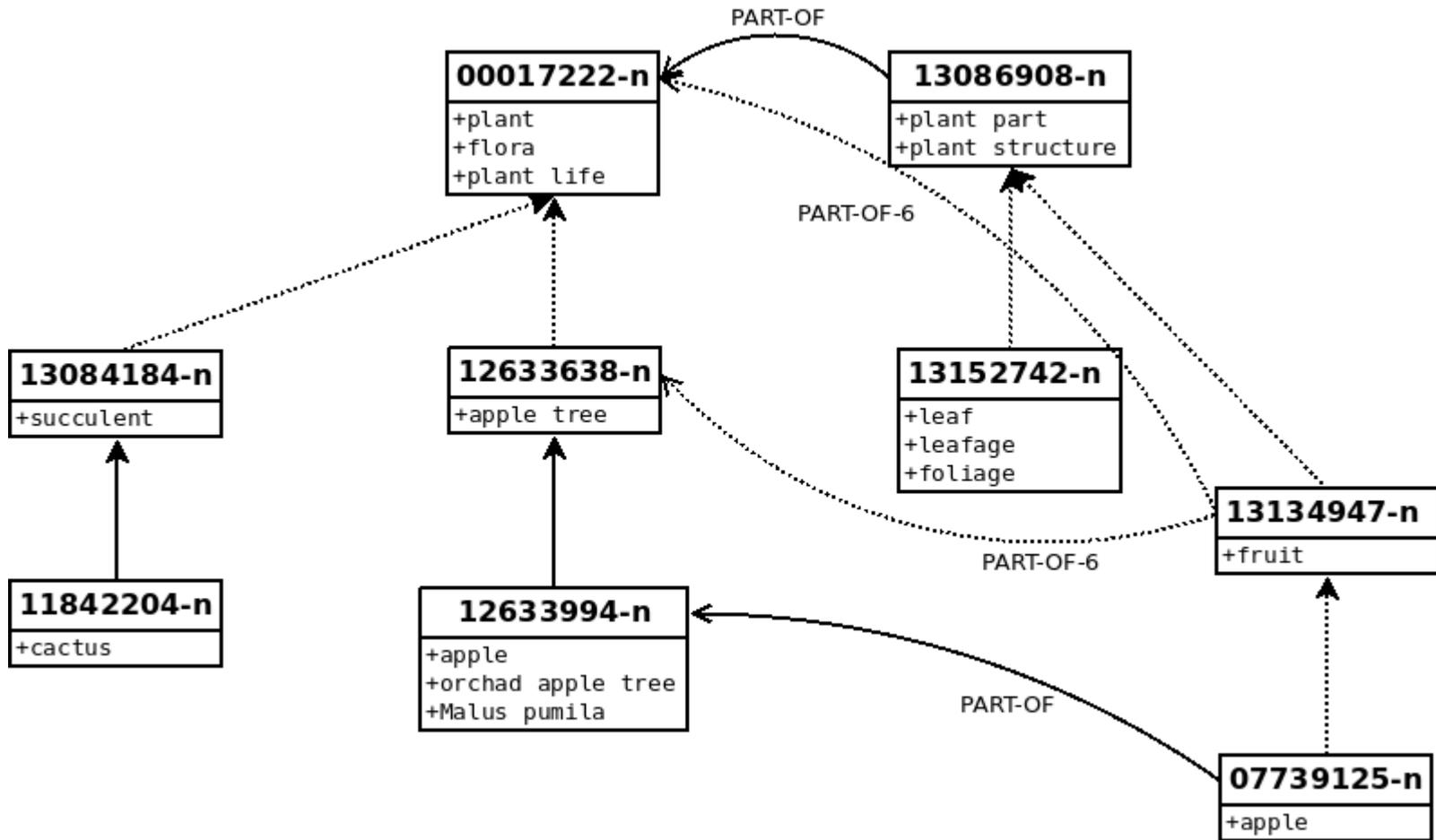
- Inheritance of part-of relations: type 5?
- Inheritance through both hyponymy of PART and WHOLE?

# Assignment 2



- Inheritance of part-of relations: type 5?
- Inheritance through both hyponymy of PART and WHOLE?

# Assignment 2



- Inheritance of part-of relations: type 6?
- The WHOLE and PART are transferred through the hypernymy?

# Prolog & NLP



German Rigau i Claramunt

<http://adimen.si.ehu.es/~rigau>

IXA group

Departamento de Lenguajes y Sistemas Informáticos

UPV/EHU