

semti

kamols

# **Polysemy** **in Controlled Natural Language Texts**

Normunds Grūzītis & Guntis Bārzdiņš

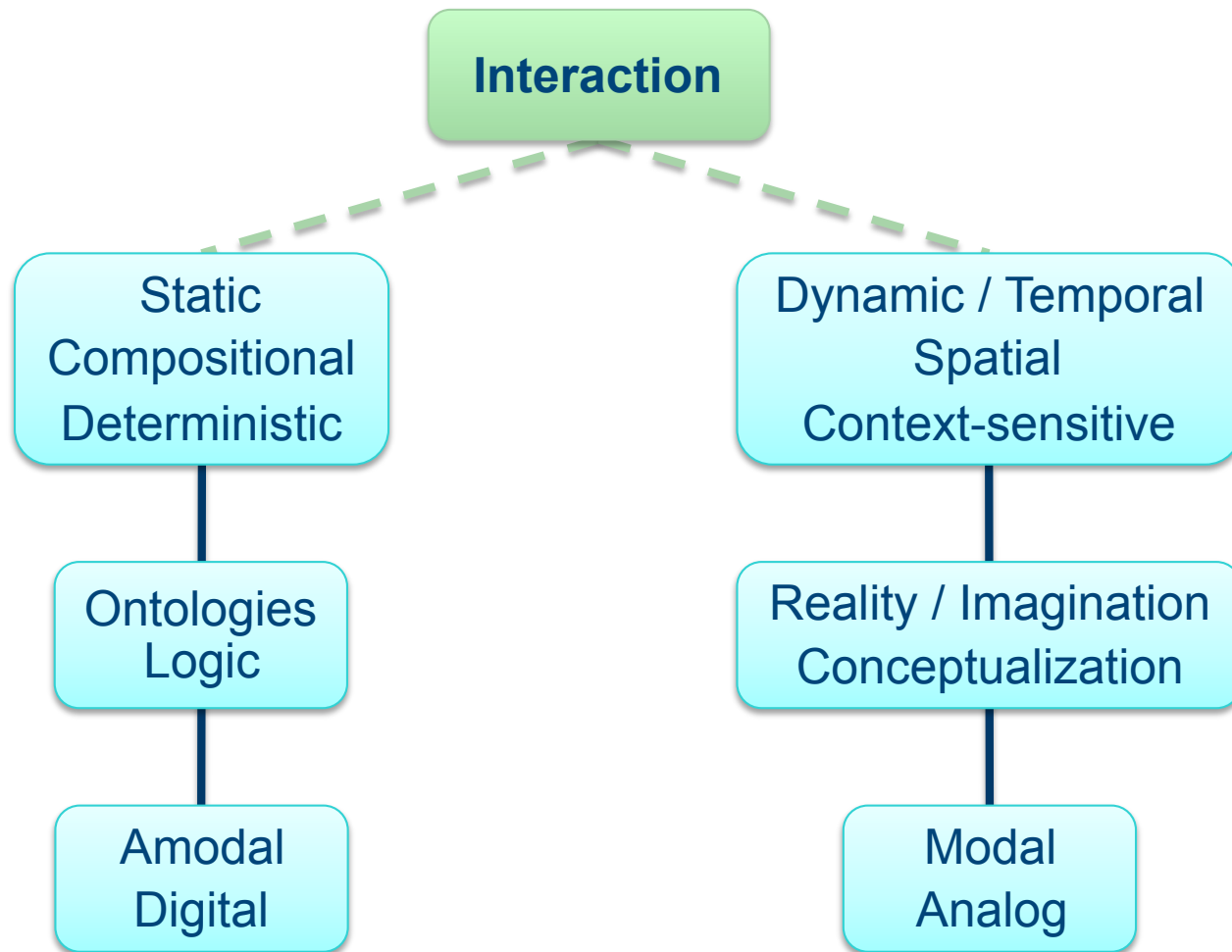
Workshop on Controlled Natural Language

8–10 June 2009, Marettimo Island, Italy

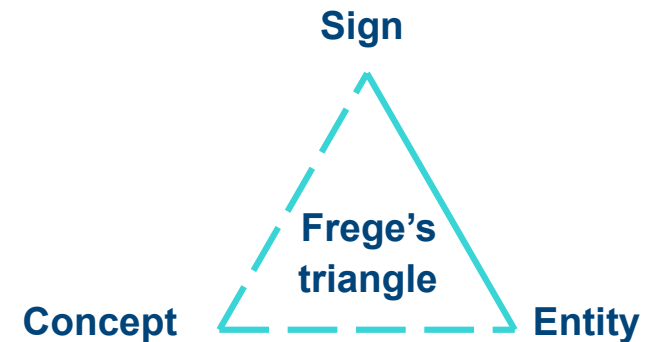
# Agenda

- **Polysemy**: causes and types
- Supporting polysemy in two alternative controlled natural languages
  - **Declarative** CNL
    - Ontological knowledge for WSD
  - **Procedural** CNL
    - Semantics is not based in FOL

# Two Subsets of Natural Language



# Polysemy



- **'Finite'** set of words (signs)
  - **Unlimited** number of (new) concepts
- ⇒ **Reuse** of existing words in different **contexts**
- 1) **Metaphorically** (figurative senses)  
*"Language is a graveyard of dead **metaphors**"* (Leary, 1994)
  - 2) **Metonymically**  
e.g., "library" for "building of library"
  - 3) Collocations → multi-word units

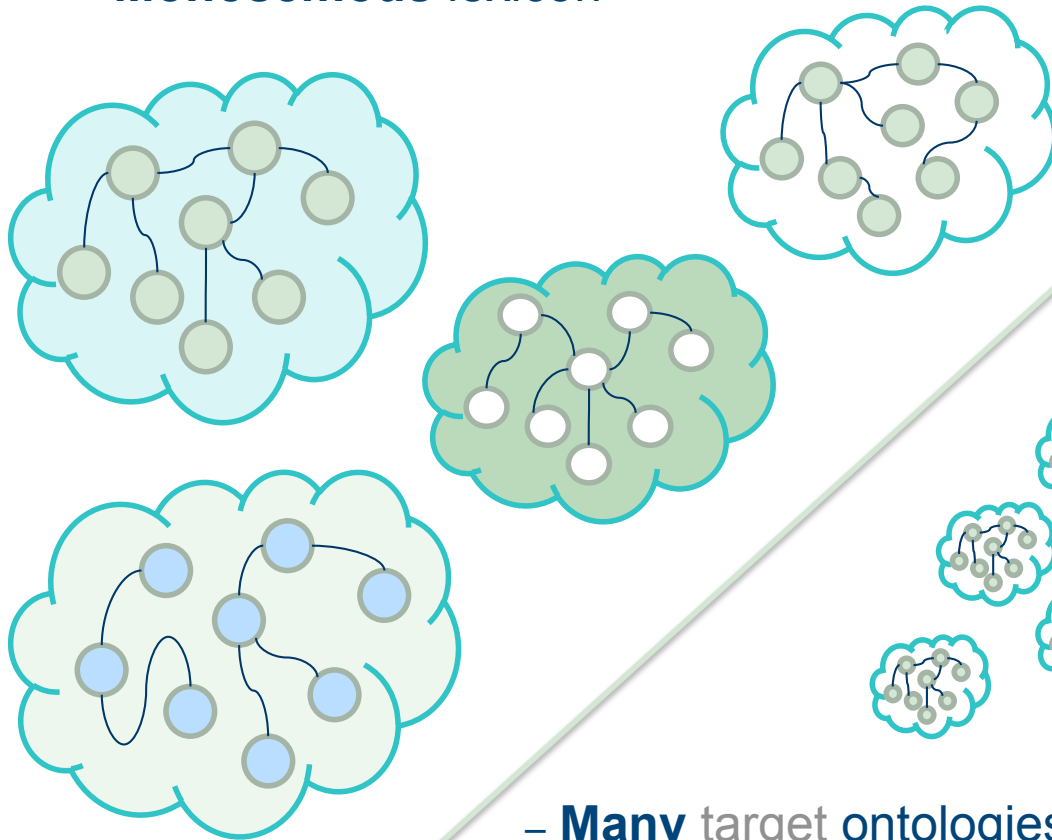
# Polysemy in a Declarative CNL

# Ontological vs. Factual Sentences

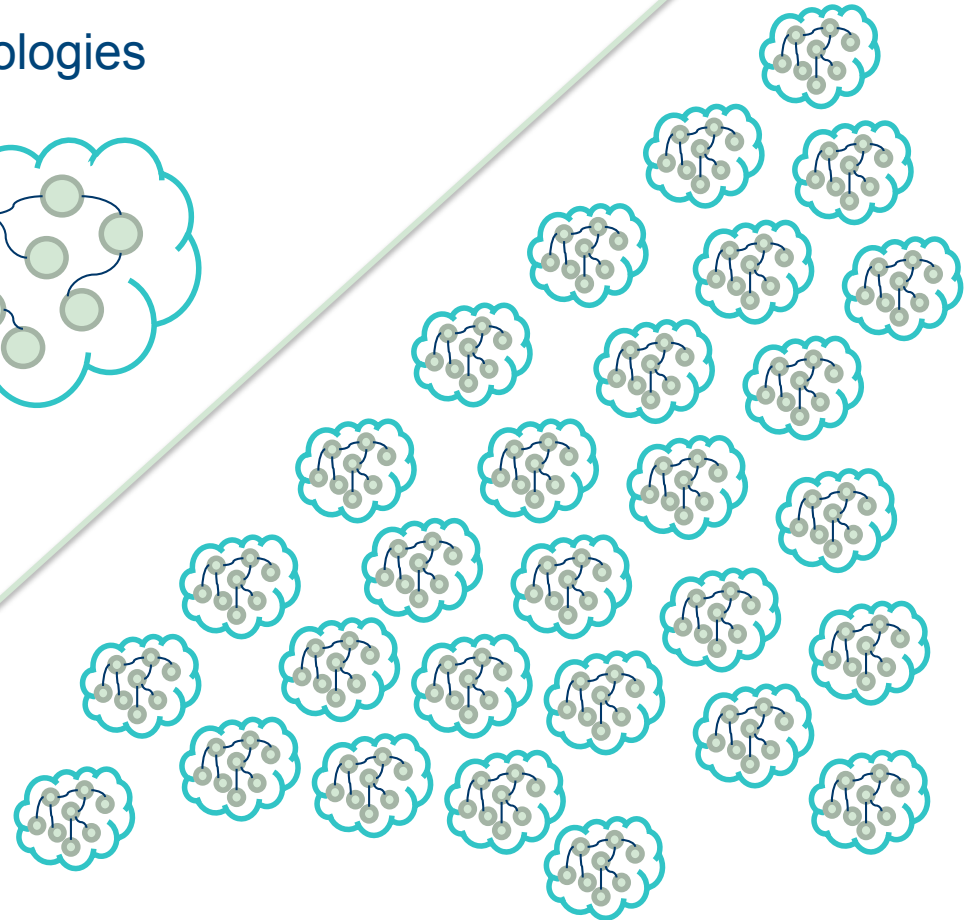
- *Every mouse is an animal.*
- *The black mouse is not working properly.*
  - *It is used by no computer.*
- CNL for T-Box vs. **A-Box**
  - Relieve **average users** of providing ontological sentences
    - Leave creation of consistent ontologies to knowledge engineers and domain experts
  - ⇒ **Polysemy** should appear only in the **factual** sentences, which can refer to the mix of domain ontologies
    - Ontology **population** with facts
      - Information extraction (IE)
      - Web page descriptions in CNLs (Semantic Web)
    - ⇒ **Multi-lingual semantic search engine**

# User's perspective

- **One** or few **consistent** target ontologies
- **Monosemous** lexicon



- **Many** target ontologies that may be mutually **inconsistent**
- **'Polysemous'** lexicon



# Micro-ontologies

- Requirements
    - **Internally consistent**
      - OWL DL compliant
    - **Lexicon**-driven (concept naming)
    - **Syntax**-driven (property mapping)
- } cues for invoking
- Consequences
    - A set of **translation equivalents** and synonyms can be attached to a concept or property
      - Ontologies themselves are language-independent



# WSD as Ontology Merging

- *Two sides of the same coin*
- **Difficult**: match the equivalent concepts & properties
  - Facing the **word-sense disambiguation** problem
    - **Lexical** naming & **syntactic** mapping guidelines → **hints**
- **Easy**: ensure that the merger is consistent
  - OWL DL **reasoners**
- **Interpretation** = consistent matching & merging

# Multi-domain Communication

T-Box	Micro-ontologies	
	Domain	Axioms
	Buildings	Every building is a construction and has a roof. Every <b>library</b> is a <b>building</b> .
	Collections	Every collection is an abstract-entity that contains some items. Every <b>library</b> is a <b>collection</b> that contains some publications.
	General	Every construction is a physical-entity. <b>No</b> physical-entity is an abstract-entity.
A-Box	Assertions	
	<i>There is a <b>library</b> that has a green roof.</i> <i>The <b>library</b> contains some valuable publications.</i>	

# Multi-domain Communication

T-Box	Micro-ontologies	
	Domain	Axioms
	Merged ontology	Every building is a construction and has a roof. Every <b>library[building]</b> is a <b>building</b> .
		Every collection is an abstract-entity that contains some items. Every <b>library[collection]</b> is a <b>collection</b> that contains some publications.
		Every construction is a physical-entity. <b>No</b> physical-entity is an abstract-entity.
A-Box	Assertions	
	<i>There is a library[building] that has a green roof.</i> <i>The library[collection] contains some valuable publications.</i>	

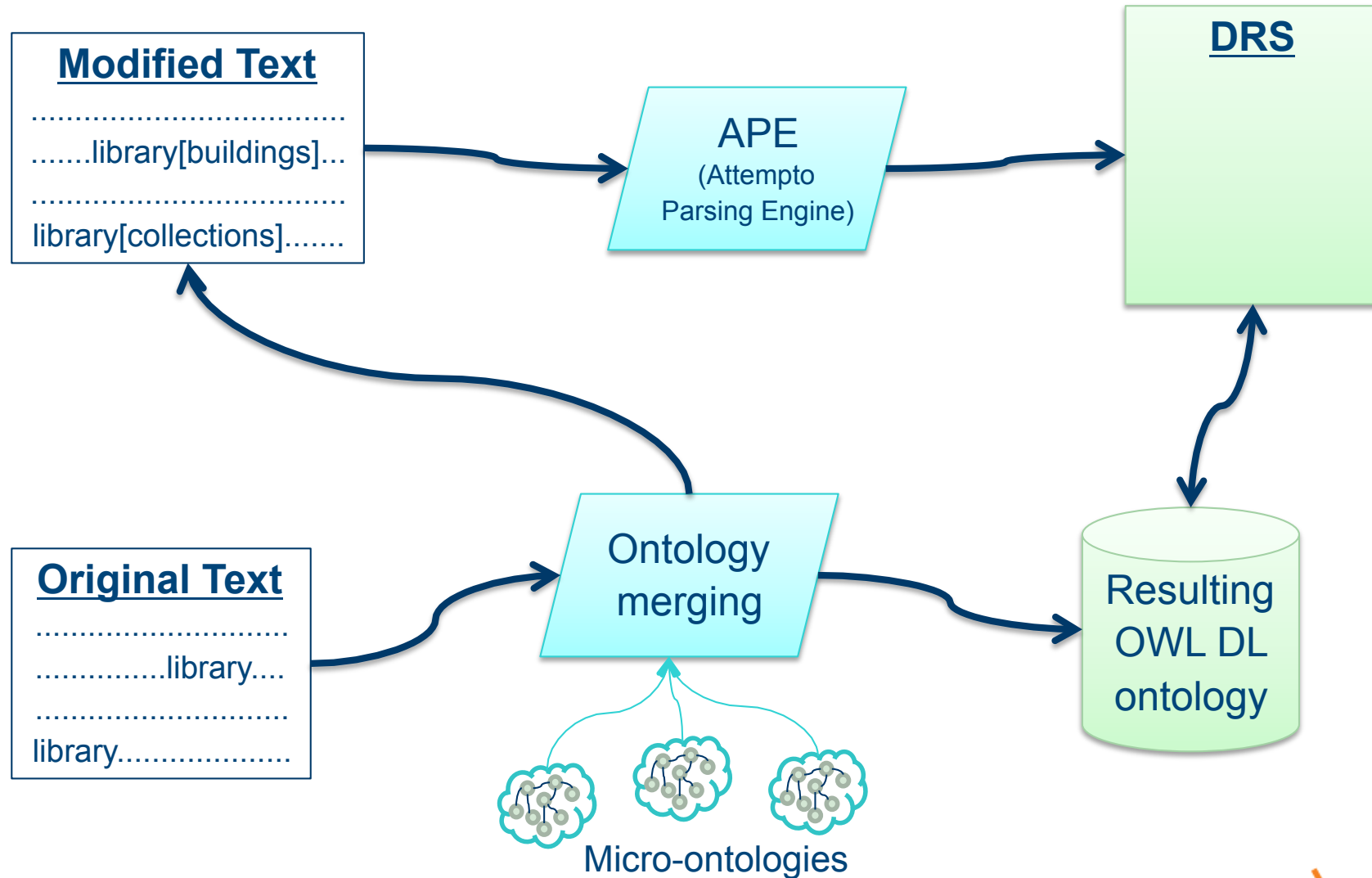
Solution found through an exhaustive search (with possible user interaction)

# Multi-lingual Communication

T-Box	Micro-ontologies	
	Domain	Axioms
	#1	$\forall x (\text{artifact}(x) \rightarrow \neg \text{body-part}(x))$ $\forall x (\text{footwear}(x) \rightarrow \text{artifact}(x))$
	#2	$\forall x (\text{shoe}_{\text{kurpe}}(x) \rightarrow \text{footwear}(x))$ $\forall xy (\text{polish}_{\text{pucēt}}(x,y) \rightarrow \text{person}(x) \ \& \ \text{footwear}(y))$
A-Box	#3	$\forall x (\text{nail}_{\text{nags}}(x) \rightarrow \text{body-part}(x))$ $\forall xy (\text{polish}_{\text{vīlēt}}(x,y) \rightarrow \text{person}(x) \ \& \ \text{nail}_{\text{nags}}(y))$
	Assertions	
	Source text	Target text
	<i>John <u>polishes</u> a <u>shoe</u>.</i> <i>Ann <u>polishes</u> some red <u>nails</u>.</i>	<i>Jānis <u>pucē</u> vienu <u>kurpi</u>.</i> <i>Anna <u>vīlē</u> sarkanus <u>nagus</u>.</i>

OWL DL micro-ontologies as interlingua

# The Overall Picture

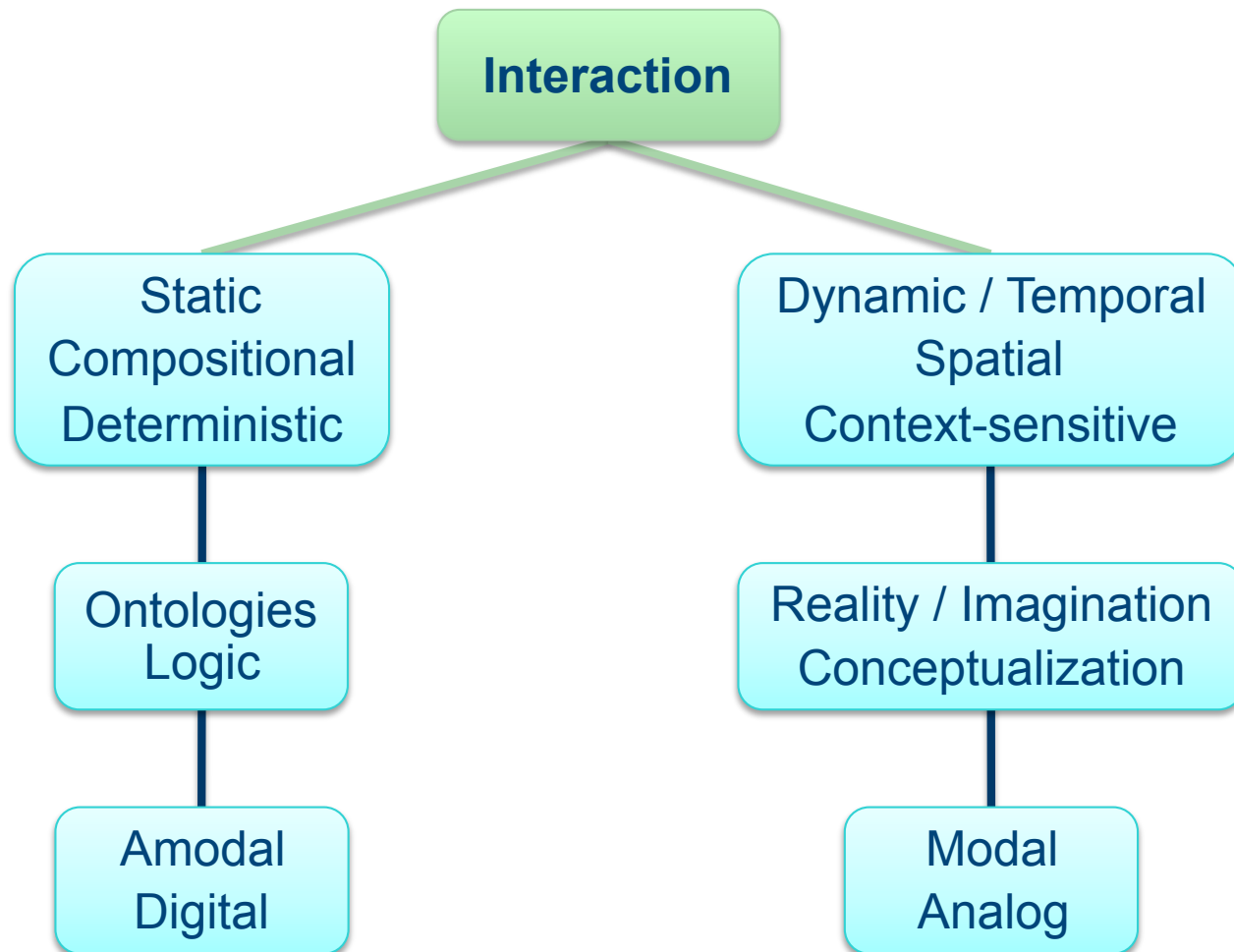


# Discussion

- User doesn't have to provide the target ontology
  - Unlimited 'repository' of **cross-language** micro-ontologies, that are **implicitly reused**
- User only **populates** existing ontologies with facts
  - Automatic word-sense disambiguation
- **Adaptation** of existing domain-ontologies
  - Lexical-driven naming conventions
  - Creation of **bridging**-ontologies if necessary
- No changes to existing 'monosemous' CNL machinery

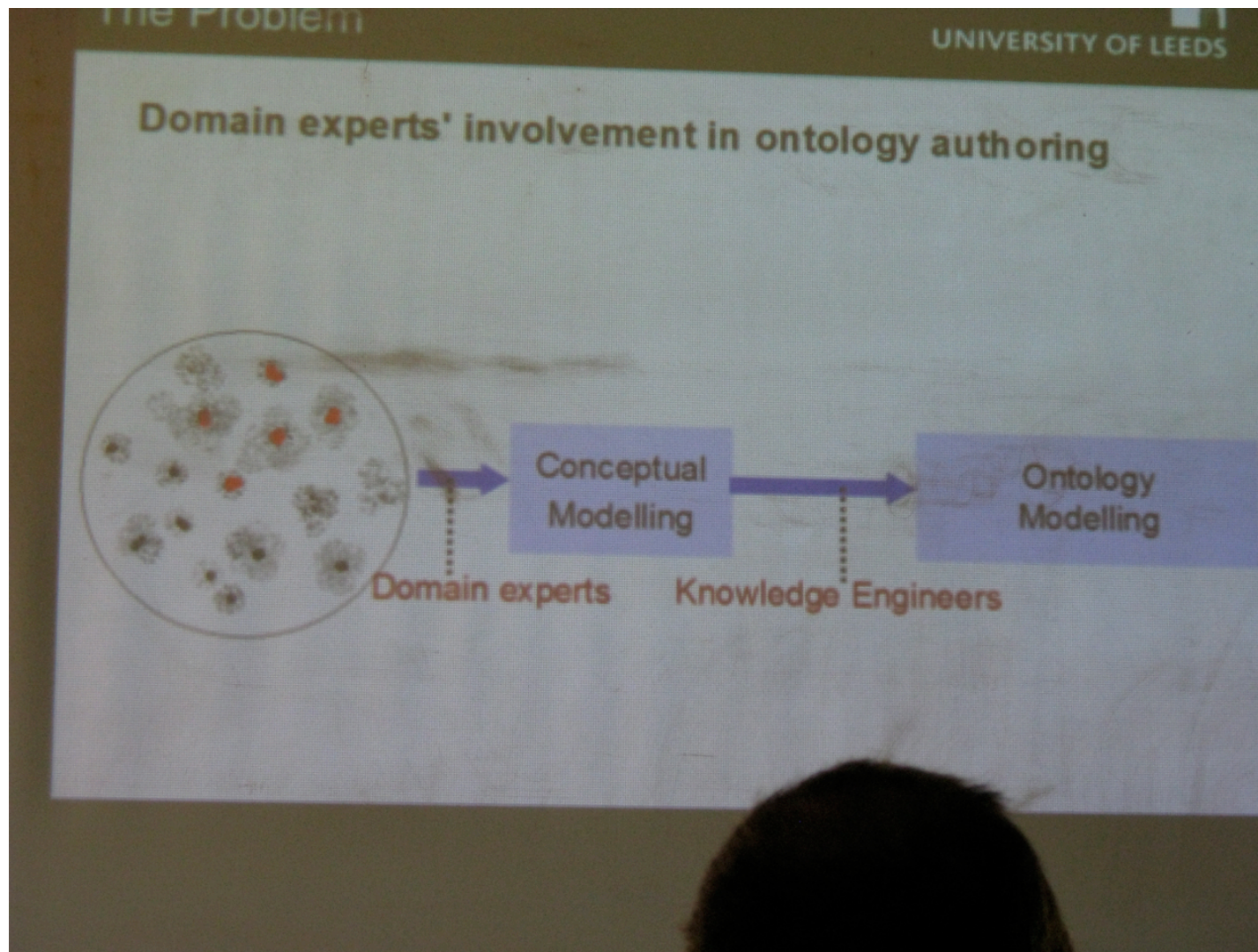
# Polysemy in a Procedural CNL

# Two Subsets of Natural Language

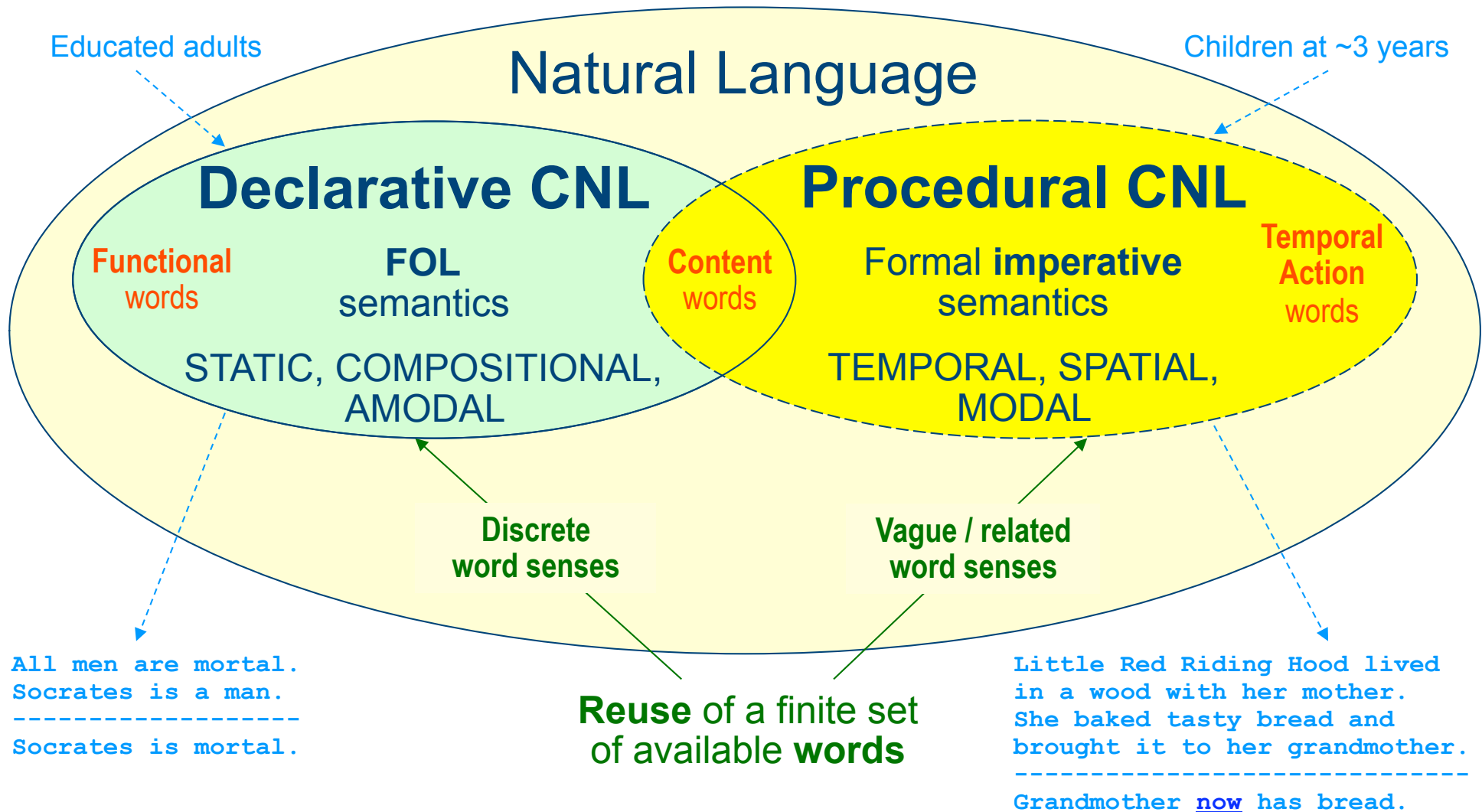




# Ronald Denaux slide



# Declarative vs. Procedural CNL



# FrameNet

- Developed in ISCI, Berkley by C.Fillmore et.al.
- Consists of ~800 *frames* (generic situations and objects) and their arguments – *frame elements*
- Derived from extensive text corpus evidence – new frames caused only by unique *argument structure*
- Frames organized in *inheritance* hierarchies
- Largely language independent
  - LexicalUnits assigned to frames
    - back.n (Observable\_bodyparts)
    - back.n (Part\_orientational)
    - back.v (Self\_motion)
    - back.a (Part\_orientational)

Bringing

Definition:

This frame concerns the movement of a **Theme** and an **Agent** and/or **Carrier**. The **Agent**, a person or other sentient entity, controls the shared **Path** by moving the **Theme** during the motion. In other words, the **Agent** has overall motion in directing the motion of the **Theme**. The **Carrier** may be a separate entity, or it may be the **Agent's** body. The **Constant location** may be a subregion of the **Agent's** body or (a subregion of) a vehicle that the **Agent** uses.

Karl CARRIED the books across campus to the library on his head.  
Karl CARRIED the books across campus to the library in his truck.  
Karl CARRIED the books across campus to the library by truck.  
The truck CARRIED the books across campus to the library in specially designed boxes.

The FEs include **Path**, **Goal**, and **Source**. **Area** is an area that contains the motion when the path is understood as irregular. This frame emphasizes the path of movement as opposed to the FEs Source or Goal as in Filling or Placing.

FEs:

Core:

**Agent [Agt]**  
Semantic Type  
Sentient

The **Agent** is a sentient being who physically controls the movement of the **Theme** via the carrier, accompanying the **Theme**.

Karl CARRIED the books across the campus to the library.

**Area [Area]**

**Area** is used for description of a general area in which the carrying action takes place when the motion is understood to be irregular or not to consist of a single, linear path.

**Carrier [Car]**

The **Carrier** provides support for the **Theme**. Movement of the **Carrier** results in movement of the **Theme**.

The boat FERRIED the troops across the river.

**Goal [goal]**  
Semantic Type  
Goal

**Goal** identifies the endpoint of the path.

Karl CARRIED the books to the library.

**Path [Path]**

Path along which carrying occurs.

Karl CARRIED the books across the campus.

**Source [sou]**  
Semantic Type  
Source

**Source** indicates the beginning of the path along which the **Theme** travels.

Karl HAULED the books from the library to the office.

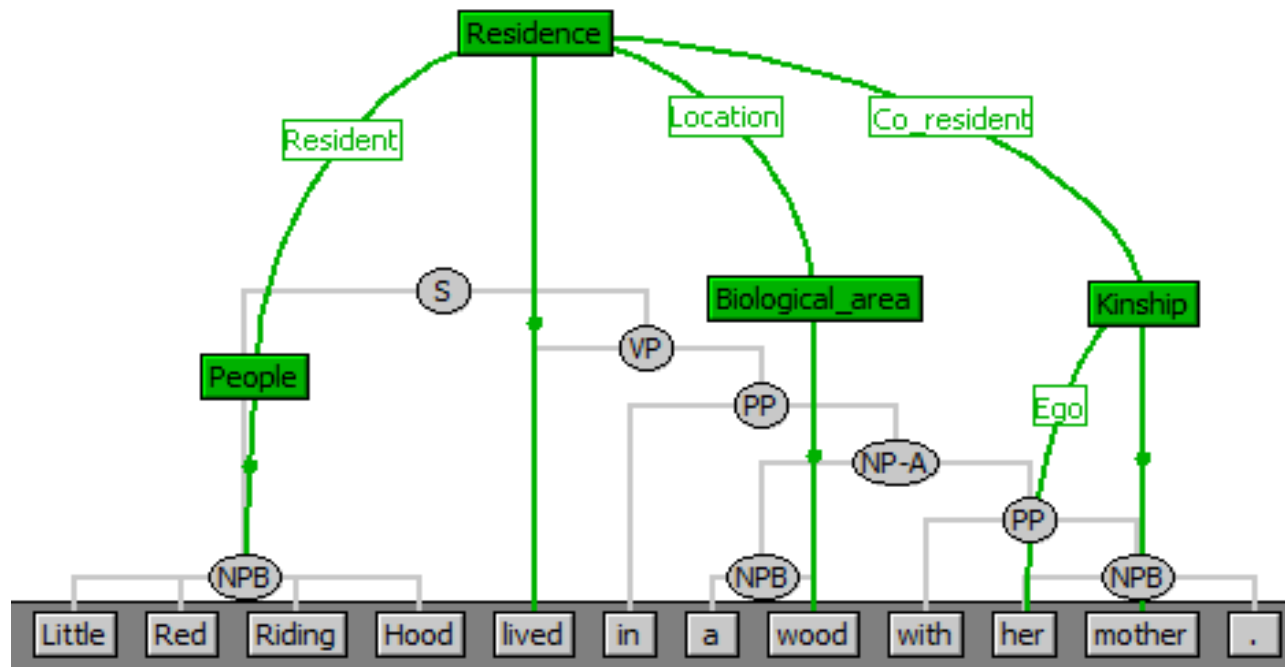
**Theme [Theme]**  
Semantic Type  
Physical object

The objects being carried.

Karl TOTED the books to the car.

# What is a Procedural CNL?

- **Procedural CNL Definition:** text that 100% maps into sequential FrameNet OBJECT and SITUATION frames



- **Polysemy:** many lexemes map into the same frame; specific lexemes used only for **anaphora** resolution and visual identification (**icons**)

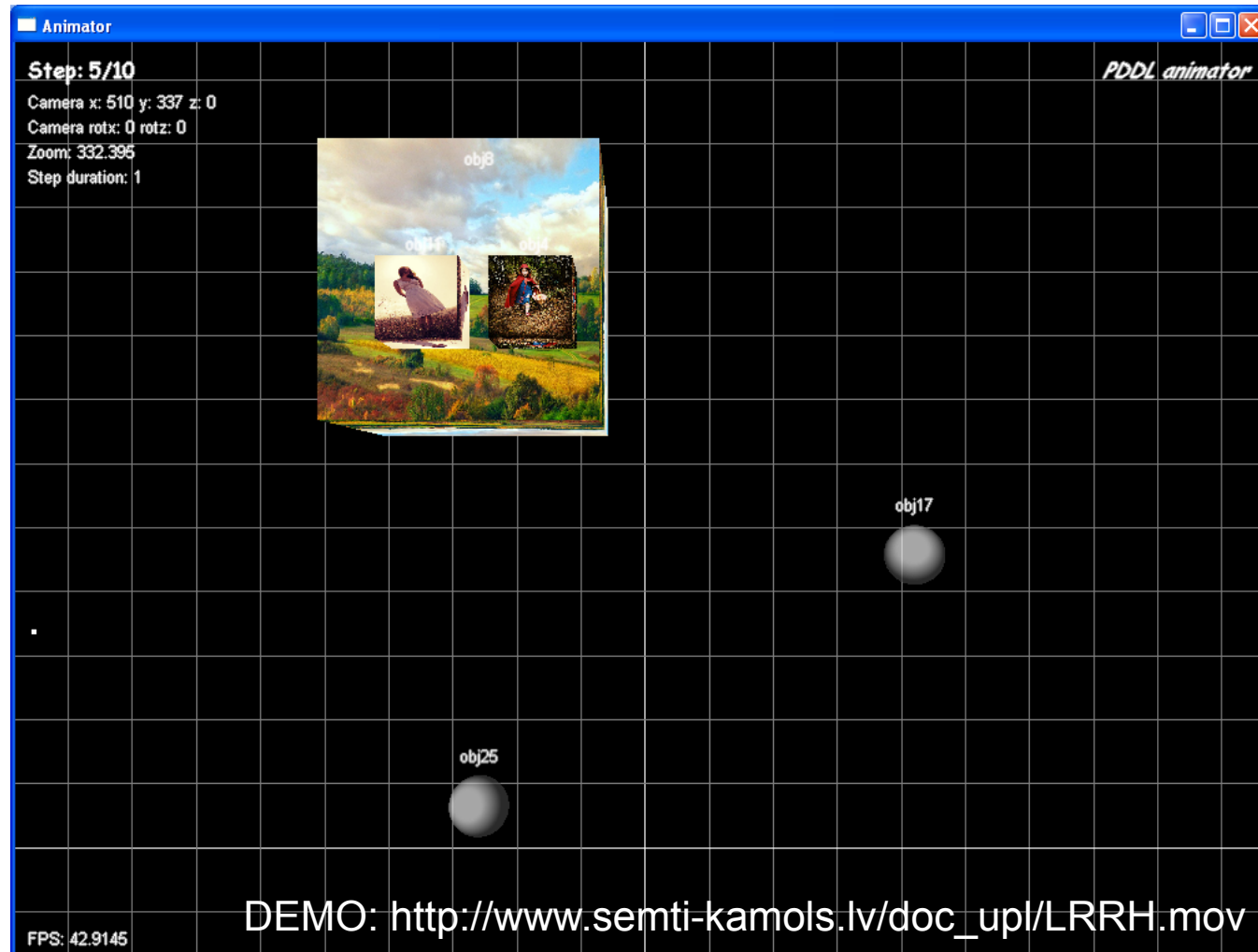
# Text Example in Procedural CNL

FrameNet annotation  
+ anaphora resolution

1. Little Red Riding Hood
2. lived
3. in a wood
4. with her mother.
5. She baked
6. tasty
7. bread
8. and brought it
9. to her grandmother.

1. **people**  
person=obj4 icon="littleredridinghood.m3d"
2. **residence**  
co-resident=obj11 location=obj8 resident=obj4
3. **biological\_area**  
locale=obj8 icon="wood.m3d"
4. **kinship**  
alter=obj11 ego=obj4 icon="mother.m3d"
5. **cooking\_creation**  
cook=obj4 food=obj15
6. **chemical\_sense\_description**  
perception\_source=obj15 icon="tasty.label"
7. **food**  
food=obj15 icon="bread.m3d"
8. **bringing**  
agent=obj4 goal=obj25 theme=obj15
9. **kinship**  
alter=obj25 ego=obj4 icon="grandmother.m3d"

# Discourse is Model: 3D Animation



- Incremental semantic interpretation word-by-word



# Role of PDDL

- Planning Domain Description Language (PDDL)
  - Developed by Drew McDermott for planning competitions
  - Central concepts are OBJECTS and ACTIONS
  - ACTIONS have *precondition* and *effect*
  - Planning problem: given an initial and *goal* states, find a sequence of actions (*plan*) leading from initial to goal state
- PDDL role in Procedural CNL
  - Mapping of FrameNet OBJECTS and sequential SITUATIONS into PDDL language OBJECTS and ACTIONS preserves semantics
  - Planning can be used to fill-in missing actions not mentioned in the text (e.g., to eat an apple, it first needs to be picked up)

TEXT → FrameNetANNOTATION → AnaphoraRESOLUTION → PDDLmapping → 3Danimation

# PDDL: Classic Logistics Example

## Domain description

```
(define (domain logistics-strips)
  (:requirements :strips)
  (:predicates (OBJ ?obj)
    (TRUCK ?truck)
    (LOCATION ?loc)
    (AIRPLANE ?airplane)
    (CITY ?city)
    (AIRPORT ?airport)
    (at ?obj ?loc)
    (in ?obj ?obj)
    (in-city ?obj ?city))

  (:action LOAD-TRUCK
    :parameters
      (?ob ?truc ?loc)
    :precondition
      (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
        (at ?truck ?loc) (at ?obj ?loc))
    :effect
      (and (not (at ?obj ?loc)) (in ?obj ?truck)))

  (:action LOAD-AIRPLANE
    :parameters
      (?ob ?airplan ?loc)
    :precondition
      (and (OBJ ?obj) (AIRPLANE ?airplane)
        (LOCATION ?lo (at ?obj ?loc) (at ?airplane ?loc))
    :effect
      (and (not (at ?obj ?loc)) (in ?obj ?airplane)))

  (:action UNLOAD-TRUCK
    :parameters
      (?obj
        ?truck
        ?loc)
    :precondition
      (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
        (at ?truck ?loc) (in ?obj ?truck))
```

## Planning problem description

```
(define (problem log001)
  (:domain logistics-strips)
  (:objects
    package1
    package2
    package3

    airplane1
    airplane2
    ...
  )
  (:init
    (at package1 pgh-po)
    (at package2 pgh-po)
    (at package3 pgh-po)

    (at airplane1 pgh-airport)
    (at airplane2 pgh-airport)

    (at bos-truck bos-po)
    (at pgh-truck pgh-po)
    (at la-truck la-po)
    ...
  )
  (:goal (and
    (at package1 bos-po)
    (at package2 la-po)
    (at package3 bos-po)
  ))
)
```

## Plan (problem solution)

```
1 (load-truck package2 pgh-truck pgh-po)
1 (drive-truck bos-truck bos-po bos-airport bos)
1 (load-truck package3 pgh-truck pgh-po)
1 (drive-truck la-truck la-po la-airport la)
1 (load-truck package1 pgh-truck pgh-po)
2 (drive-truck pgh-truck pgh-po pgh-airport pgh)
3 (unload-truck package3 pgh-truck pgh-airport)
3 (unload-truck package2 pgh-truck pgh-airport)
3 (unload-truck package1 pgh-truck pgh-airport)
4 (load-airplane package1 airplane1 pgh-airport)
4 (load-airplane package2 airplane2 pgh-airport)
4 (load-airplane package3 airplane1 pgh-airport)
5 (fly-airplane airplane2 pgh-airport la-airport)
5 (fly-airplane airplane1 pgh-airport bos-airport)
6 (unload-airplane package1 airplane1 bos-airport)
6 (unload-airplane package2 airplane2 la-airport)
6 (unload-airplane package3 airplane1 bos-airport)
7 (load-truck package2 la-truck la-airport)
7 (load-truck package1 bos-truck bos-airport)
7 (load-truck package3 bos-truck bos-airport)
8 (drive-truck bos-truck bos-airport bos-po bos)
8 (drive-truck la-truck la-airport la-po la)
9 (unload-truck package3 bos-truck bos-po)
9 (unload-truck package2 la-truck la-po)
9 (unload-truck package1 bos-truck bos-po)
```



# PDDL: FrameNet Example

## Domain description

```
(define (domain framenet)

(:action residence
:parameters
  (?co_resident ?location ?resident)
:effect
  (residence ?co_resident ?location ?
resident))

(:action bringing
:parameters
  (?agent ?goal ?theme)
:precondition
  (in ?theme ?agent)
:effect
  (and (at ?agent ?goal) (at ?
theme ?goal) ))

(:action people
:parameters
  (?person ?sprite)
:effect
  (sprite ?person ?sprite))
```

## Plan (extracted directly from the input text)

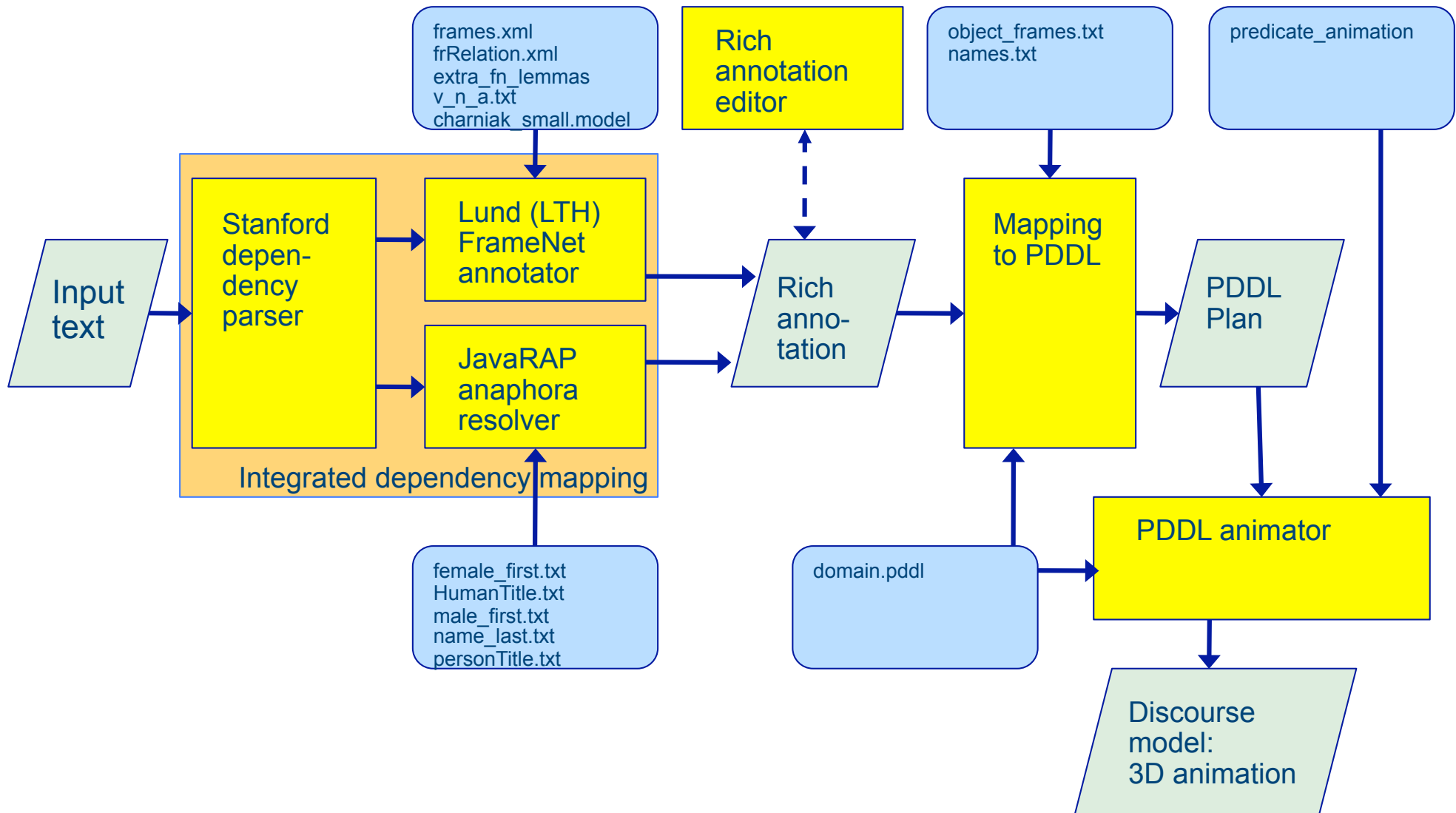
```
1: people obj4 "littleredridinghood"
2: residence obj11 obj8 obj4
3: biological_area obj8 "wood"
4: kinship obj11 obj4 NULL "mother"
5: cooking_creation obj4 obj17 NULL
6: chemical-sense_description obj17 NULL "tasty"
7: food NULL obj17 "bread"
8: bringing obj4 obj25 obj17
9: kinship obj25 obj4 NULL "grandmother"
```

## Planning problem description – not used\* in Proceural CNL

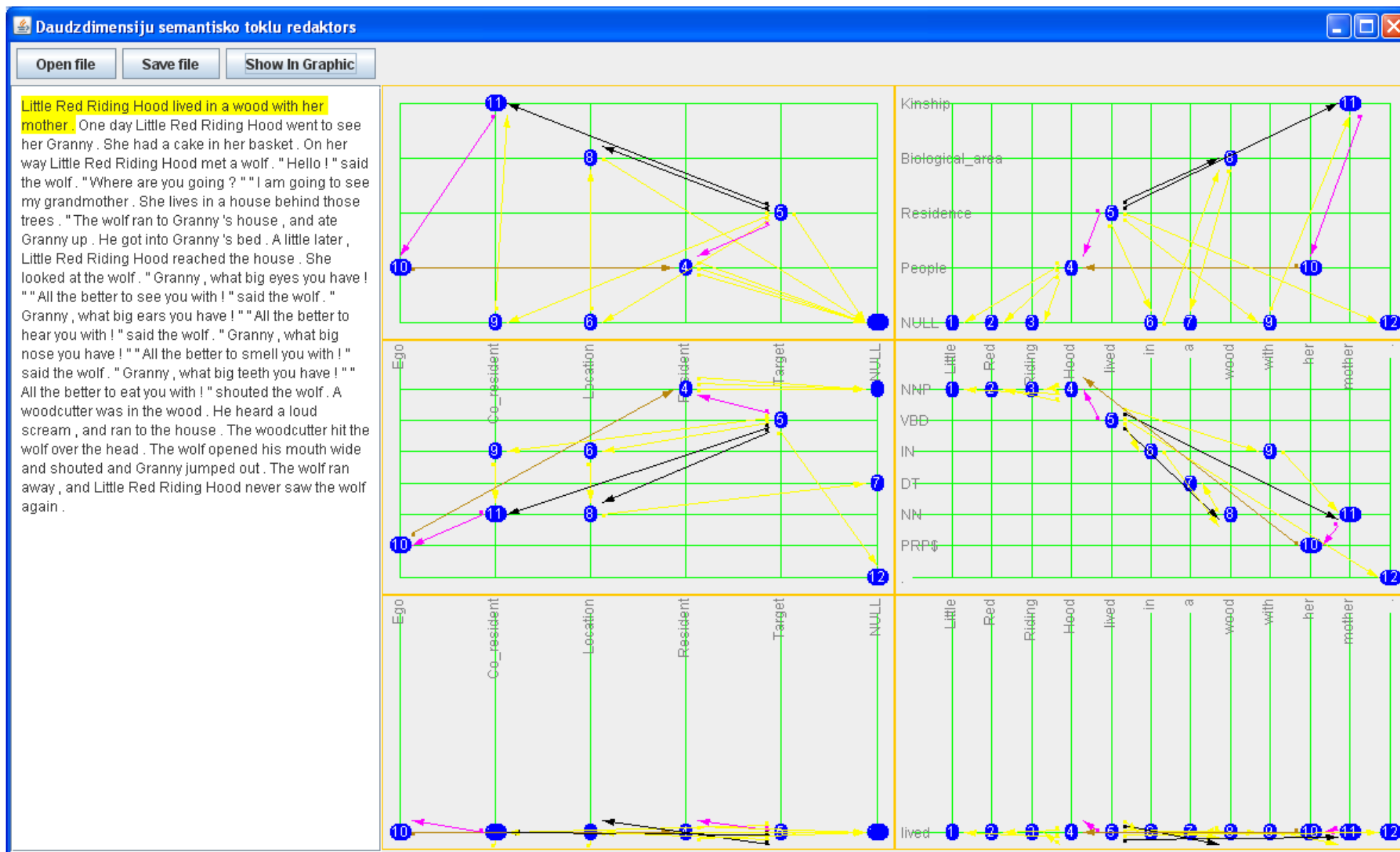
One could envision a special PlanningDomainDescription CNL

\* - micro-planning: to eat an apple, it first needs to be picked up

# Proof-of-concept Implementation (not yet a truly “controlled” NL)



# Rich Annotation Editor




# Discussion

- How to integrate Declarative and Procedural CNL?
  - Syntactically: add ACE functional words, predictive parser
  - Semantically: ACE/OWL classes, properties define icons for objects and their static relationships (“A is a mother of B”). OWL constraints remain as invisible rules, which should be checked after each planned action. FOL model builder could generate objects and their relationships.
- How to implement reasoning in Procedural CNL?
  - Spatial, temporal conceptualisation (“vision”) – check, whether the generated 3D animation includes a scene triggering perception of the queried situation
    - “Did LittleRedRidingHood visited her grandmother?”
    - “Did grandmother got some bread at the end?”
- Potential applications: control of devices
  - Especially, with the help of visual feedback

# Polysemy summary

- To remain “natural”, a multi-domain CNL must support ambiguity in the form of (controlled) polysemy
  - library [collection], library [building], live [residence],...
  - Ambiguity can be resolved through domain identification
    - micro-ontologies, FrameNet frames, Wittgenstein’s communication games, etc.
- For domain-concept naming, natural language relies on heavy reuse of “small” set of well-known words
  - Through multiword-units, metaphors, metonymy

( bird +  mountain =  island)

	→			山		mountain
	→			鳥		bird
	→			島		island

# Thank you!