FrameNet Resource Grammar Library for GF

(FN-CNL)

Normunds Grūzītis, Pēteris Paikens, Guntis Bārzdiņš

Institute of Mathematics and Computer Science,
University of Latvia

3rd Workshop on Controlled Natural Language (CNL), 29–31 August 2012, Zurich

Our Goal

- To develop a wide coverage CNL
  1. For paraphrasing NL in an unambiguous CNL
  2. For semantically correct translation of CNL paraphrases (a multilingual CNL)
  3. For question answering (about the discourse conveyed by the CNL text)
- This is a complex task – we build on two resources
  A. FrameNet (FN)
  B. Grammatical Framework (GF)
- The result we call FrameNet-CNL (FN-CNL)

Procedural ACE/OWL (PAO CNL) in: Polysemy in Controlled Natural Language Texts, CNL-2009 Workshop, LNCS/LNAI 5972, Springer, Heidelberg, 2010, pp. 102–120 (Gruzitis N. and Barzdins G.)

FN-CNL example: Sophie's world (first sentences)

FN-CNL AbstractSyntax ⟷ FN-CNL Multilingual ConcreteSyntax

NL text	Objects	FN Events	GF-EN Paraphrase	GF-LV Paraphrase
Sophie Amundsen was on her way home from school.	X1:Sophie Amundsen; X72:home; X73:school; X3:way;	E1:Self_motion(self_mover:X1; source:X73; goal:X72; path:X3)	E1:Sophie Amundsen moved from school to home.	E1:Sofija Amundsena pārvietojās no skolas uz mājām
She had walked the first part of the way with Joanna.	X4: the first part of X3; X5:Joanna;	E2: Self_motion(self_mover:X1; path:X4; co_theme:X5; time:during E1)	E2:During E1 the first part of the way Sophie Amundsen walked with Joanna.	E2: E1 laikā ceļa pirmo pusi Sofija Amundsena gāja kopā ar Jūrunu.
They had been discussing robots.	X6: robots;	E3: Discussion (interlocutors: X1,X5; topic:X6; time:during E2)	E3:During E2 Sophie Amundsen and Joanna discussed robots.	E3: E2 laikā Sofija Amundsena un Jūruna apsprieda robotus.
Joanna thought		E4:Opinion(cognizer:X5; opinion:E5; time:during E3)	E4:During E3 Joanna stated E5.	E4: E3 laikā Jūruna apgalvoja E5.
the human brain was like an advanced computer.	X7:the human brain; X8: an advanced computer;	E5: Similarity (entity1:X7; entity2:X8)	E5:The human brain is similar to an advanced computer.	E5: Cilvēka smadzenes ir līdzīgas sarežģītam datoram.

QuestionAnswering (FN-CNL FormalSemantics)

Was Joanna at school?	Yes.
Is human brain similar to a computer?	I do not know, but Joanna thinks so.

Grammatical Framework (GF)

- A toolbox for creation of multilingual CNLs
  A. Includes a predefined syntactic interlingua (common "abstract grammar”)
  B. Includes "resource grammars" for translating core syntax of English and 20 other languages to/from the "abstract grammar"
  C. Includes a parser and generator (largely equivalent to Prolog DCG for CFG)
- Limitation – a **purely syntactical framework**. Lexical and frame semantics must be coded into each GF-application from scratch

FrameNet (FN)

- Focused on lexical and frame semantics
  A. Identifies ~1000 **frames**: prototypical situations (coarse, language independent) with participating semantic roles (**frame-elements**, FE)
  B. Identifies **lexical-units** invoking each frame and their syntactical valence patterns for realizing FEs
  C. Based on true corpus evidence in several languages
- Limitation – not entirely formal. Difficult to map towards sentence grammar, except for **verb core valences**. Full-text FrameNet annotation is highly overlapping (not tree-like)

We are primarily interested in verb frames and their core FEs
(non-core FEs are largely same for all verbs and thus can be treated as verb-independent modifiers)

GF Resource Grammar Library (RGL)

- GF facilitates **reusability** by splitting the grammar development in two levels:
  - General-purpose ***resource grammars*** cover morphological paradigms and the realization of syntactic structures
    - Require in-depth linguistic knowledge about each particular language, as well as in-depth GF knowledge
    - Implement a common **syntactic API**
  - Domain-specific ***application grammars*** (CNLs) are built on top of resource grammars
    - However, application grammar developers still have to manipulate with **syntactic constructors**

Proposal: FrameNet RGL

- Built on top of the syntactic GF RGL and FrameNet
  - A common **semantic interlingua** (API)
  - Provides **mapping** from semantic valences to their syntactic realization
- Application grammar (CNL) developers would manipulate with **semantic constructors** (predicates)
  - **Predicates**: domain and language independent
    - The robust verb frames
  - **Arguments**: domain and language specific
    - Frame core elements; remains the current approach

Use-Case: MOLTO Phrasebook

- Precise translation of standard touristic phrases
- Implemented in **15+** languages
- Defines ~40 categories and ~300 functions
  - 20+ "actions" ≈ **frames** (`ALive`, `ALike`, `AWant`, `AWantGo` etc.)
    - *I live in Belgium[.]*
    - *I want a pizza[.]*
    - *[Do] you like this pizza[?]*
    - *I want to go to a museum[.]*

Abito in Belgio[.]
(Italian)

```
                PSentence : Phrase
                          |
                  SProp : Sentence
                          |
             PropAction : Proposition
                          |
Residence      ALive : Action
                  /        \
IMale : Person        Belgium : Country
Resident              Location
```

Phrasebook: Abstract & Common Concrete

```
cat -- abstract categories
  Action  ;
  Country ;
  Person  ;
fun -- abstract functions
  IMale   : Person  ;
  Belgium : Country ;
  ALive   : Person -> Country -> Action ;
```

```
lincat -- category linearization types
  Action  = Cl ; -- clause
  Country = NP ;
  Person  = {name : NP ; isPron : Bool ; poss : Quant} ;
lin -- function linearization rules
  IMale, IFemale = mkPerson Syntax.i_Pron ;
```

RGL API: Syntactic Constructors

Function	Arguments	Value	Example
mkCl	NP VP	Cl	*she always sleeps*
mkCl	NP V2 NP	Cl	*she loves him*
mkCl	NP VV VP	Cl	*she wants to sleep*
...			
mkVP	VP Adv	VP	*to sleep here*
...			
mkNP	Det CN	NP	*the old man*
mkNP	PN	NP	*Paris*
mkNP	Pron	NP	*we*
...			
mkCN	N	CN	*house*
...			
mkAdv	Prep NP	Adv	*in the house*

Phrasebook: English

```
Belgium = mkNP (mkPN "Belgium") ;
Museum = mkPlaceKind "museum" "at" ; -- {name:NP ; at:Adv ; to:Adv}
Pizza = mkCN (mkN "pizza") ;
```

```
-- Cl -> NP VP // VP -> VP Adv // Adv -> Prep NP
ALive pers country = mkCl pers.name
  (mkVP (mkVP (mkV "live")) (mkAdv SyntaxEng.in_Prep country)) ;

-- Cl -> NP V2 NP
ALike pers item = mkCl pers.name (mkV2 (mkV "like")) item ;

-- Cl -> NP V2 NP
AWant pers obj = mkCl pers.name (mkV2 (mkV "want")) obj ;

-- Cl -> NP VV VP // VP -> VP Adv
AWantGo pers place = mkCl pers.name SyntaxEng.want_VV
  (mkVP (mkVP IrregEng.go_V) place.to) ;
```

Observations

- When one gets used to..
  - the syntactic API
  - the best practices, trade-offs and workarounds
- ..it becomes a rather routine work (for a native or fluent speaker) to "copy-paste-edit" the clause level patterns
  - among functions, applications and (partly) even languages
  - providing a miniature domain-specific framenet for each application
- But beware of "exceptions": verb-dependent realizations of clauses (e.g. *love* vs. *like* in Russian Phrasebook)
  - *Я*$_{[NOM]}$ ***люблю*** *тебя*$_{[ACC]}$ (*I love you*)
  - *Я*$_{[NOM]}$ ***нравлю*** *эту пиццу*$_{[ACC]}$ → *Мне*$_{[DAT]}$ ***нравится*** *эта пицца*$_{[NOM]}$
    (**I am liked by this pizza** → *I like this pizza*)

FrameNet Data

- E.g. the Residence frame:
  - Core elements: Resident, Co_resident, Location
  - Lexical units: *camp*, *dwell*, *inhabit*, *live*, *lodge*, *occupy*, ..., *stay*
- E.g. the lexical entry Residence.live:

FE	Total	Pattern
Resident	143	NP.Ext (90%)
Co_resident	14	PP.Dep (86%)
Location	131	PP.Dep (81%)

PP.Dep (Co_resident):

with 9
among 3

PP.Dep (Location):

in 71
on 8
...

Total	Patterns		
98	**Resident**		**Location**
71%	NP.Ext		PP.Dep
7	**Resident**	**Co_resident**	
86%	NP.Ext	PP.Dep	
7	**Resident**	**Co_resident**	**Location**
86%	NP.Ext	PP.Dep	PP.Dep

Assumptions

- There is a common syntactic realization of a frame that is reused by **most** verbs that evoke the frame
  - There can be **alternative** realizations that are specific to particular verbs or groups of verbs (systematic exceptions)
  - Prepositions, in general, do not depend on the frame, but often there is a **dominant** preposition per frame element (if it is realized as a PP)
- It is possible to choose a **default** lexical unit (LU) per frame to be used in the linearization, if a specific verb is not provided
  - The most general and/or the most frequently used LU
- In the CNL settings, it is often sufficient that only **core** valences (according to FrameNet) are available

Semantic vs. Syntactic API

- **ALive** *p co* =
  - **mkCl** *p*.name (**mkVP** (**mkVP** (**mkV** "live")) (**mkAdv** **in_Prep** *co*))
  - **mkCl** *p*.name (**mkVP** (**mkVP** (**mkV** "abitare")) (**mkAdv** **in_Prep** *co*))
  - **Residence** *p*.name NIL *co*
    Resident **Co_resident** **Location**
- **ALike** *p it* =
  - **mkCl** *p*.name (**mkV2** (**mkV** "like")) *it*
  - **mkCl** *it* (**mkV2** (**mkV** (**piacere_64** "piacere")) **dative**) *p*.name
  - **Experiencer_focus** "piacere" *p*.name *it* NIL NIL
    LU **Experiencer** **Content** **Event** **Topic**
- **AWantGo** *p pl* =
  - **mkCl** *p*.name **want_VV** (**mkVP** (**mkVP** IrregEng.**go_V**) *pl*.to)
  - **mkCl** *p*.name **want_VV** (**mkVP** (**mkVP** LexiconIta.**go_V**) *pl*.to)
  - **Desiring** *p*.name (**Motion** **go_V**.s NIL NIL *pl*.name)
    Experiencer **Event** **LU** **Theme** **Source** **Goal**

FrameNet RGL: Implementation

English resource grammar

```
Residence res NIL loc =                         -- NP NIL NP
  mkCl res (mkVP (mkVP (mkV "live"))
    (mkAdv  in_Prep  loc)) ;                    -- “in”: 67%

Residence lu res NIL loc =                      -- V NP NIL Adv
  mkCl res (mkVP (mkVP (mkV lu)) loc) ;

Residence lu res co_res loc =                   -- V NP Adv Adv
  mkCl res (mkVP (mkVP (mkVP (mkV lu)) co_res) loc) ;
```

Italian RG

```
Experiencer_focus lu exp cont NIL NIL = case lu of {
  "amare"    => mkCl exp (mkV2 (mkV "amare")) cont ;
  "piacere" =>
    mkCl cont (mkV2 (mkV (piacere_64 "piacere")) dative) exp ;
} ;
```

FrameNet RGL API: Semantic Constructors

Function (FN frame)	Arguments	Value	Mapping to FEs
Residence	NP NIL NP	Cl	Resident Co_resident Location
	Str NP NIL Adv	Cl	
	Str NP Adv Adv	Cl	
Experiencer_focus	NP NP NIL NIL	Cl	Experiencer Content Event Topic
	Str NP NP NIL NIL	Cl	
	...		
Motion	NP NP NP	Cl	Theme Source Goal
	Str NP NP NP	Cl	
	Str NIL NIL NP	VP	
Desiring	NP VP	Cl	Experiencer Event
	Str NP VP	Cl	
...			

FrameNet RGL API: Next Step

Function (FN frame)	Arguments	Value
Residence	Resident Location	Cl
	Str Resident Location	Cl
	Str Resident Co_resident Location	Cl
Experiencer_focus	Experiencer Content	Cl
	Str Experiencer Content	Cl
	...	
Motion	Theme Source Goal	Cl
	Str Theme Source Goal	Cl
	Str Goal	VP
Desiring	Experiencer Event	Cl
	Str Experiencer Event	Cl
...		

FrameNet RGL: Ongoing Work

- Language-specific realization of CNL clauses can be hidden behind the language-independent FrameNet API
  - Frames can be specified already in the common concrete syntax
  - Resulting grammars would more generic and easier to extend
- Language-specific FrameNet resource grammars can be acquired semi-automatically from FrameNet data that include mapping to syntactic patterns and statistics from FrameNet-annotated corpora
- Abstract syntax trees could be annotated with FrameNet
  - Parsing with FrameNet RGL = semantic parsing (semantic role labeling)

FN-CNL example: Sophie's world (first sentences)

FN-CNL AbstractSyntax ⟷ FN-CNL Multilingual ConcreteSyntax

NL text	Objects	FN Events	GF-EN Paraphrase	GF-LV Paraphrase
Sophie Amundsen was on her way home from school.	X1:Sophie Amundsen; X72:home; X73:school; X3:way;	E1:Self_motion(self_mover:X1; source:X73; goal:X72; path:X3)	E1:Sophie Amundsen moved from school to home.	E1:Sofija Amundsena pārvietojās no skolas uz mājām
She had walked the first part of the way with Joanna.	X4: the first part of X3; X5:Joanna;	E2: Self_motion(self_mover:X1; path:X4; co_theme:X5; time:during E1)	E2:During E1 the first part of the way Sophie Amundsen walked with Joanna.	E2: E1 laikā ceļa pirmo pusi Sofija Amundsena gāja kopā ar Jūrunu.
They had been discussing robots.	X6: robots;	E3: Discussion (interlocutors: X1,X5; topic:X6; time:during E2)	E3:During E2 Sophie Amundsen and Joanna discussed robots.	E3: E2 laikā Sofija Amundsena un Jūruna apsprieda robotus.
Joanna thought		E4:Opinion(cognizer:X5; opinion:E5; time:during E3)	E4:During E3 Joanna stated E5.	E4: E3 laikā Jūruna apgalvoja E5.
the human brain was like an advanced computer.	X7:the human brain; X8: an advanced computer;	E5: Similarity (entity1:X7; entity2:X8)	E5:The human brain is similar to an advanced computer.	E5: Cilvēka smadzenes ir līdzīgas sarežģītam datoram.

QuestionAnswering (FN-CNL FormalSemantics)

Was Joanna at school?	Yes.
Is human brain similar to a computer?	I do not know, but Joanna thinks so.

FN-CNL Formal Semantics

- Semantic gap between NL and FN-CNL is bridged by
  A. FrameNet annotation
  B. Anaphora tracking for objects
  C. Chronology tracking

- The key observation: verb frames are largely universal – domain-specific are only objects (NPs)
  – Decoupling via subPropertyOf mechanism:
    "I went to kitchen" / "I went to Zurich"
    household domain   travel domain

Commercial Transaction Scenario

RDF-triples (micro-relations)

The event type known as Commercial Transaction is a small abstract bit of history in which:

- **Stage 1**: *B wants G, owns M; S wants M, owns G.*
- **Transition**: by explicit or implicit contract -
  (a) *B gives M to S;*
  (b) *S gives G to B*
- **Stage 2**: *B owns G; S owns M*

Time (sequence of states)

The verb *pay* evokes this frame; in particular, it evokes the bit about the buyer handing the money over to the seller. The question

How much did Will pay for his laptop?

invites us to have in mind an instance of this frame in which

Will is B,
the laptop is G,
the seller, S, is irrelevant, and
the requested information concerns M.

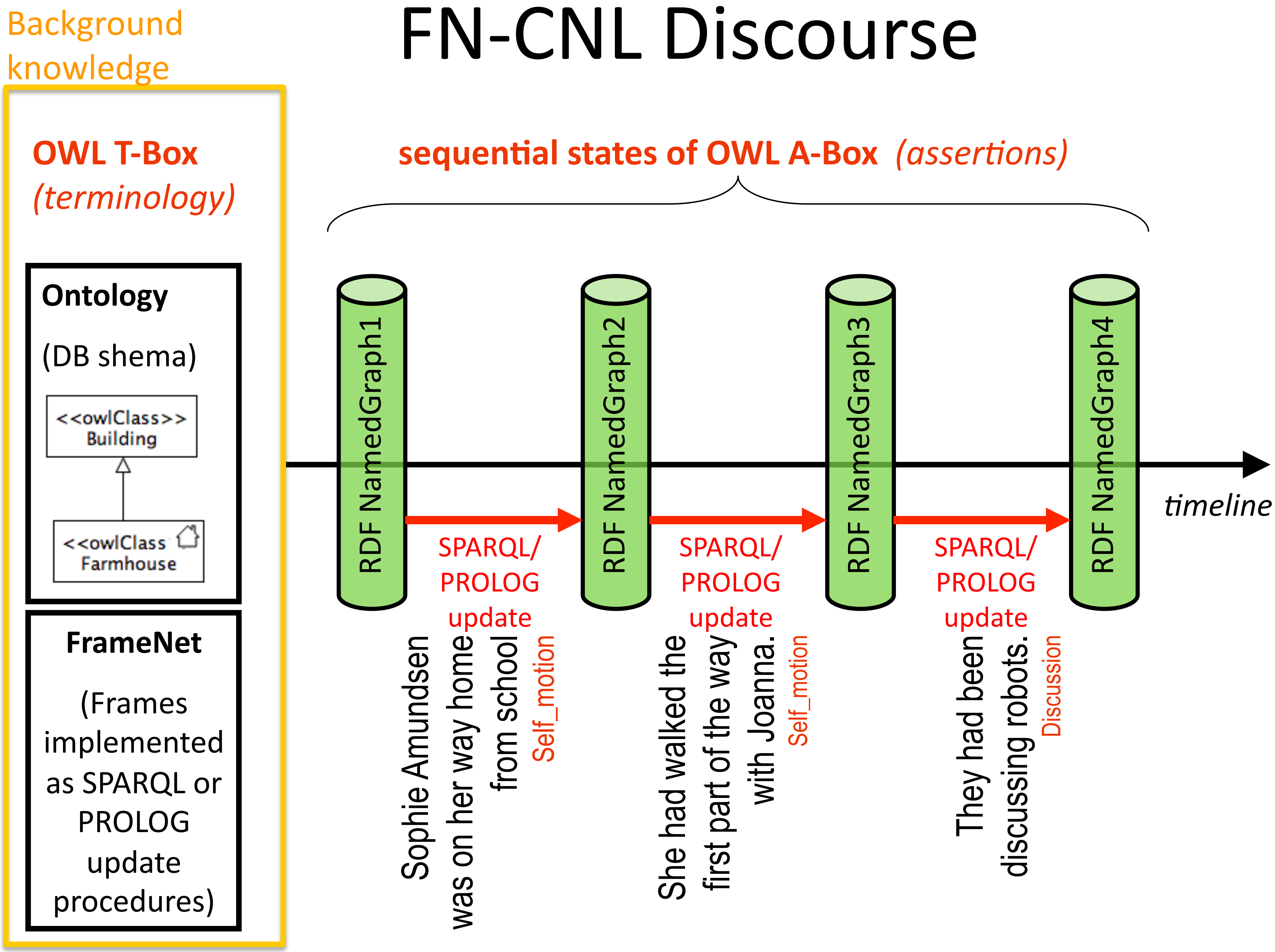
Procedure implementing this Frame

A slide from C.J. Fillmore presentation at FrameNet Masterclass (TLT-8, Milano, 2009)

Formalizing FN-CNL Semantics

- Formal FN-CNL semantics based on
  A. Frames implemented as stepwise procedures creating and deleting **RDF-triples** (micro-relations)
  B. Anaphors are typed in **domain ontology**
     - Domain-ontology may include inference rules/ procedures going beyond DL limitations
  C. The **discourse** is history of all **steps in time**, modelled as a sequence of RDF named-graphs

FN-CNL Discourse

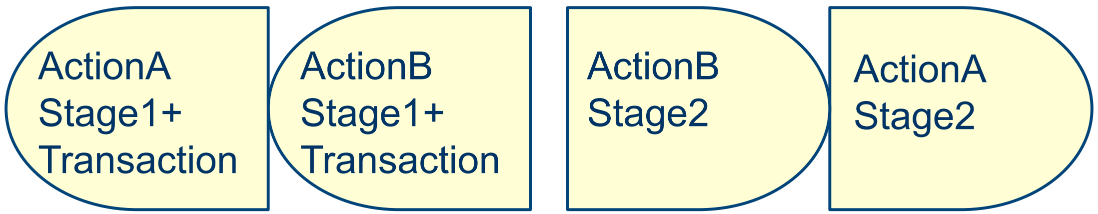


Self_motion frame implementation

- Prolonged events implemented as two steps:
  1. STAGE1 + TRANSACTION
  2. STAGE2
- Splitting of prolonged events in two steps enables implementation of “during” temporal relation

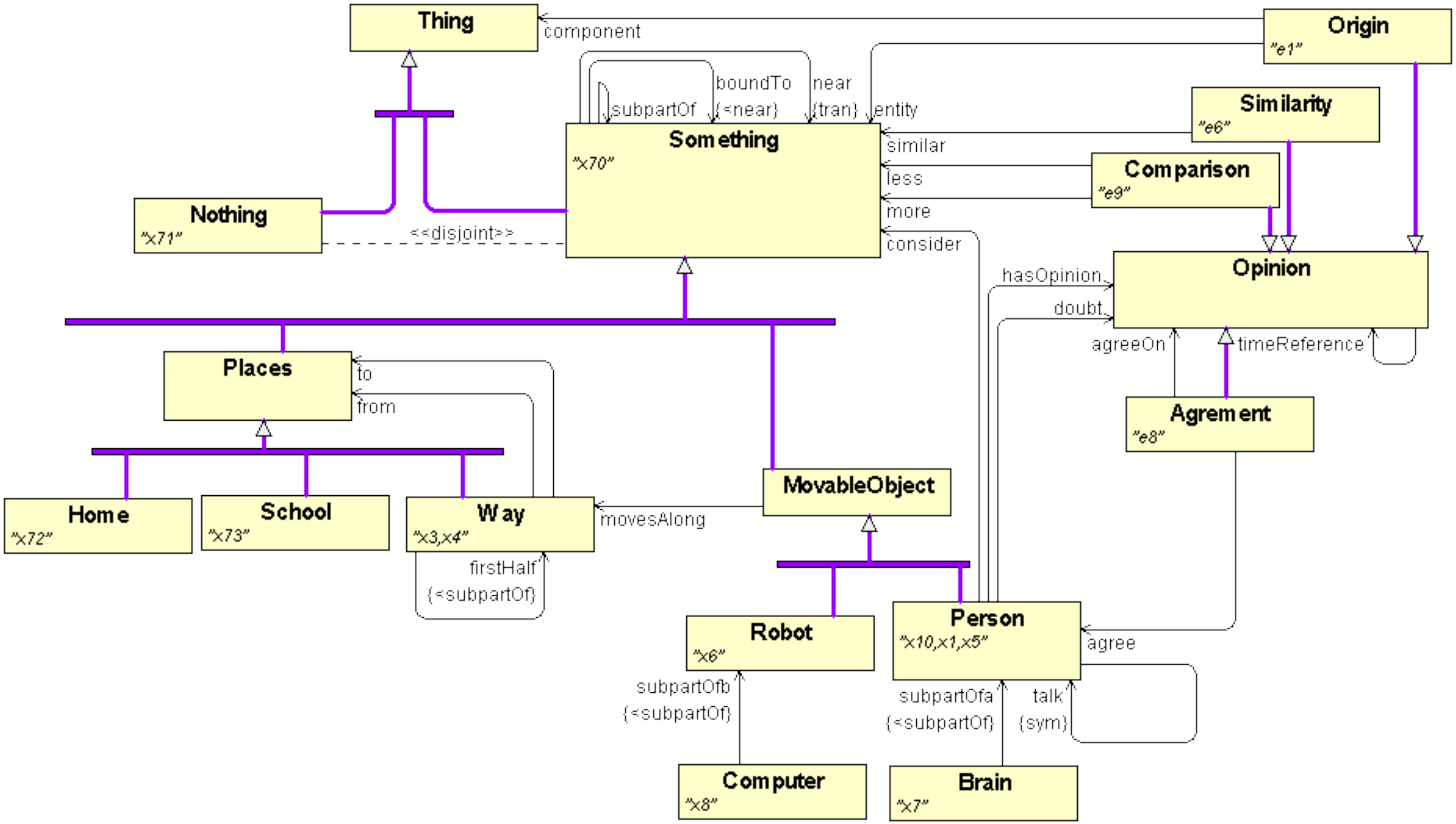
```
selfMotion(1,GR:N,SelfMover,Goal,Source,Path,Cotheme) :-
    rdf_assert(SelfMover, movesAlong,Path,GR:N),
    rdf_assert(Path, from,Source,GR:N),
    rdf_assert(Path, to,Goal,GR:N),
    rdf_assert(Cotheme,boundTo,SelfMover,GR:N).

selfMotion(2,GR:N,SelfMover,Goal,Source,Path,Cotheme) :-
    rdf_retractall(SelfMover,movesAlong,Path,GR:N),
    rdf_retractall(Cotheme,boundTo,SelfMover,GR:N),
    rdf_retractall(SelfMover,near,Source,GR:N),
    rdf_assert(SelfMover, near,Goal,GR:N).
```



Sophie's world domain-ontology

(enables additional inferences over discourse)



Plural properly shall be treated as subclasses (with existential quantification) or whole class (with huniversal quantification). This primarilly applies to ontological sentences covered by ACE/OWL.

Ontology visualised by http://owlgred.lumii.lv/

Question answering over discourse: SPARQL+Reasoner

- Question in FN-CNL
  - “Was Joanna(x5) at school(x73)?”
- Question in SPARQL
  - SELECT ?step
    WHERE { GRAPH ?step {x5 near x73}}
- Answer
  - ?step = “b:10”
  - This means “Yes, there was step b:10 when Joanna was at school”

```
C:\pellet-2.3.0>pellet query  --query-file gbquery.txt rez.owl >reasongb.txt
Query Results:


step
==========================================================================
b:10
```

This Pellet example is constructed, because since StarDog introduction Pellet does not support NamedGraphs in SPARQL

Conclusions and Future work

- FN-CNL is manually tested on few examples, full evaluation requires implementation:
  - Mapping between FN-CNL abstract-syntax and multilingual concrete syntaxes in GF can be semi-automatically generated from FrameNet data (moderate amount of work)
  - Implementation of formal semantics requires defining RDF-triple interpretation procedures for all FrameNet frames (large amount of work)

Backup slides

Frame and Lexical Unit example

Residence

Lexical Unit Index

Definition:

This frame has to do with people (the Residents) residing in Locations, sometimes with a Co-resident.

Peter LIVES in New York.

Sue *is* an INHABITANT of Los Angeles.

FEs:

Core:

Co_resident [Co-resident] — A person or group of people that the resident is staying with or among.
Boris still LIVES with his parents.

Location [Location] — The place in which somebody resides.
Semantic Type: Location
Sue LIVES in Berkeley.

Resident [Res] — The individual(s) that reside at the Location.
Hannah LIVES in San Francisco.

live.v

Frame: Residence

Definition:

COD: make one's home in a particular place or with a particular person

Frame Elements and Their Syntactic Realizations

The Frame Elements for this word sense are (with realizations):

Frame Element	Number Annotated	Realization(s)
Co_resident	(11)	PP[among].Dep (3) PP[with].Dep (8)
Location	(60)	PP[in].Dep (27) PP[above].Dep (4) PP[on].Dep (5) AJP.Dep (1) AVP.Dep (6) NP.Ext (1) PP[across].Dep (3) PP[amongst].Dep (1) PP[around].Dep (3) PP[at].Dep (4) PP[down].Dep (3) PP[of].Dep (1) PP[outside].Dep (1)
Resident	(65)	NP.Ext (63) CNI.-- (2)

Valence Patterns:

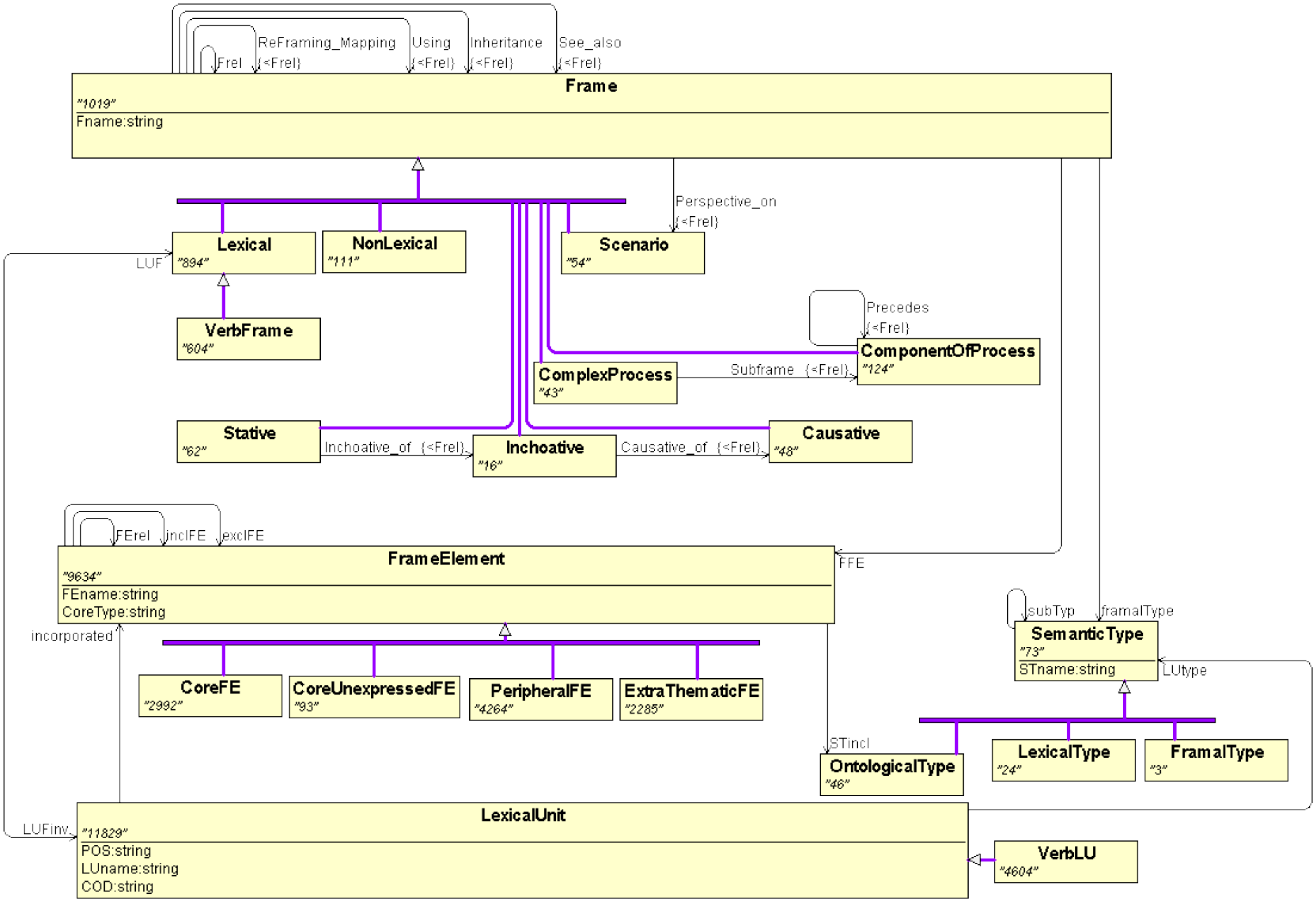
These frame elements occur in the following syntactic patterns:

Number Annotated	Patterns		
6 TOTAL	Co_resident	Location	Resident
(1)	PP[among] Dep	PP[in] Dep	NP Ext
(1)	PP[with] Dep	PP[above] Dep	NP Ext

We are mostly interested in verb Frames and their Core FEs

(non-core FEs are largely same for all verb frames and thus can be treated independently of specific verbs)

Size of FrameNet 1.5 (basis of FN-CNL)



GF mini-grammar as Prolog DCG

```
interface Syntax = Grammar -
   [UseCl,PredVP,ComplV2,UseV,DetCN,ModCN,CompAP,AdAP,
    ConjS,ConjNP,UseN,UseA,Pres,Perf,Pos,Neg] **  open Grammar in {
oper
  mkS = overload {
    mkS : Cl -> S = UseCl Pres Pos ;
    mkS : Tense -> Cl -> S = \t -> UseCl t Pos ;
    mkS : Pol -> Cl -> S = UseCl Pres ;
    mkS : Tense -> Pol -> Cl -> S = UseCl ;
    mkS : Conj -> S -> S -> S = ConjS ; } ;
  mkCl = overload {
    mkCl : NP -> V  -> Cl = \np,v -> PredVP np (UseV v) ;
    mkCl : NP -> V2 -> NP -> Cl = \np,v,o -> PredVP np (ComplV2 v o) ;
    mkCl : NP -> A  -> Cl = \np,a -> PredVP np (CompAP (UseA a)) ;
    mkCl : NP -> AP -> Cl = \np,ap -> PredVP np (CompAP ap) ;
    mkCl : NP -> VP -> Cl = PredVP ;  } ;
  mkAP = overload {
    mkAP : A -> AP = UseA ;
    mkAP : AdA -> AP -> AP = AdAP ;  } ;
  mkNP = overload {
    mkNP : Det -> N -> NP = \d,n -> DetCN d (UseN n) ;
    mkNP : Det -> CN -> NP = \d,n -> DetCN d n ;
    mkNP : Conj -> NP -> NP -> NP = ConjNP ; } ;
  mkCN = overload {
    mkCN : N  -> CN = UseN ;
    mkCN : A  -> N  -> CN = \a,n -> ModCN (UseA a) (UseN n) ;
    mkCN : A  -> CN -> CN = \a,n -> ModCN (UseA a) n ;
    mkCN : AP -> N  -> CN = \a,n -> ModCN a (UseN n) ;
    mkCN : AP -> CN -> CN = \a,n -> ModCN a n ; } ;
  presTense : Tense = Pres ;
  perfTense : Tense = Perf ;
  posPol : Pol = Pos ;
  negPol : Pol = Neg ;}
```

```
abstract Grammar = {
  flags startcat = S ;
  cat
    S ; Cl ; NP ; VP ; AP ; CN ; Det ; N ; A ; V ; V2 ; AdA ;  Tense ; Pol ; Conj ;
  data
    UseCl   : Tense -> Pol -> Cl -> S ;
    PredVP  : NP -> VP -> Cl ;
    ComplV2 : V2 -> NP -> VP ;
    DetCN   : Det -> CN -> NP ;
    ModCN   : AP -> CN -> CN ;
    CompAP  : AP -> VP ;
    AdAP    : AdA -> AP -> AP ;
    ConjS   : Conj -> S  -> S  -> S ;
    ConjNP  : Conj -> NP -> NP -> NP ;
    UseV    : V -> VP ;
    UseN    : N -> CN ;
    UseA    : A -> AP ;

    a_Det, the_Det, every_Det : Det ;
    this_Det, these_Det : Det ;
    that_Det, those_Det : Det ;
    i_NP, she_NP, we_NP : NP ;
    very_AdA : AdA ;
    Pos, Neg : Pol ;
    Pres, Perf : Tense ;
    and_Conj, or_Conj : Conj ;}
```

```
concrete TestEng of Test = GrammarEng ** open
ParadigmsEng in { lin
  man_N = mkN "man" "men" ;
  woman_N = mkN "woman" "women" ;
  house_N = mkN "house" ;
  tree_N = mkN "tree" ;
  big_A = mkA "big" ;
  small_A = mkA "small" ;
  green_A = mkA "green" ;
  walk_V = mkV "walk" ;
  arrive_V = mkV "arrive" ;
  love_V2 = mkV2 "love" ;
  please_V2 = mkV2 "please" ;}
```

```
xS([yUseCl,T,P,C]) --> xTense(T), xPol(P), xCl(C).
xS([yUseCl,yPres,P,C]) --> xPol(P), xCl(C).
xS([yUseCl,T,yPos,C]) --> xTense(T), xCl(C).
xS([yUseCl, yPres, yPos,C]) --> xCl(C).
xS([yConjS,C,S1,S2]) --> xConj(C), xS(S1), xS(S2).
xCl([yPredVP,NP,VP]) --> xNP(NP), xVP(VP).
xNP([yDetCN,D,N]) --> xDet(D), xCN(N).
xNP([yConjNP,C,N1,N2]) --> xConj(C), xNP(N1), xNP(N2).
xCN([yModCN,A,N]) --> xAP(A), xCN(N).
xCN([yUseN,N]) --> xN(N).
xVP([yCompAP,AP]) --> xAP(AP).
xVP([yComplV2,V,N]) --> xV2(V), xNP(N).
xVP([yUseV,V]) --> xV(V).
xAP([yAdAP,AdA,AP]) --> xAdA(AdA), xAP(AP).
xAP([yUseA,A]) --> xA(A).

xDet(a_Det) --> [a].
xDet(every_Det) --> [every].
xDet(this_Det) --> [this].
xDet(these_Det) --> [these].
xDet(that_Det) --> [that].
xDet(those_Det) --> [those].
xNP(i_NP) --> [i].
xNP(she_NP) --> [she].
xNP(we_NP) --> [we].
xAdA(very_AdA) --> [very].
xPol(yPos) --> [posPol].
xPol(yNeg) --> [negPol].
xTense(yPres) --> [presTense].
xTense(yPerf) --> [perfTense].
xConj(and_Conj) --> [and].
xConj(or_Conj) --> [or].

xN(man_N) --> [man]; [men].
xN(woman_N) --> [woman]; [women].
xN(house_N) --> [house].
xN(tree_N) --> [tree].
xA(big_A) --> [big].
xA(small_A) --> [small].
xA(green_A) --> [green].
xV(walk_V) --> [walk].
xV(arrive_V) --> [arrive].
xV2(love_V2) --> [love].
xV2(please_V2) --> [please].
```

```
?- xS(A,[i,love,that,very,big,tree],[]),writeln(A),xS(A,B,[]), writeln(B),fail.
[yUseCl,yPres,yPos,[yPredVP,i_NP,[yComplV2,love_V2,[yDetCN,that_Det,[yModCN,[yAdAP,very_AdA,[yUseA,big_A]],[yUseN,tree_N]]]]]]
[i,love,that,very,big,tree]
```

RDF NamedGraphs sequence (discourse)

```
[x1,near,x73,b:7]

[x1,near,x73,b:8]
[x1,movesAlong,x3,b:8]
[x3,from,x73,b:8]
[x3,to,x72,b:8]

[x1,near,x73,b:9]
[x1,movesAlong,x3,b:9]
[x3,from,x73,b:9]
[x3,to,x72,b:9]
[x1,movesAlong,x4,b:9]
[x5,boundTo,x1,b:9]

[x1,near,x73,b:10]
[x1,movesAlong,x3,b:10]
[x3,from,x73,b:10]
[x3,to,x72,b:10]
[x1,movesAlong,x4,b:10]
[x5,boundTo,x1,b:10]
[x1,talk,x5,b:10]
[x1,consider,x6,b:10]
[x5,consider,x6,b:10]

[x1,near,x73,b:11]
[x1,movesAlong,x3,b:11]
[x3,from,x73,b:11]
[x3,to,x72,b:11]
[x1,movesAlong,x4,b:11]
[x5,boundTo,x1,b:11]
[x1,talk,x5,b:11]
[x1,consider,x6,b:11]
[x5,consider,x6,b:11]
[x5,hasOpinion,e6,b:11]

[x1,near,x73,b:12]
[x1,movesAlong,x3,b:12]
[x3,from,x73,b:12]
[x3,to,x72,b:12]
[x1,movesAlong,x4,b:12]
[x5,boundTo,x1,b:12]
[x1,talk,x5,b:12]
[x1,consider,x6,b:12]
[x5,consider,x6,b:12]
[x5,hasOpinion,e6,b:12]
[e6,similar,x7,b:12]
[e6,similar,x8,b:12]

[x1,near,x73,b:13]
[x1,movesAlong,x3,b:13]
[x3,from,x73,b:13]
[x3,to,x72,b:13]
[x1,movesAlong,x4,b:13]
[x5,boundTo,x1,b:13]
[x1,talk,x5,b:13]
[x1,consider,x6,b:13]
[x5,consider,x6,b:13]
[x5,hasOpinion,e6,b:13]
[e6,similar,x7,b:13]
[e6,similar,x8,b:13]
[x1,doubt,e8,b:13]

[x1,near,x73,b:14]
[x1,movesAlong,x3,b:14]
[x3,from,x73,b:14]
[x3,to,x72,b:14]
[x1,movesAlong,x4,b:14]
[x5,boundTo,x1,b:14]
[x1,talk,x5,b:14]
[x1,consider,x6,b:14]
[x5,consider,x6,b:14]
[x5,hasOpinion,e6,b:14]
[e6,similar,x7,b:14]
[e6,similar,x8,b:14]
[x1,doubt,e8,b:14]
[e8,agreeOn,e6,b:14]
[e8,agree,x1,b:14]
[e8,agree,x5,b:14]

[x1,near,x73,b:15]
[x1,movesAlong,x3,b:15]
[x3,from,x73,b:15]
[x3,to,x72,b:15]
[x1,movesAlong,x4,b:15]
[x5,boundTo,x1,b:15]
[x1,talk,x5,b:15]
[x1,consider,x6,b:15]
[x5,consider,x6,b:15]
[x5,hasOpinion,e6,b:15]
[e6,similar,x7,b:15]
[e6,similar,x8,b:15]
[x1,doubt,e8,b:15]
[e8,agreeOn,e6,b:15]
[e8,agree,x1,b:15]
[e8,agree,x5,b:15]
[e9,more,x9,b:15]
[e9,less,x10,b:15]

[x1,near,x73,b:16]
[x1,movesAlong,x3,b:16]
[x3,from,x73,b:16]
[x3,to,x72,b:16]
[x1,movesAlong,x4,b:16]
[x5,boundTo,x1,b:16]
[x1,talk,x5,b:16]
[x1,consider,x6,b:16]
[x5,consider,x6,b:16]
[x5,hasOpinion,e6,b:16]
[e6,similar,x7,b:16]
[e6,similar,x8,b:16]
[x1,doubt,e8,b:16]
[e8,agreeOn,e6,b:16]
[e8,agree,x1,b:16]
[e8,agree,x5,b:16]
[e9,more,x9,b:16]
[e9,less,x10,b:16]
[x1,hasOpinion,e9,b:16]

[x1,near,x73,b:17]
[x1,movesAlong,x3,b:17]
[x3,from,x73,b:17]
[x3,to,x72,b:17]
[x1,movesAlong,x4,b:17]
[x5,boundTo,x1,b:17]
[x5,hasOpinion,e6,b:17]
[e6,similar,x7,b:17]
[e6,similar,x8,b:17]
[x1,doubt,e8,b:17]
[e8,agreeOn,e6,b:17]
[e8,agree,x1,b:17]
[e8,agree,x5,b:17]
[e9,more,x9,b:17]
[e9,less,x10,b:17]
[x1,hasOpinion,e9,b:17]

[x1,near,x73,b:18]
[x1,movesAlong,x3,b:18]
[x3,from,x73,b:18]
[x3,to,x72,b:18]
[x5,hasOpinion,e6,b:18]
[e6,similar,x7,b:18]
[e6,similar,x8,b:18]
[x1,doubt,e8,b:18]
[e8,agreeOn,e6,b:18]
[e8,agree,x1,b:18]
[e8,agree,x5,b:18]
[e9,more,x9,b:18]
[e9,less,x10,b:18]
[x1,hasOpinion,e9,b:18]

[x3,from,x73,b:19]
[x3,to,x72,b:19]
[x5,hasOpinion,e6,b:19]
[e6,similar,x7,b:19]
[e6,similar,x8,b:19]
[x1,doubt,e8,b:19]
[e8,agreeOn,e6,b:19]
[e8,agree,x1,b:19]
[e8,agree,x5,b:19]
[e9,more,x9,b:19]
[e9,less,x10,b:19]
[x1,hasOpinion,e9,b:19]
[x1,near,x72,b:19]
```

Time