Universität
Konstanz

Grammar Development with LFG and XLE

Miriam Butt
University of Konstanz

Course Content

In this course you will learn:

- How to implement a computational grammar fragment within LFG (Lexical-Functional Grammar).
- Work with the grammar development platform XLE.
- Learn how to integrate finite-state morphological analyzers into the grammar.
- At the end of the course you should be able to begin building your own grammar for a language of your choice.

Course Content

What you will **not** learn:

- The course will introduce basic concepts and standard analyses within LFG.
- But the course will spend no time motivating these analyses or delving into alternatives.
- You will need to take a course on LFG (also available on-line) or work through the relevant literature (provided as part of the course).

Course Structure

- See the accompanying file for a detailed course outline.
- Each lesson contains:

1) introduction of the theoretical concepts
2) demonstrations of how to implement the concepts
3) a sample grammar fragment
4) exercises working with the grammar fragment

Course Topics

1. LFG Basics and First Steps in XLE (Walkthrough)
2. Lexical Rules: Passives and other Argument Alternations
3. Adjuncts: Adverbs and Adjectives (also xlerc file)
4. PPs: Adjuncts and Obliques
5. Generation and the Optimality Projection
6. Complements: XCOMP and COMP
7. Long Distance Dependencies and Functional Uncertainty

Course Topics II

8. Pronouns, Empty Nodes and Punctuation
9. Integrating a Finite-State Morphological Analyzer
10. Further Basic XLE Features
    - Shuffle Operator (Free Word Order)
    - Variables (Relative Clauses)
11. Outlook: Advanced features of XLE
    - Integration of Statistical Preferences
    - Complex Categories, Transfer System, Restriction Operator

Outline, Lesson 1

1. Motivation: Why Deep Grammars?
2. Demos of possible Applications
3. XLE/LFG Basics
4. The ParGram (Parallel Grammars) Collaboration

Why Deep Grammars?

1. Some Background and Motivation
2. Examples of Applications
   - Question-Answering Systems
   - Electronic Look-up/Learning Resources
     - Computer Assisted Language Learning (CALL)
     - Murrinh-Patha English Translation System
   - Automatically Annotated Corpora (Treebanks)

Deep Grammars

- Provide detailed morphosyntactic information
  - Include detailed information about dependencies within the clause (subject, object, clausal complements).
  - Tense/Aspect/Mood
  - Person/Number/Gender
  - Modification, Specification (Definite/Indefinite)
- LFG also provides detailed information about constituency and hierarchical relations of words in the clause.

Deep Grammars

- Simplified Example:

  Mary wants to leave.

  ```
  subj(want~1,Mary~3)
  comp(want~1,leave~2)
  subj(leave~2,Mary~3)
  tense(leave~2,present)
  ```

- Deep Grammars are usually manually constructed.
- Large Multilingual Grammar Development Efforts include:
  - LFG (ParGram)
  - HPSG (LinGO, Matrix, DELPH-IN)

Applications: Treebanks

- Many Natural Language Processing (NLP) tools depend on linguistically sophisticated resources.
- One example of such a resource is a so-called *treebank*.
- A treebank or *structure bank* is a corpus of sentences annotated with deep linguistic information.
- Deep grammars can be used to automatically annotate such a corpus of sentences.
- This also serves to help with *language documentation*.

Applications: Q&A

- Deep grammars also have potential to be used in sophisticated Question-Answer (Q&A) systems.
- Examples:
  - IBM's Watson (uses a dependency grammar)
  - The PARC Bridge System (led to founding of the company Powerset)

IBM's Watson

IBM's Watson made a spectacular debut playing against humans on the US game show Jeopardy.

IBM’s Watson

- Check it out on YouTube.
- Read about the design here (and elsewhere):

  Ferrucci, David, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer & Chris Welty. 2012. Building Watson: An overview of the DeepQA Project. *AI Magazine* 31(3).

PARC’s BRIDGE Q&A

- Researchers at the Palo Alto Research Center (PARC) put together a demo to show-case their ideas for a linguistically sophisticated Q&A system.
- The system includes
  - LFG grammars of the type constructed in this course.
  - Finite-State Morphological Analyzer
  - An Abstract Knowledge Representation System (AKR)
  - Integration of lexical semantic information from WordNet and other lexical resources/ontologies.

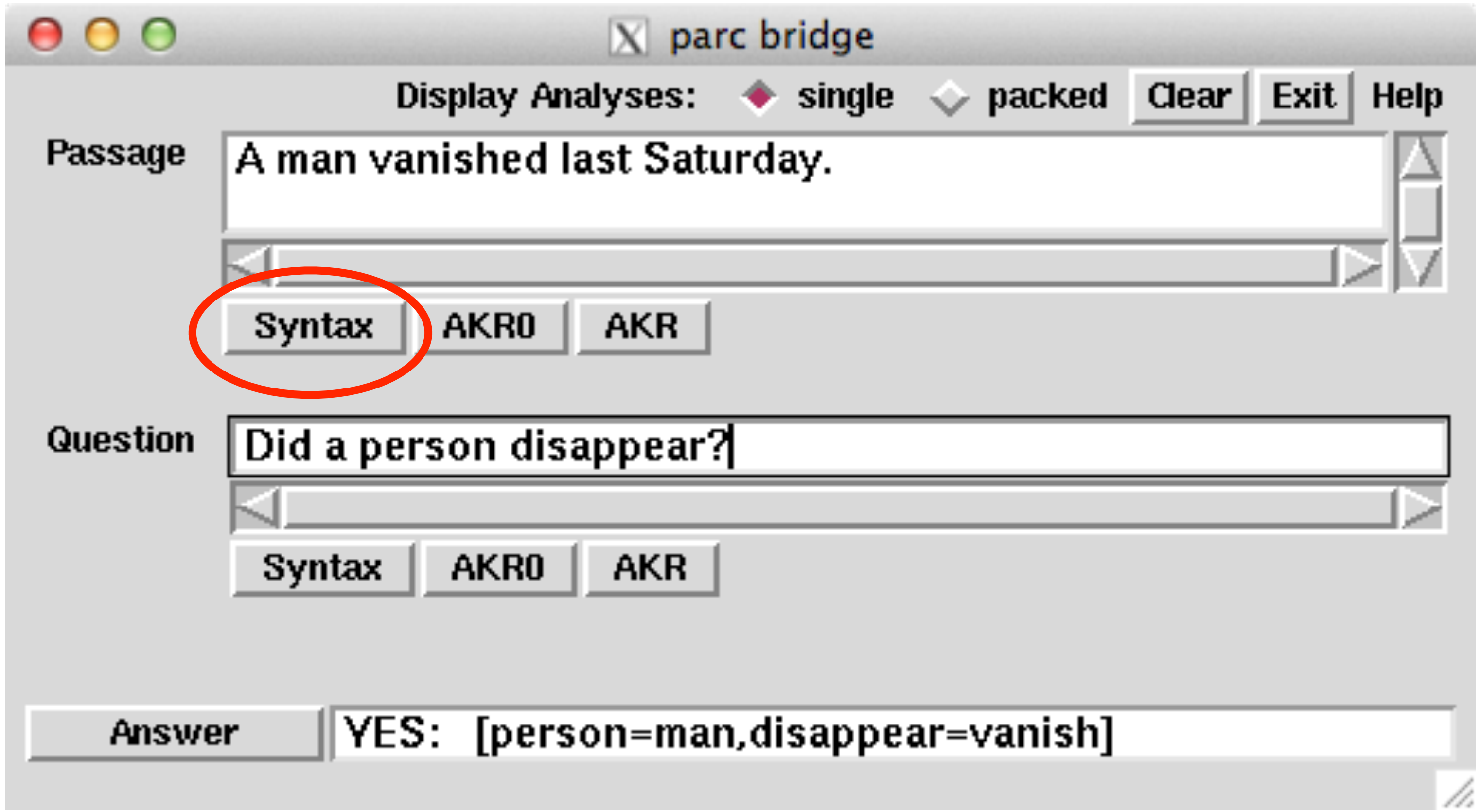
PARC's BRIDGE Q&A

Read about the system here:

Bobrow, Danny G., Cleo Condoravdi, Kyle Richardson, Richard Waldinger & Amar Das. 2011. Deducing answers to English questions from structured data. 2011. *International Conference on Intelligent User Interfaces (IUI)*. Stanford.

Bobrow, Danny G., Robert D. Cheslow, Cleo Condoravdi, Lauri Karttunen, Tracy H. King, Rowan Nairn, Valeria de Paiva, Lotti Price & Annie Zaenen. 2007. PARC's Bridge question answering system. *Proceedings of the GEAF (Grammar Engineering Across Frameworks) 2007 Workshop*. Stanford.

PARC's BRIDGE Q&A

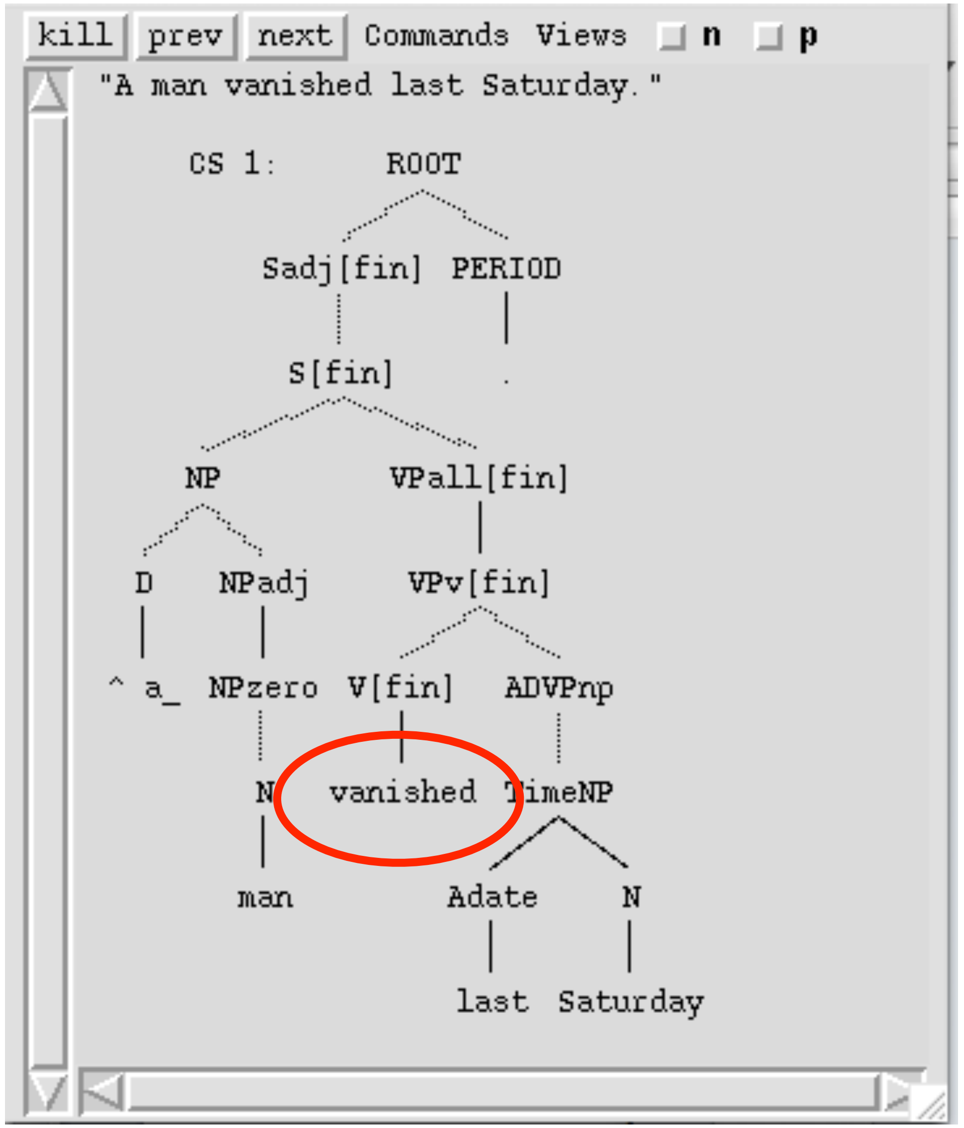


Passage: A man vanished last Saturday.

Question: Did a person disappear?

Answer: YES: [person=man,disappear=vanish]

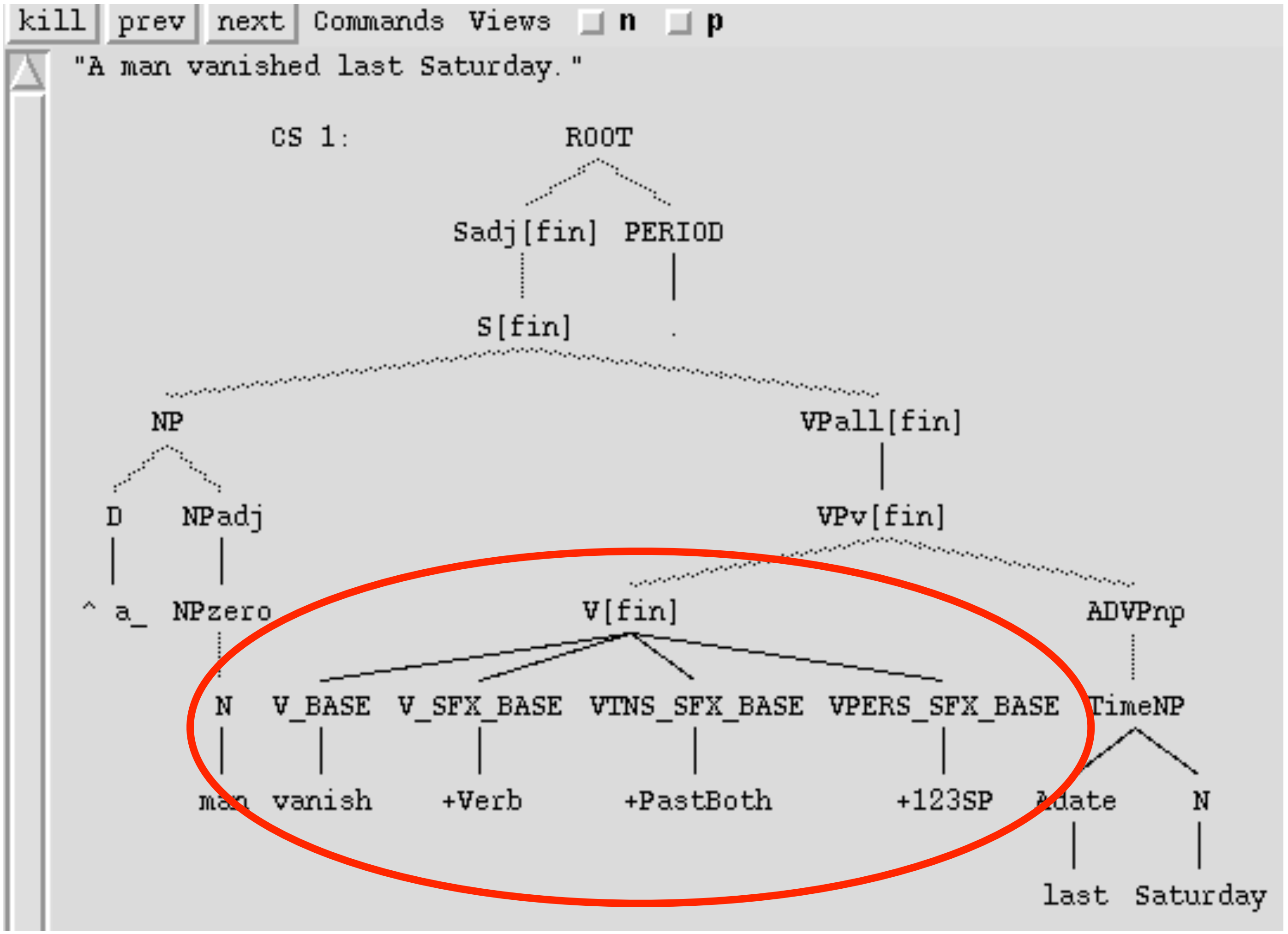
Underlying LFG Grammar



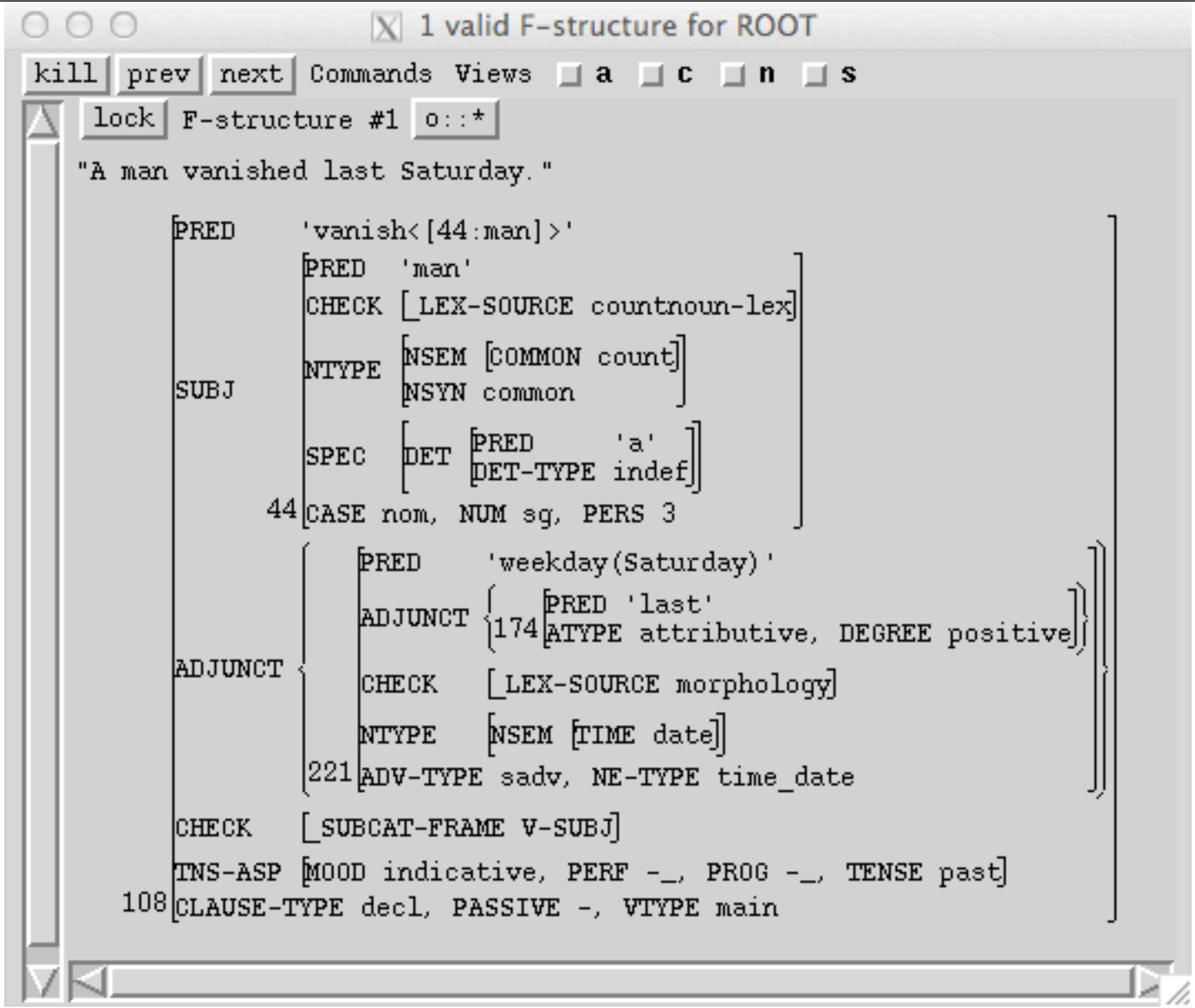
Each sentence is parsed by a large LFG grammar for English developed at PARC.

Each parse contains a constituent structure (c-str).

This shows linear order of the words, constituency and the hierarchical organization of the constituents.

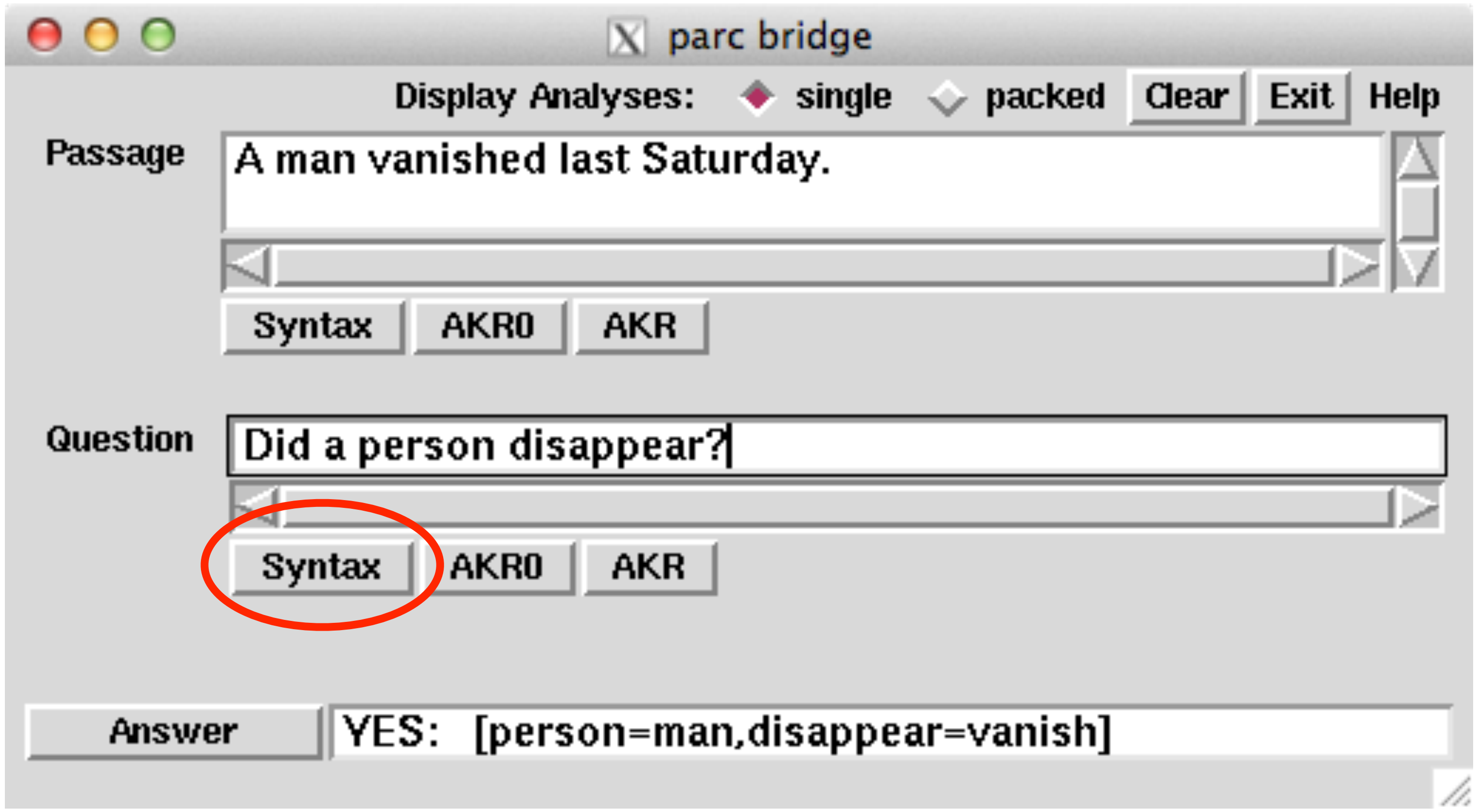


A finite-state morphological analyzer (FSM) feeds into the grammar.


1 valid F-structure for ROOT
kill prev next Commands Views a c n s
lock F-structure #1 o::*

```
"A man vanished last Saturday."

    PRED     'vanish<[44:man]>'
             PRED   'man'
             CHECK  [LEX-SOURCE countnoun-lex]
             NTYPE  NSEM [COMMON count]
    SUBJ            NSYN common
             SPEC   DET  PRED      'a'
                         DET-TYPE  indef
           44 CASE nom, NUM sg, PORS 3
                 PRED     'weekday(Saturday)'
                 ADJUNCT  {174 PRED 'last'
                               ATYPE attributive, DEGREE positive}
    ADJUNCT {    CHECK    [LEX-SOURCE morphology]
                 NTYPE    NSEM [TIME date]
             221 ADV-TYPE sadv, NE-TYPE time_date }
    CHECK    [SUBCAT-FRAME V-SUBJ]
    TNS-ASP  [MOOD indicative, PERF -_, PROG -_, TENSE past]
108 CLAUSE-TYPE decl, PASSIVE -, VTYPE main
```

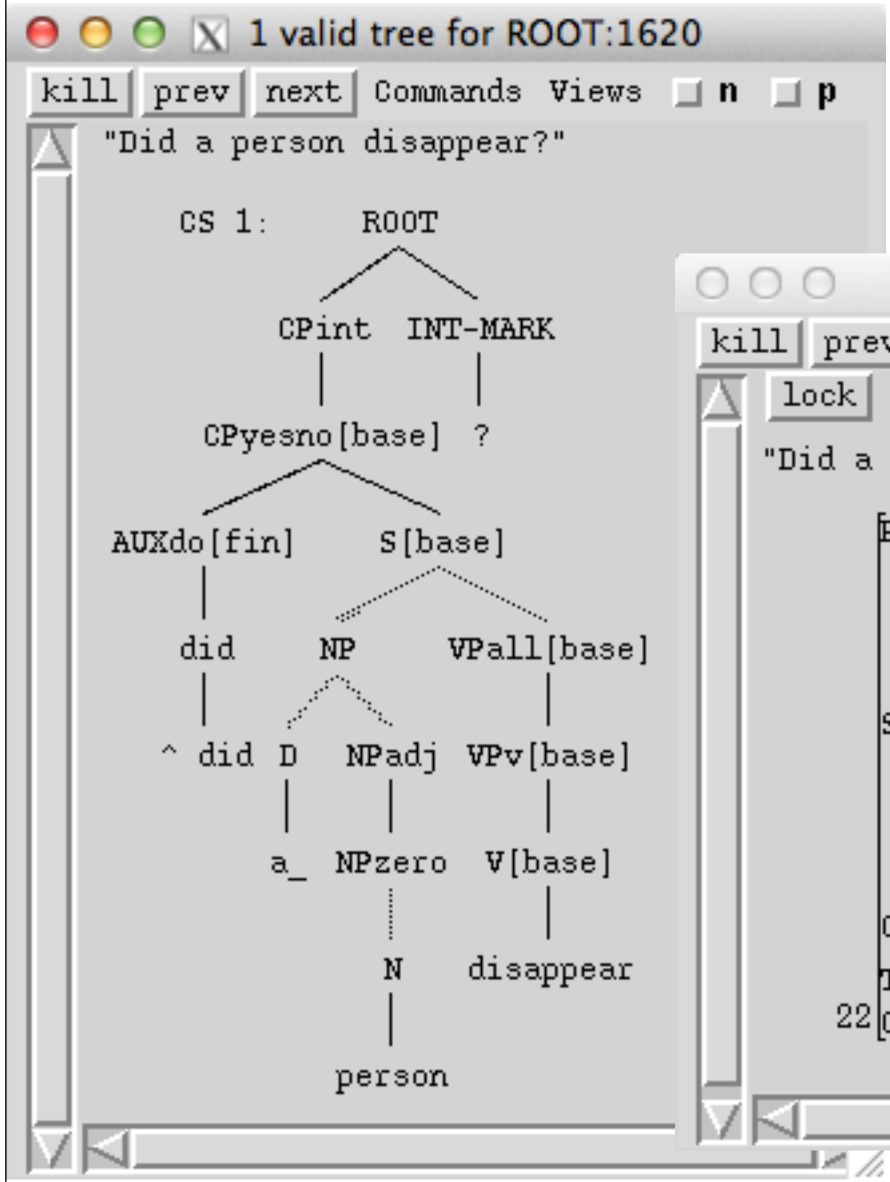
The functional structure (f-str) provides dependency and other deep linguistic information.

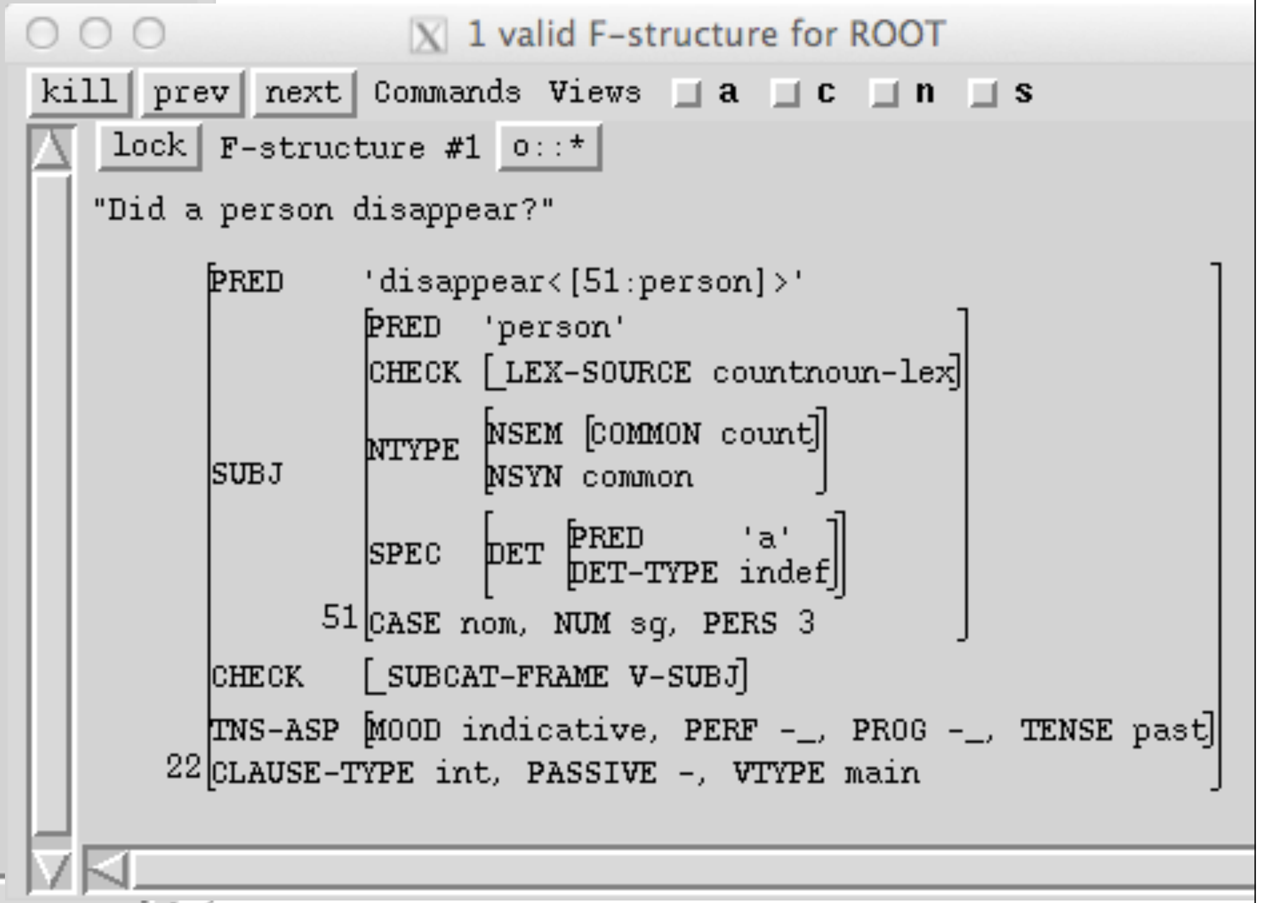


A man vanished last Saturday.

Did a person disappear?

YES: [person=man,disappear=vanish]

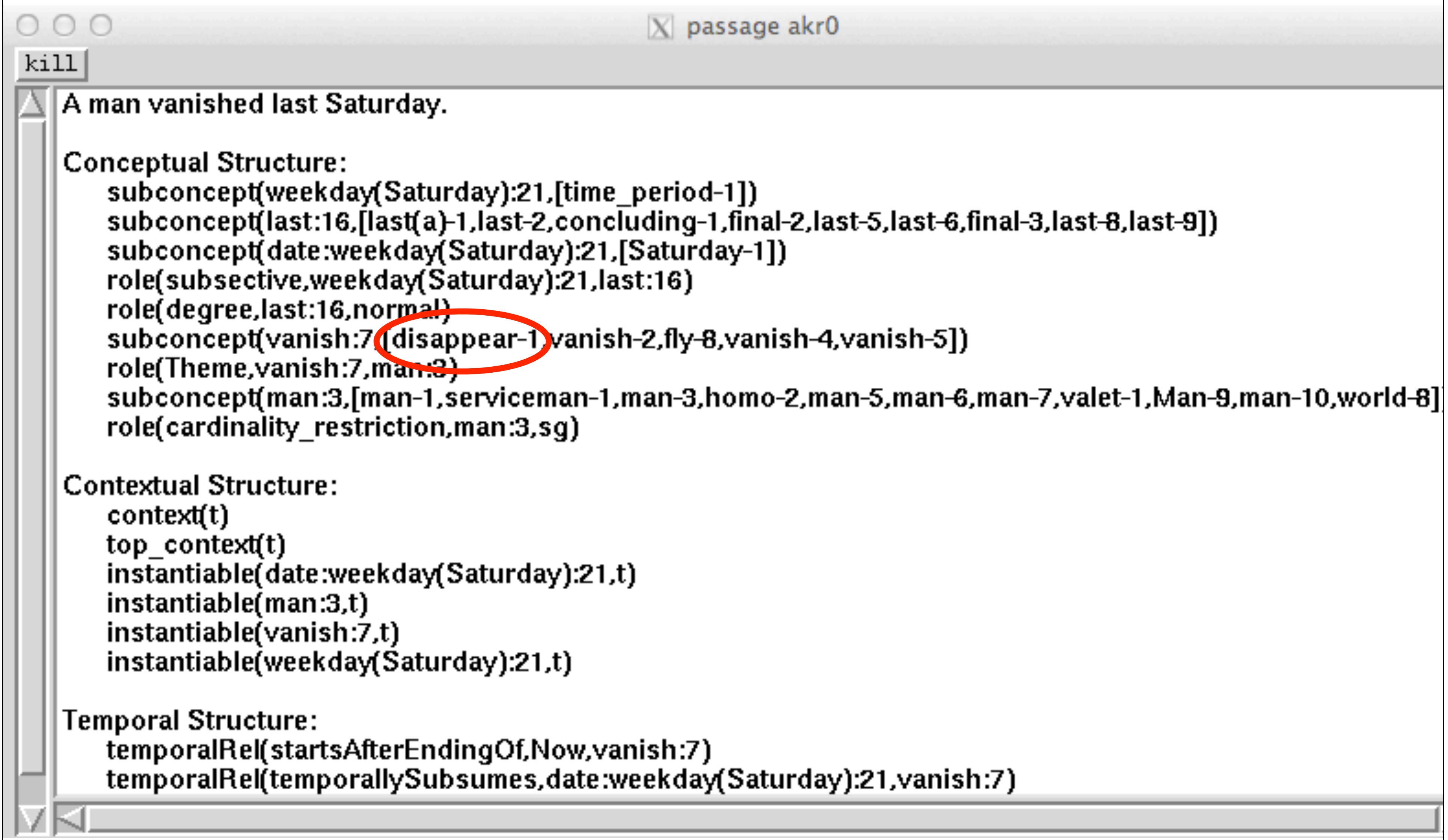




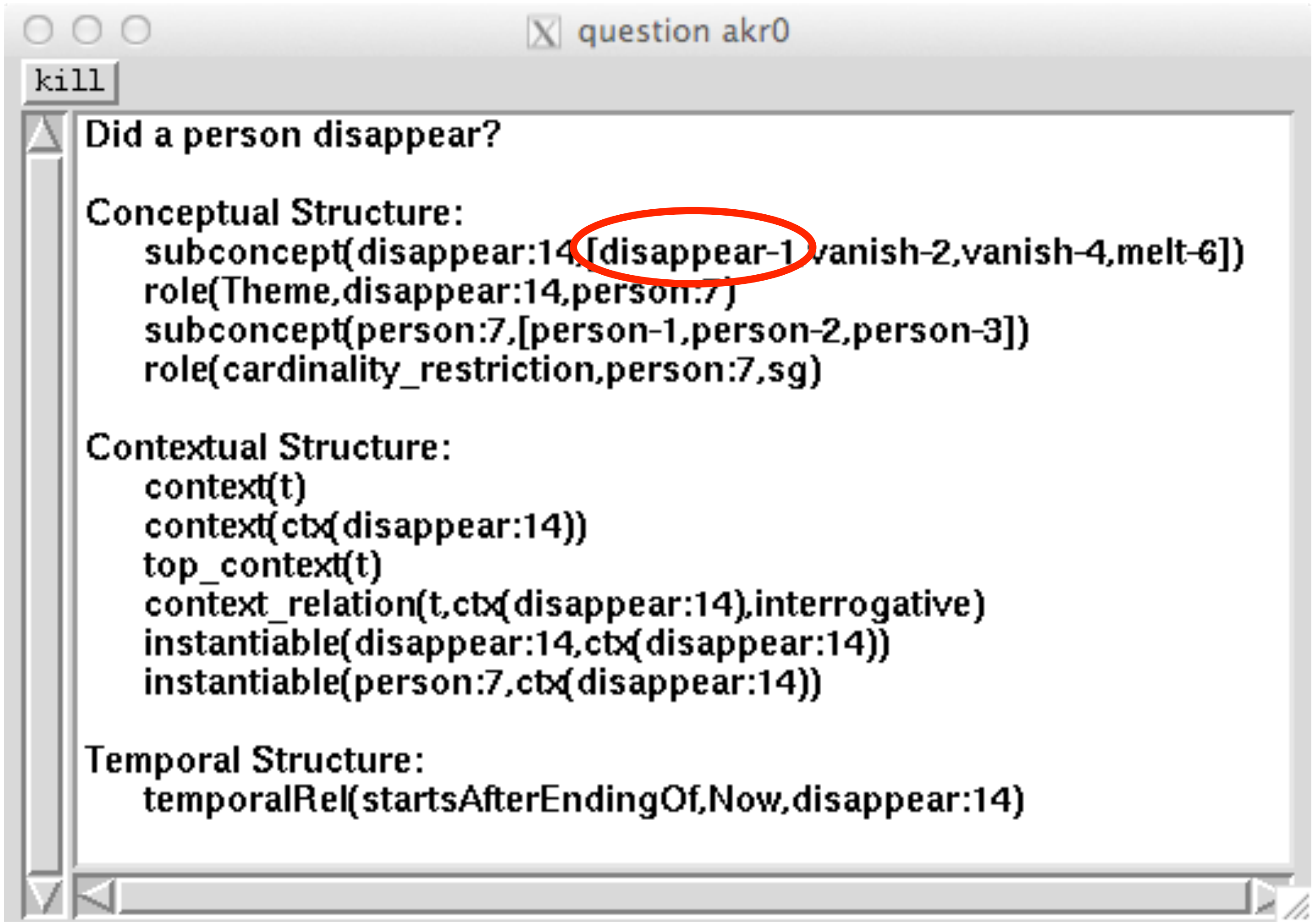
```
"Did a person disappear?"

   [PRED       'disappear<[51:person]>'                                       ]
   [           [PRED   'person'                         ]                     ]
   [           [CHECK  [LEX-SOURCE countnoun-lex]       ]                     ]
   [           [       [NSEM [COMMON count]]            ]                     ]
   [SUBJ       [NTYPE  [NSYN common        ]            ]                     ]
   [           [                                        ]                     ]
   [           [SPEC   [DET [PRED     'a'  ]]           ]                     ]
   [           [            [DET-TYPE indef]]           ]                     ]
   [         51[CASE nom, NUM sg, PERS 3                ]                     ]
   [CHECK      [SUBCAT-FRAME V-SUBJ]                                          ]
   [TNS-ASP    [MOOD indicative, PERF -_, PROG -_, TENSE past]                ]
 22[CLAUSE-TYPE int, PASSIVE -, VTYPE main                                    ]
```

Semantics (AKR)

- The f-structure provides the input for the semantic analysis, the Abstract Knowledge Representation (AKR).
- F-structures have been shown to be equivalent to Quasi-Logical Forms (QLFs) so they already contain a good approximation of basic semantics.
- The AKRs integrate lexical semantic knowledge from WordNet and other sources.
- The AKRs of the two sentences are compared to see if the answer to the question can be judged as correct

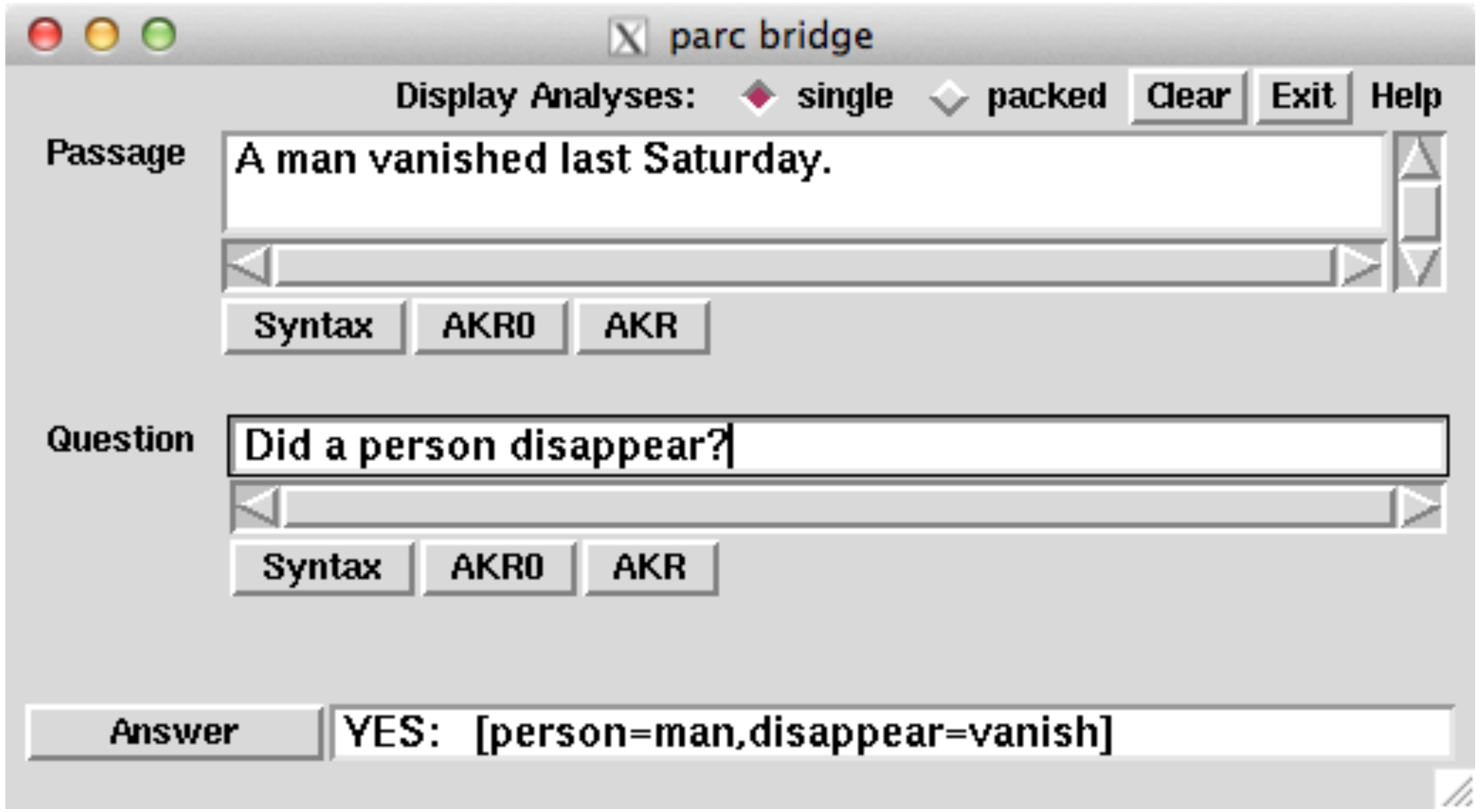


```
A man vanished last Saturday.

Conceptual Structure:
    subconcept(weekday(Saturday):21,[time_period-1])
    subconcept(last:16,[last(a)-1,last-2,concluding-1,final-2,last-5,last-6,final-3,last-8,last-9])
    subconcept(date:weekday(Saturday):21,[Saturday-1])
    role(subsective,weekday(Saturday):21,last:16)
    role(degree,last:16,normal)
    subconcept(vanish:7,[disappear-1,vanish-2,fly-8,vanish-4,vanish-5])
    role(Theme,vanish:7,man:3)
    subconcept(man:3,[man-1,serviceman-1,man-3,homo-2,man-5,man-6,man-7,valet-1,Man-9,man-10,world-8]
    role(cardinality_restriction,man:3,sg)

Contextual Structure:
    context(t)
    top_context(t)
    instantiable(date:weekday(Saturday):21,t)
    instantiable(man:3,t)
    instantiable(vanish:7,t)
    instantiable(weekday(Saturday):21,t)

Temporal Structure:
    temporalRel(startsAfterEndingOf,Now,vanish:7)
    temporalRel(temporallySubsumes,date:weekday(Saturday):21,vanish:7)
```



```
Did a person disappear?

Conceptual Structure:
    subconcept(disappear:14,[disappear-1,vanish-2,vanish-4,melt-6])
    role(Theme,disappear:14,person:7)
    subconcept(person:7,[person-1,person-2,person-3])
    role(cardinality_restriction,person:7,sg)

Contextual Structure:
    context(t)
    context(ctx(disappear:14))
    top_context(t)
    context_relation(t,ctx(disappear:14),interrogative)
    instantiable(disappear:14,ctx(disappear:14))
    instantiable(person:7,ctx(disappear:14))

Temporal Structure:
    temporalRel(startsAfterEndingOf,Now,disappear:14)
```

PARC's BRIDGE Q&A



parc bridge

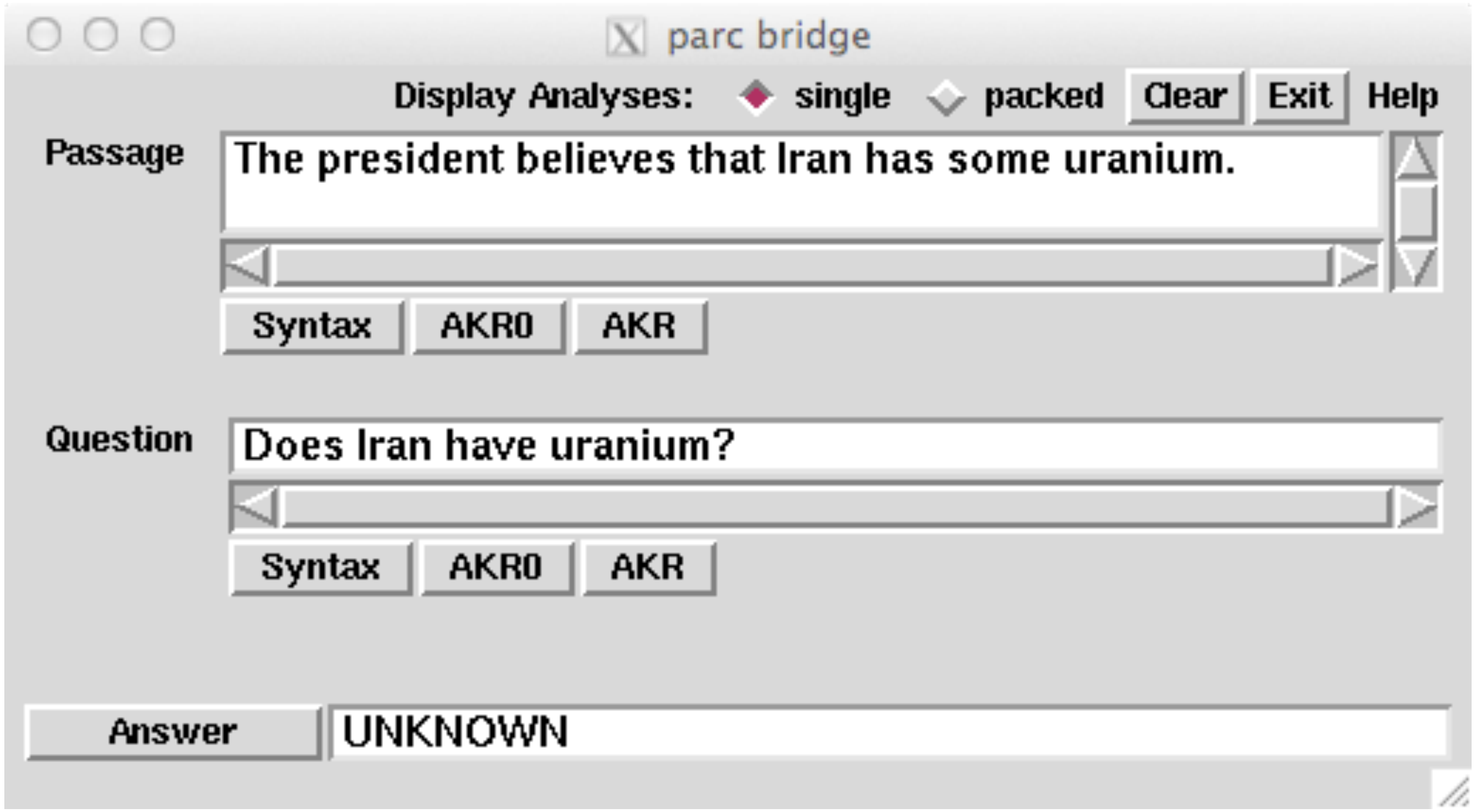
Display Analyses: [✓ single] packed

Passage: A man vanished last Saturday.

Question: Did a person disappear?

YES: [person=man,disappear=vanish]

Further Examples

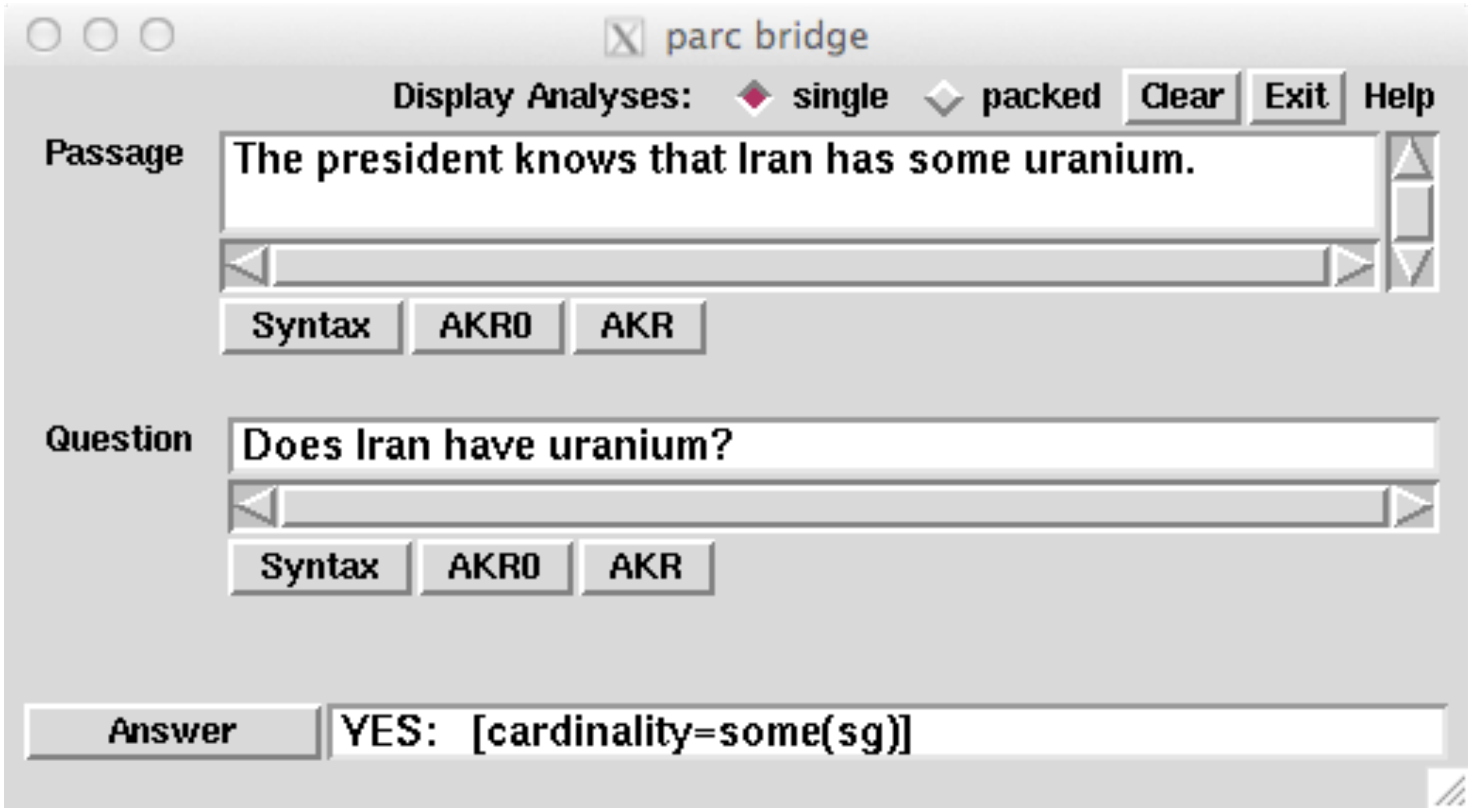


Passage: The president believes that Iran has some uranium.

Question: Does Iran have uranium?

Answer: UNKNOWN

Further Examples

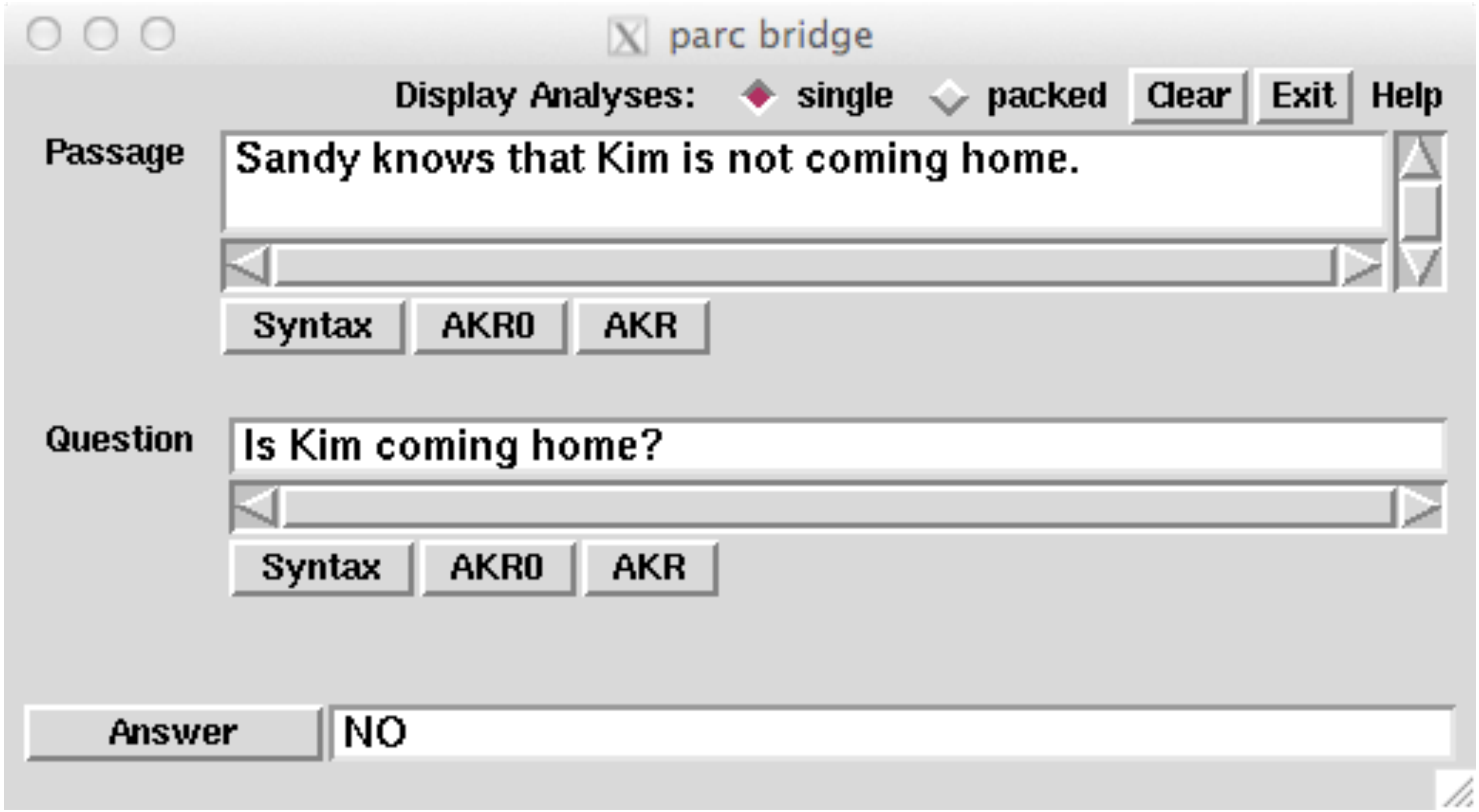


Passage: The president knows that Iran has some uranium.

Question: Does Iran have uranium?

Answer: YES: [cardinality=some(sg)]

Further Examples

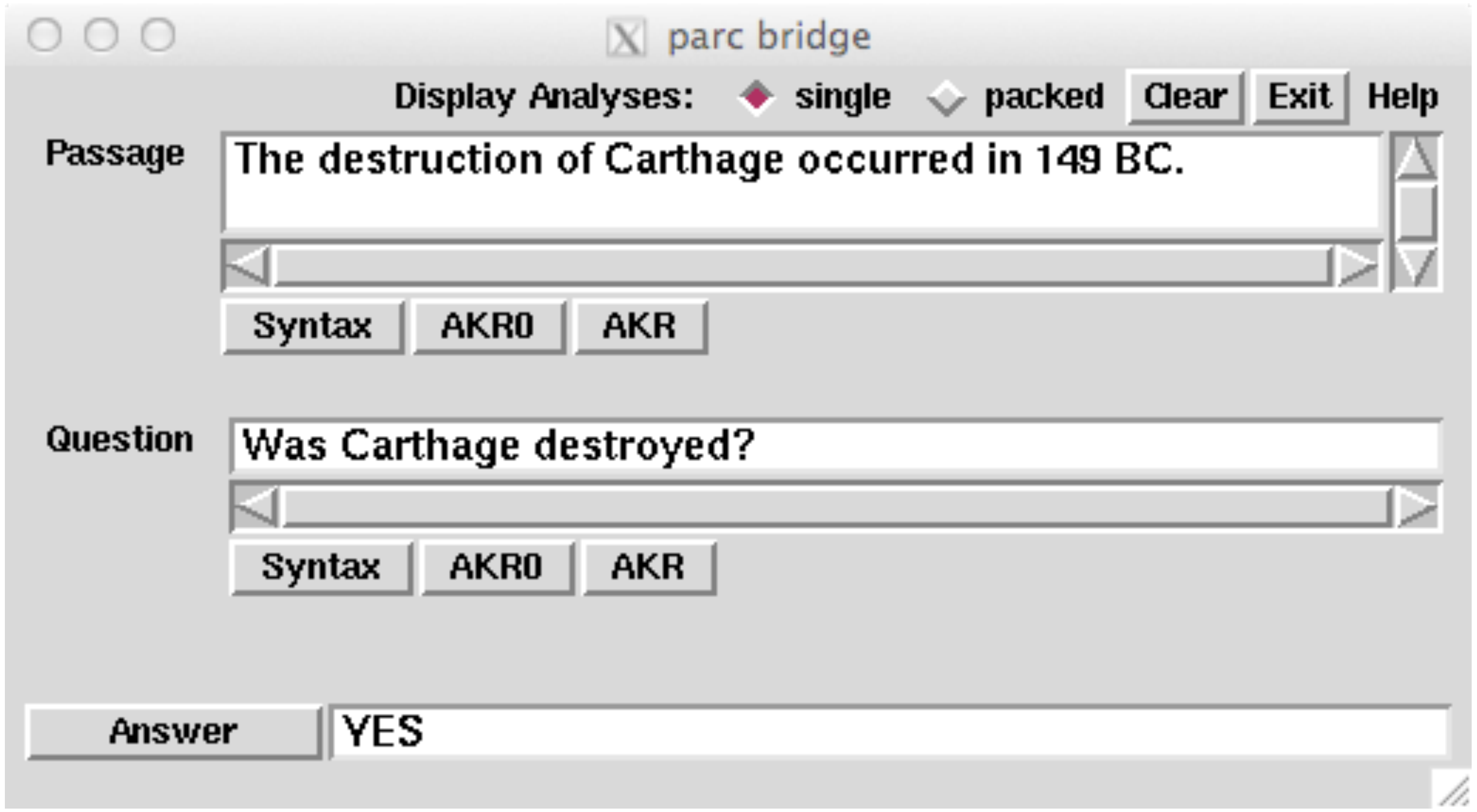


Passage: Sandy knows that Kim is not coming home.

Question: Is Kim coming home?

Answer: NO

Further Examples


parc bridge

Display Analyses: [✓ single] packed

Passage: The destruction of Carthage occurred in 149 BC.

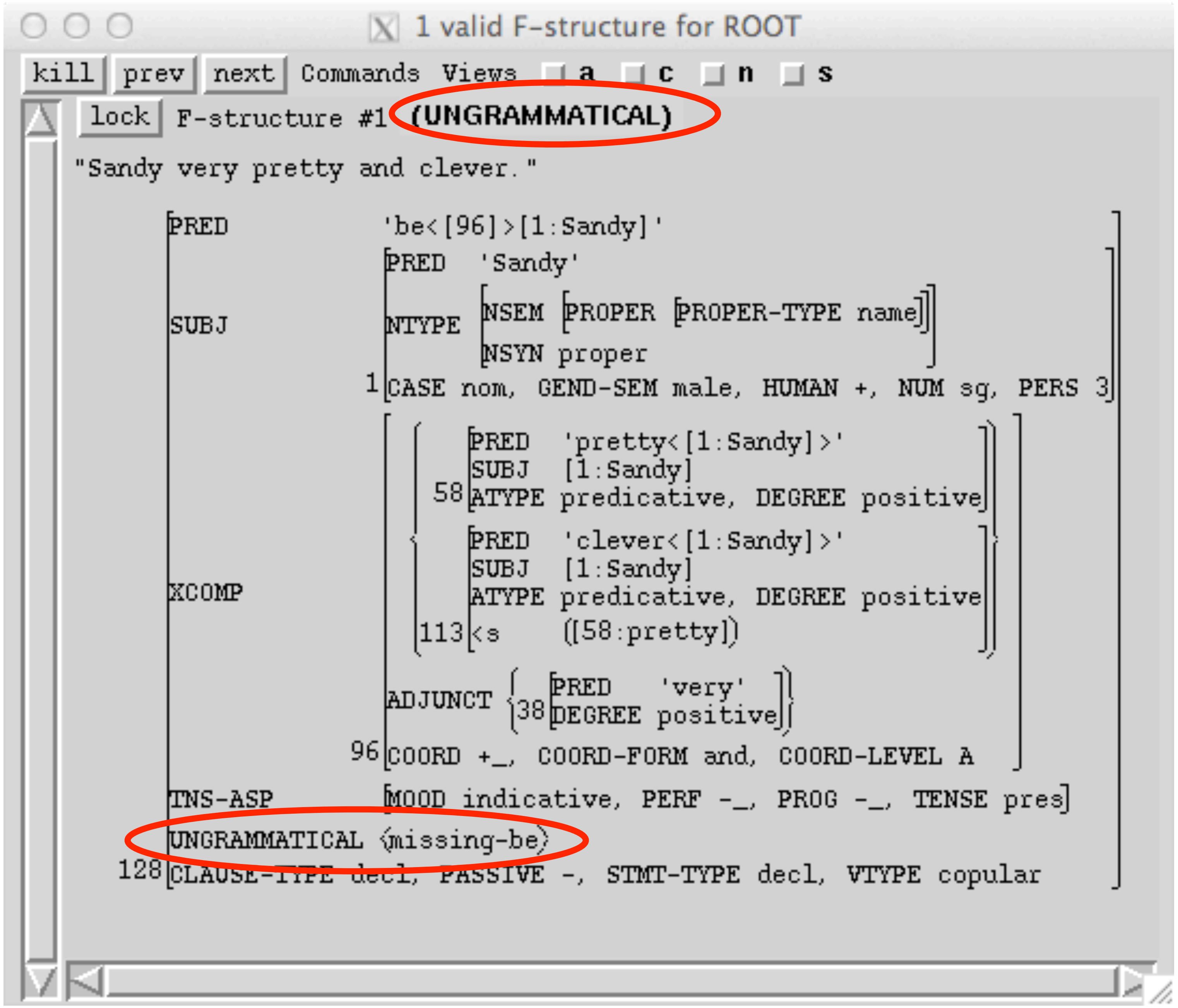
Question: Was Carthage destroyed?

Answer: YES

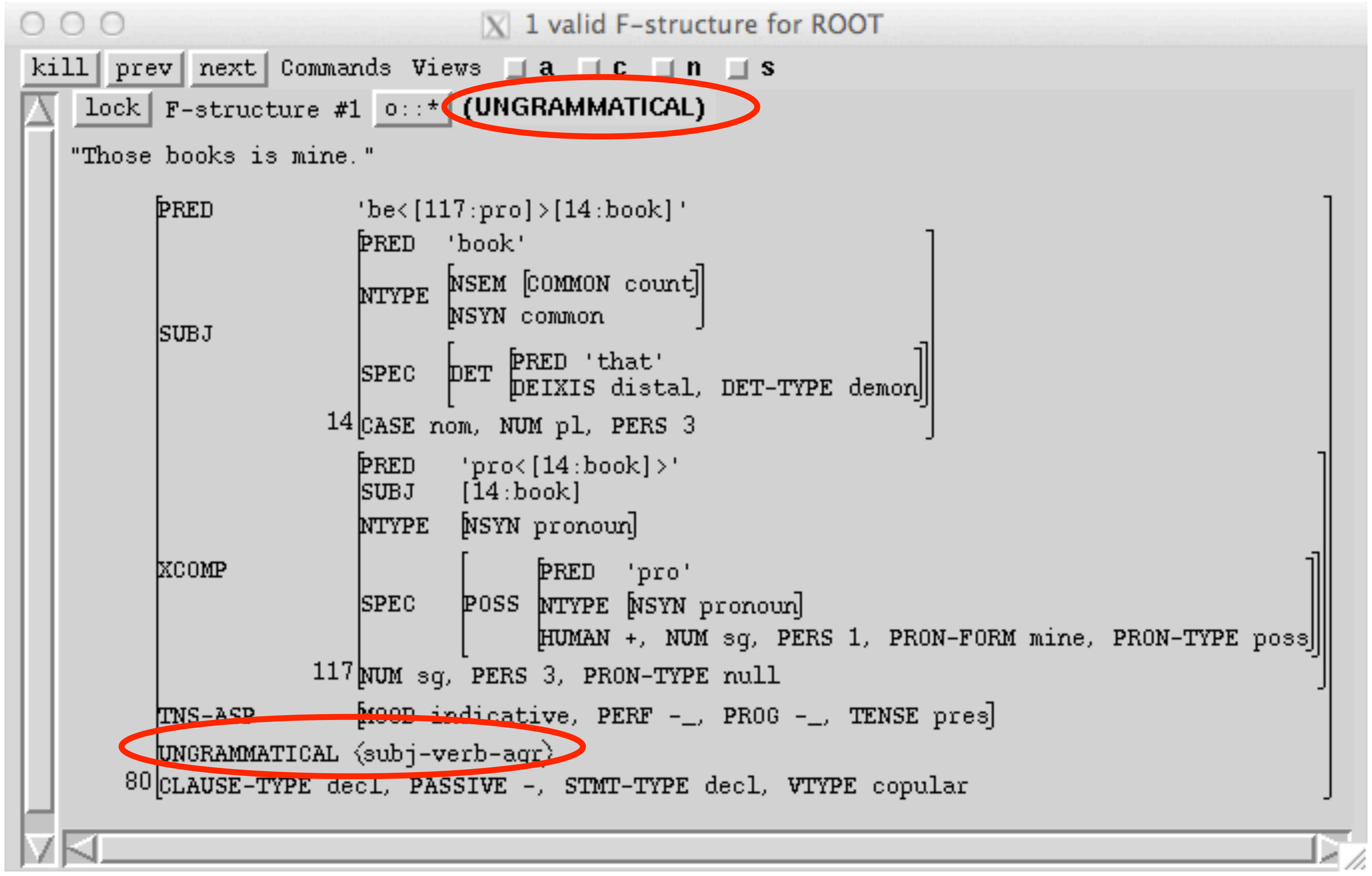
Application: CALL

- The following demo is based on the large English LFG grammar.
- The grammar was modified
  - to parse ungrammatical sentences
  - register and communicate the type of ungrammaticality
  - generate the grammatical version

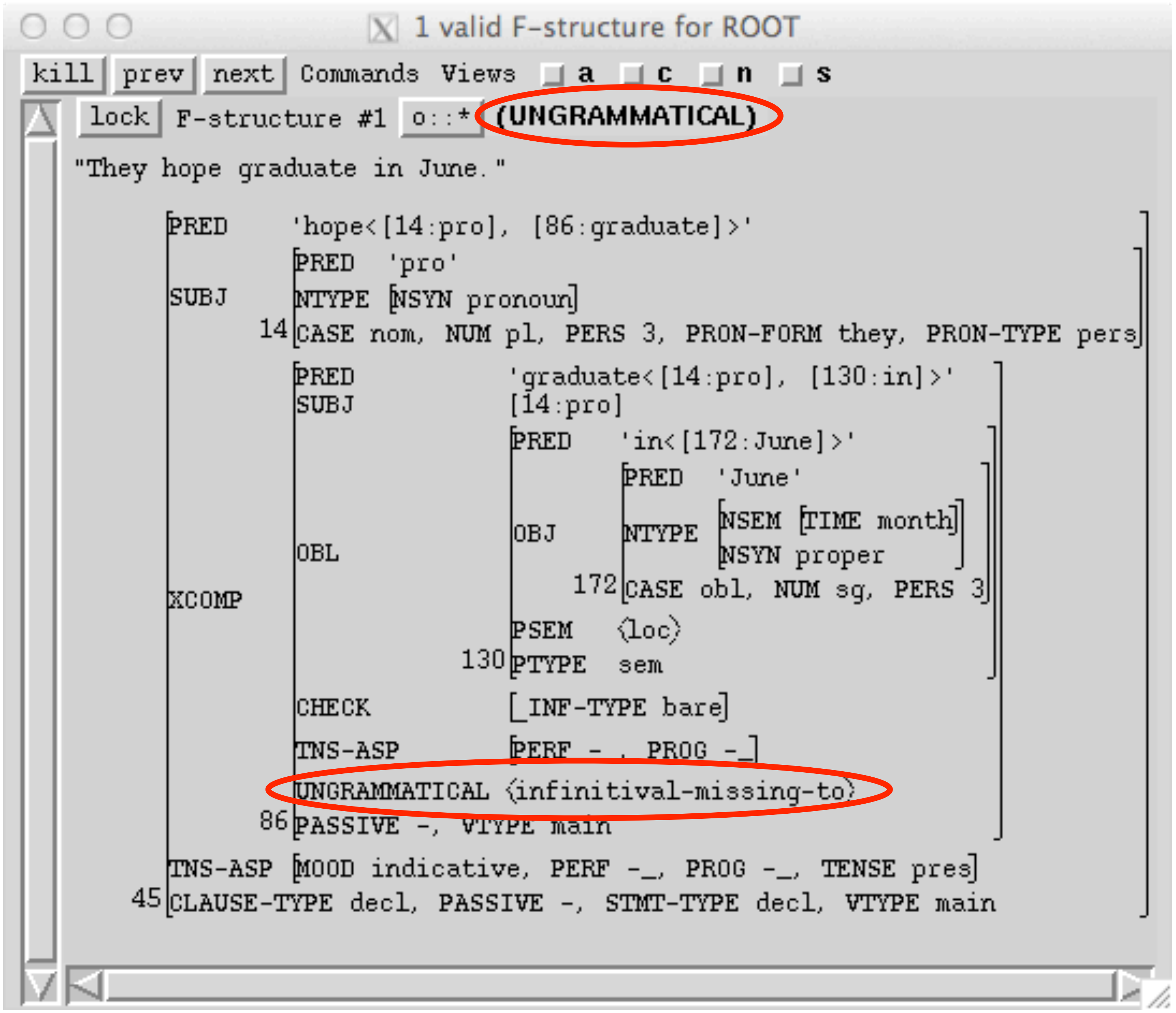
Khader, R. 2003. Evaluation of an English LFG-based Grammar as Error Checker. UMIST MSc Thesis, Manchester.



```
F-structure #1 (UNGRAMMATICAL)
"Sandy very pretty and clever."

    PRED          'be<[96]>[1:Sandy]'
                  PRED   'Sandy'
    SUBJ          NTYPE  NSEM [PROPER [PROPER-TYPE name]]
                         NSYN proper
                1 CASE nom, GEND-SEM male, HUMAN +, NUM sg, PERS 3
                      PRED   'pretty<[1:Sandy]>'
                      SUBJ   [1:Sandy]
                   58 ATYPE  predicative, DEGREE positive
                      PRED   'clever<[1:Sandy]>'
    XCOMP             SUBJ   [1:Sandy]
                      ATYPE  predicative, DEGREE positive
                  113 <s     ([58:pretty])
                  ADJUNCT  38 PRED   'very'
                              DEGREE positive
               96 COORD +_, COORD-FORM and, COORD-LEVEL A
    TNS-ASP       MOOD indicative, PERF -_, PROG -_, TENSE pres
    UNGRAMMATICAL {missing-be}
128 CLAUSE-TYPE decl, PASSIVE -, STMT-TYPE decl, VTYPE copular
```



```
(UNGRAMMATICAL)
"Those books is mine."

    PRED           'be<[117:pro]>[14:book]'
                   PRED   'book'
                   NTYPE  NSEM [COMMON count]
                          NSYN common
    SUBJ
                   SPEC   DET  PRED 'that'
                               DEIXIS distal, DET-TYPE demon
                14 CASE nom, NUM pl, PERS 3
                   PRED    'pro<[14:book]>'
                   SUBJ    [14:book]
                   NTYPE   [NSYN pronoun]
    XCOMP
                   SPEC    POSS  PRED  'pro'
                                 NTYPE [NSYN pronoun]
                                 HUMAN +, NUM sg, PERS 1, PRON-FORM mine, PRON-TYPE poss
               117 NUM sg, PERS 3, PRON-TYPE null
    TNS-ASP        MOOD indicative, PERF -_, PROG -_, TENSE pres
    UNGRAMMATICAL {subj-verb-agr}
 80 CLAUSE-TYPE decl, PASSIVE -, STMT-TYPE decl, VTYPE copular
```



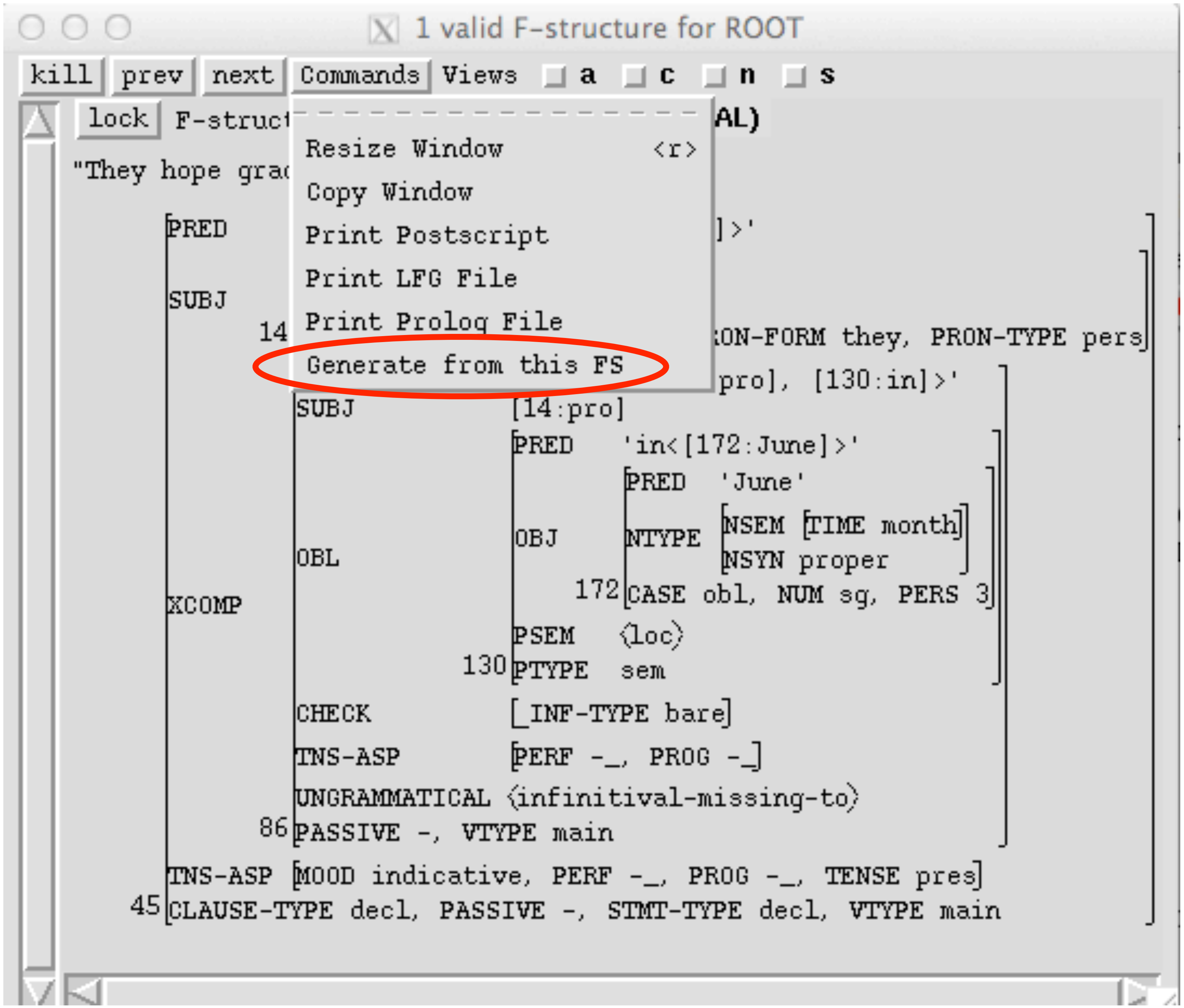
(UNGRAMMATICAL)

"They hope graduate in June."

```
   PRED     'hope<[14:pro], [86:graduate]>'
            PRED  'pro'
   SUBJ     NTYPE [NSYN pronoun]
         14 CASE nom, NUM pl, PERS 3, PRON-FORM they, PRON-TYPE pers
            PRED                'graduate<[14:pro], [130:in]>'
            SUBJ                [14:pro]
                                PRED   'in<[172:June]>'
                                       PRED   'June'
                                              NSEM [TIME month]
            OBL                 OBJ    NTYPE  NSYN proper
                                   172 CASE obl, NUM sg, PERS 3
                                PSEM   <loc>
                            130 PTYPE  sem
   XCOMP
            CHECK               [INF-TYPE bare]
            TNS-ASP             [PERF -_, PROG -_]
            UNGRAMMATICAL <infinitival-missing-to>
         86 PASSIVE -, VTYPE main
   TNS-ASP  [MOOD indicative, PERF -_, PROG -_, TENSE pres]
45 CLAUSE-TYPE decl, PASSIVE -, STMT-TYPE decl, VTYPE main
```

Generation: CALL

- So far we have seen only the parsing capability of the grammar.
- But all LFG/XLE grammars can use the grammar to generate.
- For a given f-structure, the grammar goes through all the available rules to produce the possible output specified by the grammar.
- The LFG/XLE grammars can work with underspecified input – this is what happens in the case with ungrammatical input.



```
AL)
"They hope grad

      PRED                                    ]>'
      SUBJ
          14                                  ON-FORM they, PRON-TYPE pers
                                              pro], [130:in]>'
              SUBJ          [14:pro]
                            PRED    'in<[172:June]>'
                                    PRED   'June'
                            OBJ     NTYPE  NSEM [TIME month]
              OBL                          NSYN proper
                                 172 CASE obl, NUM sg, PERS 3
      XCOMP                 PSEM    <loc>
                        130 PTYPE   sem
              CHECK         [INF-TYPE bare]
              TNS-ASP       [PERF -_, PROG -_]
              UNGRAMMATICAL <infinitival-missing-to>
          86  PASSIVE -, VTYPE main
      TNS-ASP [MOOD indicative, PERF -_, PROG -_, TENSE pres]
   45 CLAUSE-TYPE decl, PASSIVE -, STMT-TYPE decl, VTYPE main
```

XLE in the Shell

This is what XLE looks like when called up from a command line:

```
% xle

XLE loaded from /usr/local/xle/bin/xle.
XLEPATH = /usr/local/xle.
Copyright (c) 1993-2001 by the Xerox Corporation and
Copyright (c) 2002-2009 by the Palo Alto Research Center.
All rights reserved. This software is made available AS IS,
and PARC and the Xerox Corporation make no warranty about
the software, its performance or its conformity to any specification.
XLE version 2.6.4 (built Aug 10, 2009 09:39 -0700)
Type 'help' for more information.
loading /Users/mutt/pargram/call/english-call.lfg...
Grammar has 382 rules with 8736 states, 22764 arcs, and 29349 disjuncts (36235
DNF).
```

XLE in the Shell

This is what it looks like when the grammar generates from an f-structure.

```
% regenerate "They hope graduate in June."
parsing {They hope graduate in June.}
*2+40 solutions, 0.050 CPU seconds, 0.000MB max mem, 535
subtrees unified

They hope to graduate in June.

regeneration took 0.12 CPU seconds.
%
```

Paradigms via Underspecified Generation

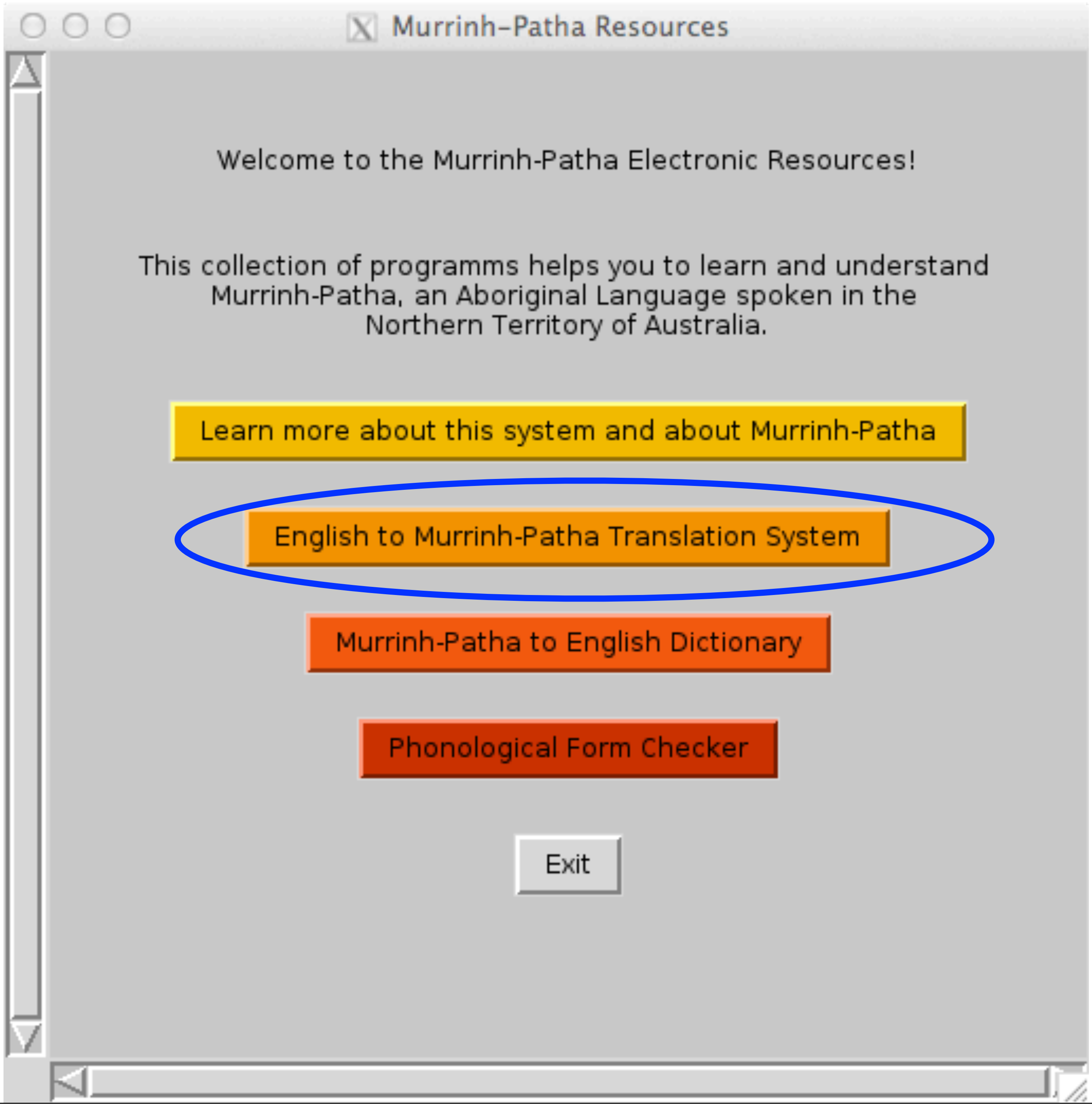
- Next is an example in which the grammar has been asked to generate without any tense/aspect information specified as part of the input.
- The result is the generation of the entire verbal paradigm of the language.

```
% regenerate "John sleeps."
parsing {John sleeps.}
1+5 solutions, 0.020 CPU seconds, 0.000MB max mem, 67 subtrees unified

John
  {   { will {have been|be}
      |was
      |{has|had} been
      |is}
    sleeping.
   |will {have slept.|sleep.}
   |{has|had} slept.
   |slept.
   |sleeps.}

regeneration took 0.09 CPU seconds.
%
```

More CALL Potential

- The following demo is based on work by Melanie Seiss on the Australian Aboriginal language Murrinh-Patha.
- The system uses:
  - the large English LFG grammar
  - an LFG grammar and FSM for Murrinh-Patha
  - the XLE/XFR transfer system for machine translation
  - Perl scripts and a TCL/TK Interface

Seiss, Melanie & Rachel Nordlinger. 2011. An Electronic Dictionary and Translation System for Murrinh-Patha. *Proceedings of the EUROCALL 2011 Conference*, University of Nottingham.

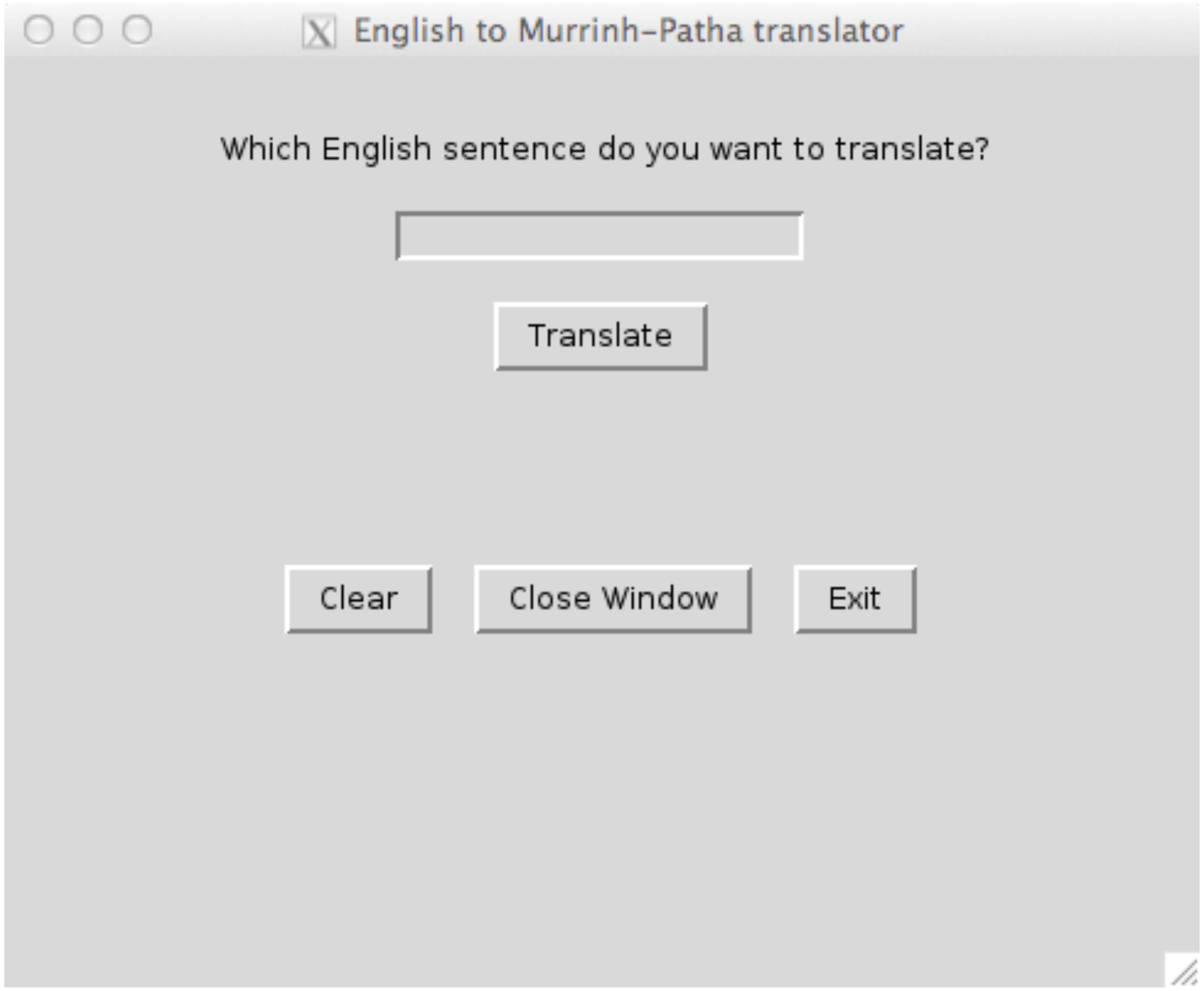


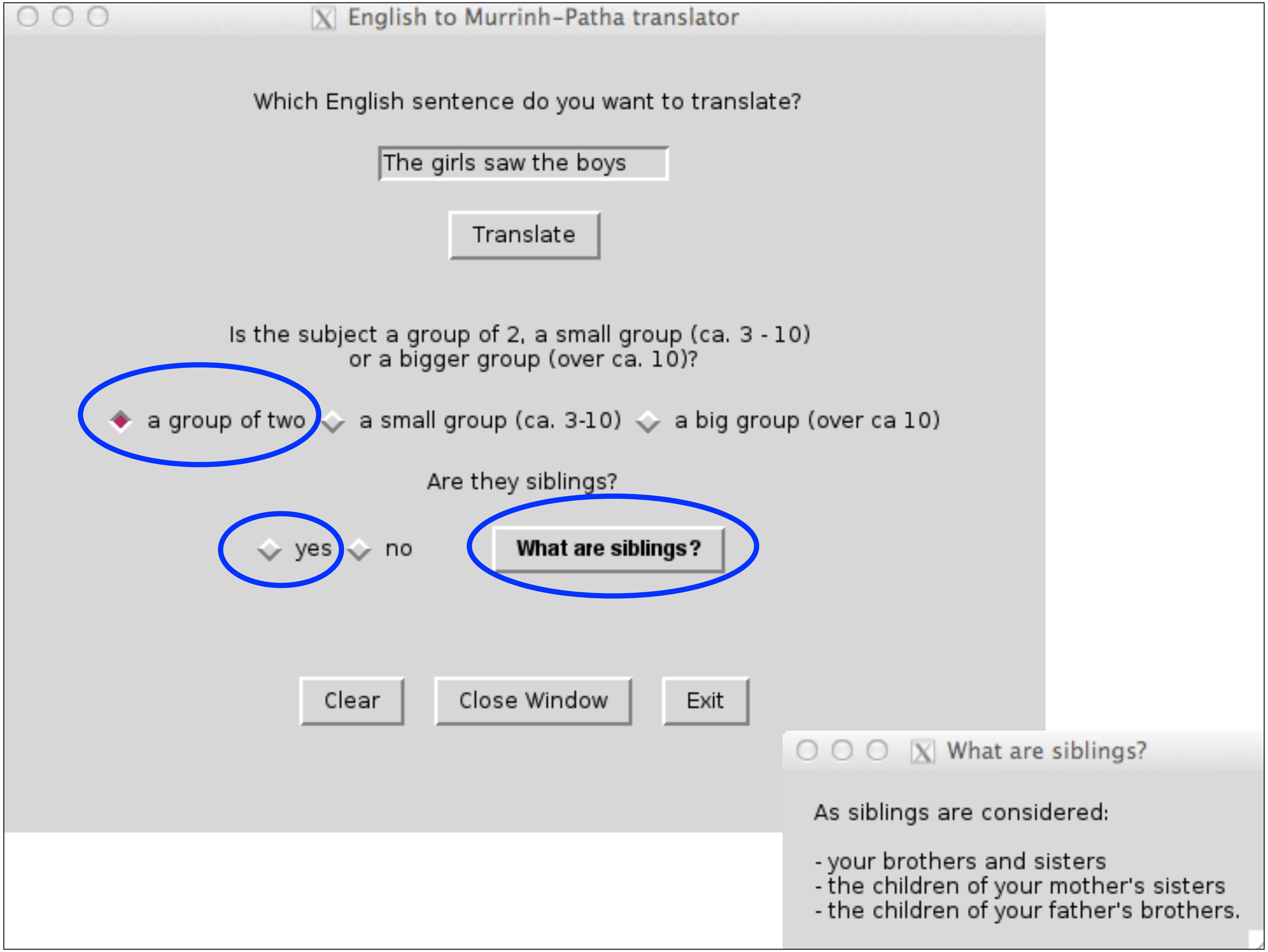
Murrinh–Patha Resources

Welcome to the Murrinh-Patha Electronic Resources!

This collection of programms helps you to learn and understand Murrinh-Patha, an Aboriginal Language spoken in the Northern Territory of Australia.

Learn more about this system and about Murrinh-Patha
English to Murrinh-Patha Translation System
Murrinh-Patha to English Dictionary
Phonological Form Checker
Exit


English to Murrinh-Patha translator
Which English sentence do you want to translate?
Translate
Clear
Close Window
Exit


English to Murrinh-Patha translator

Which English sentence do you want to translate?

The girls saw the boys

Translate

Is the subject a group of 2, a small group (ca. 3 - 10)
or a bigger group (over ca. 10)?

[✓ a group of two] a small group (ca. 3-10) a big group (over ca 10)

Are they siblings?

yes no

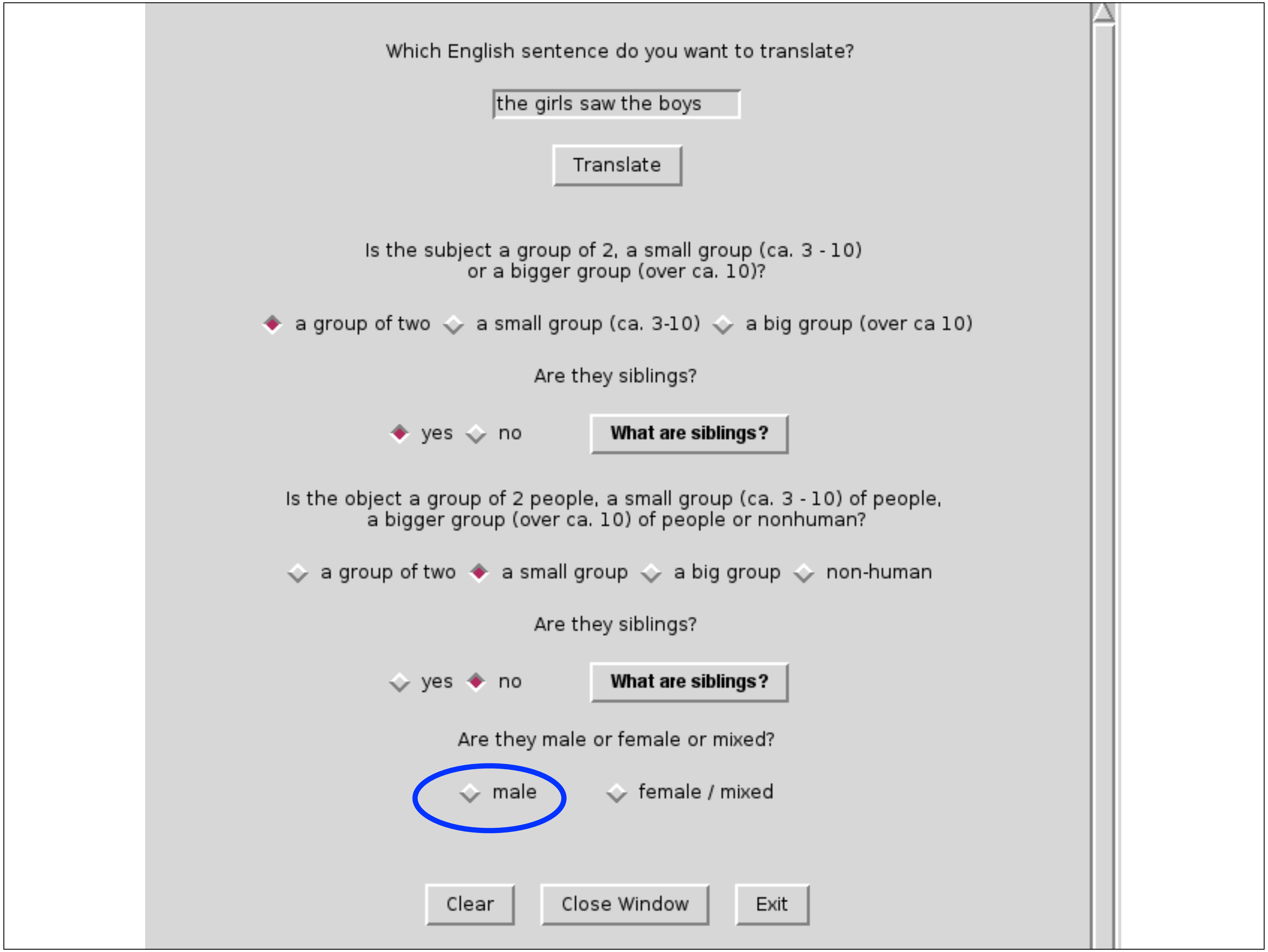
What are siblings?

Clear Close Window Exit

What are siblings?

As siblings are considered:

- your brothers and sisters
- the children of your mother's sisters
- the children of your father's brothers.



Which English sentence do you want to translate?

the girls saw the boys

Translate

Is the subject a group of 2, a small group (ca. 3 - 10)
or a bigger group (over ca. 10)?

[✓ a group of two] a small group (ca. 3-10) a big group (over ca 10)

Are they siblings?

[✓ yes] no

What are siblings?

Is the object a group of 2 people, a small group (ca. 3 - 10) of people,
a bigger group (over ca. 10) of people or nonhuman?

a group of two [✓ a small group] a big group non-human

Are they siblings?

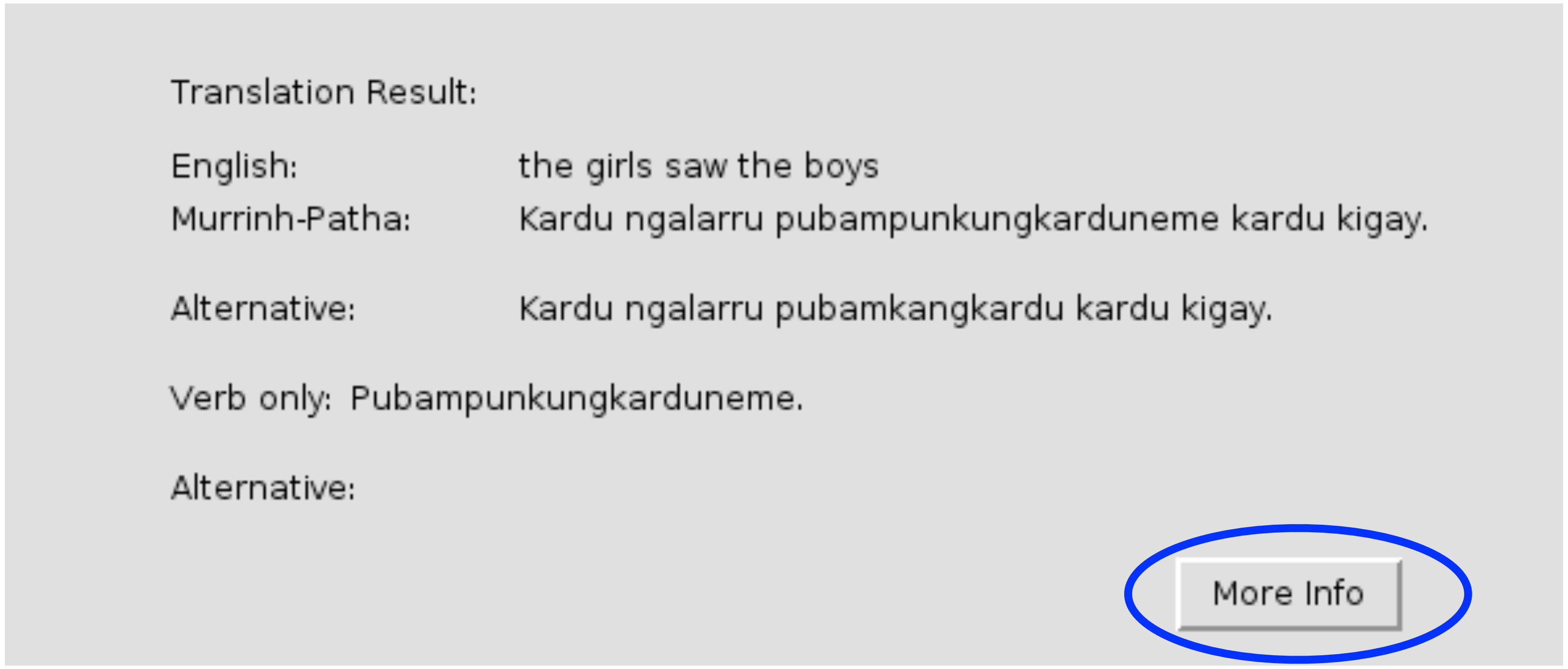
yes [✓ no]

What are siblings?

Are they male or female or mixed?

[✓ male] female / mixed

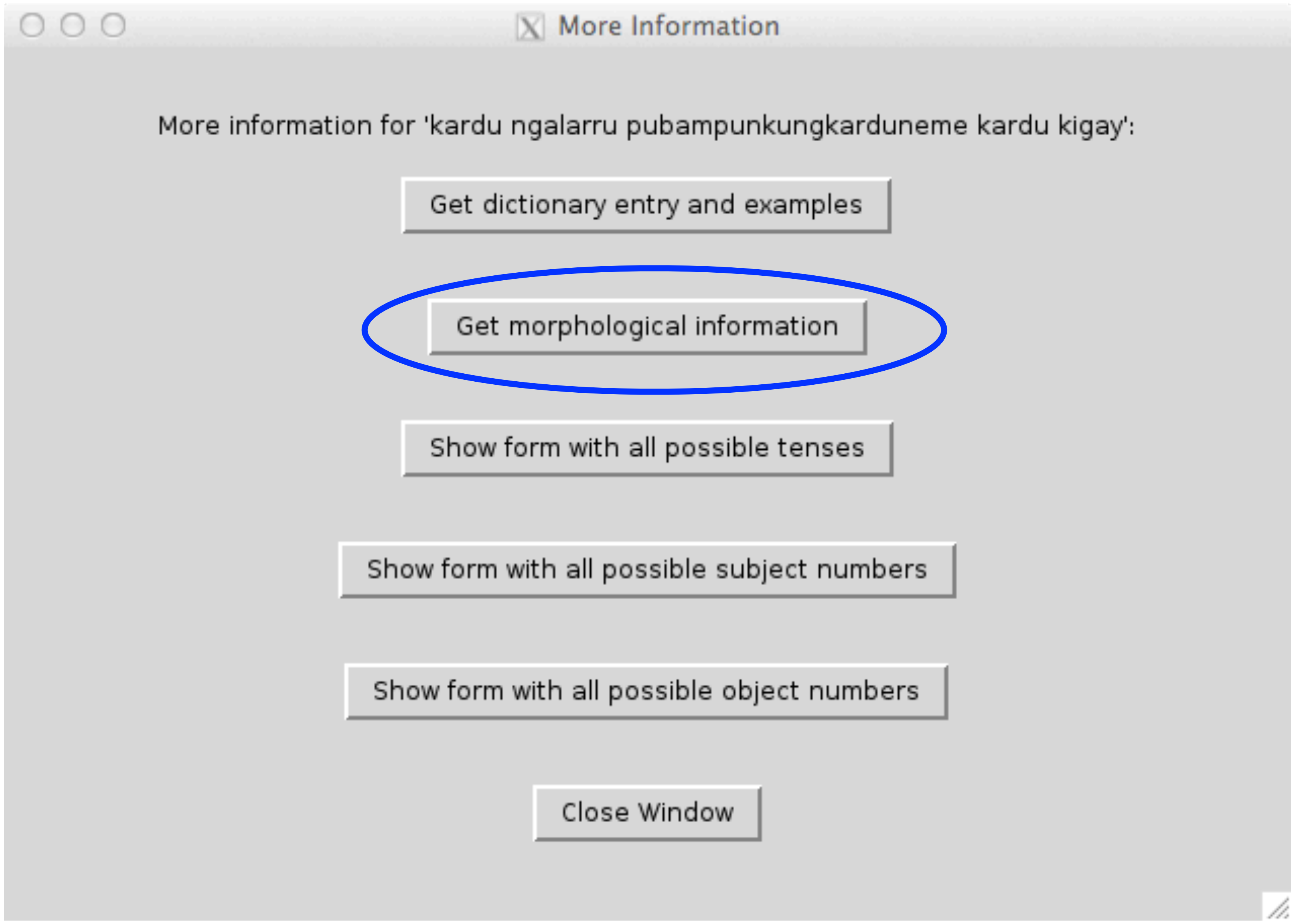
Clear Close Window Exit


Translation Result:

English:	the girls saw the boys
Murrinh-Patha:	Kardu ngalarru pubampunkungkarduneme kardu kigay.
Alternative:	Kardu ngalarru pubamkangkardu kardu kigay.

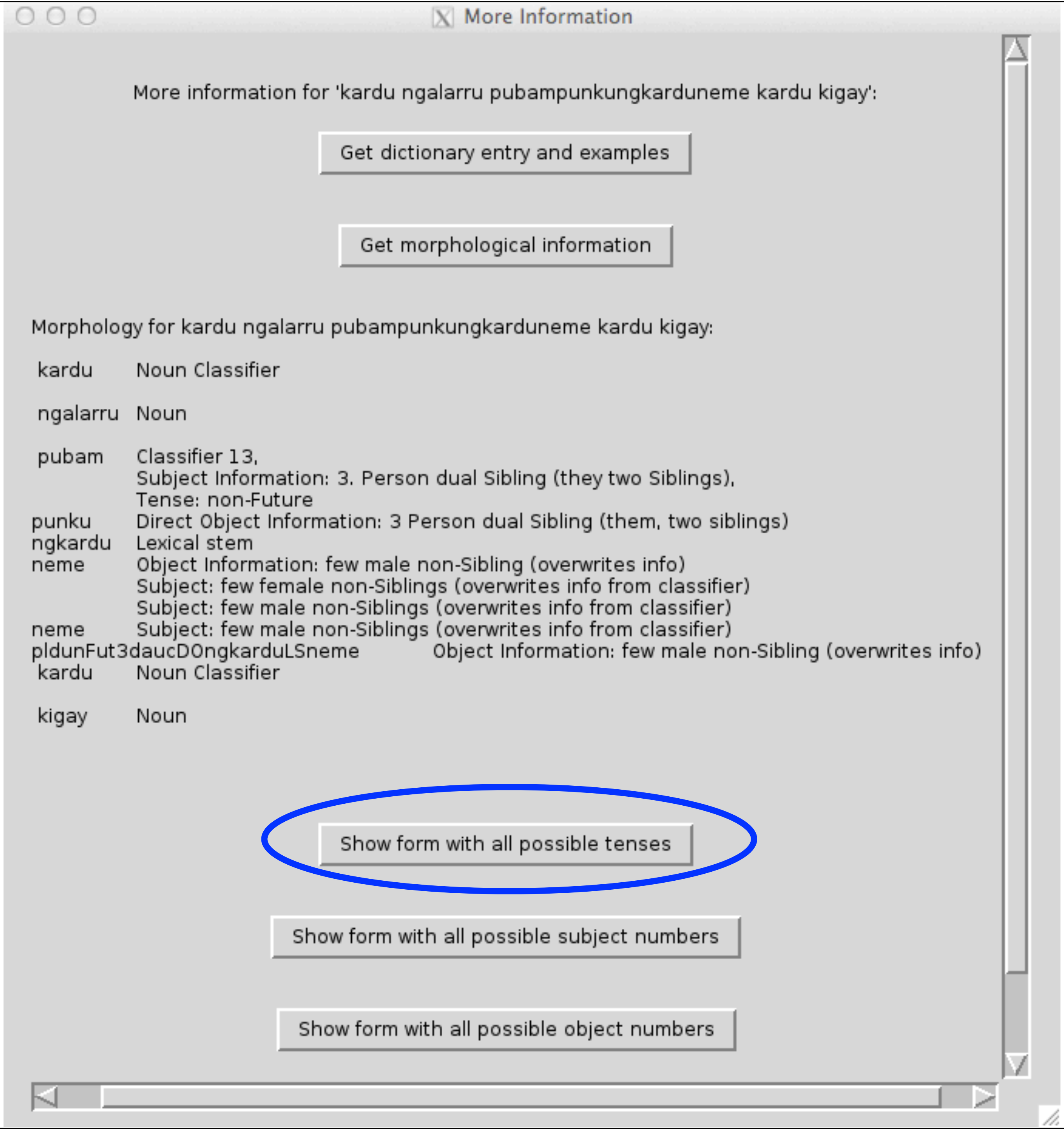
Verb only: Pubampunkungkarduneme.

Alternative:



More information for 'kardu ngalarru pubampunkungkarduneme kardu kigay':

Get dictionary entry and examples
Get morphological information
Show form with all possible tenses
Show form with all possible subject numbers
Show form with all possible object numbers
Close Window

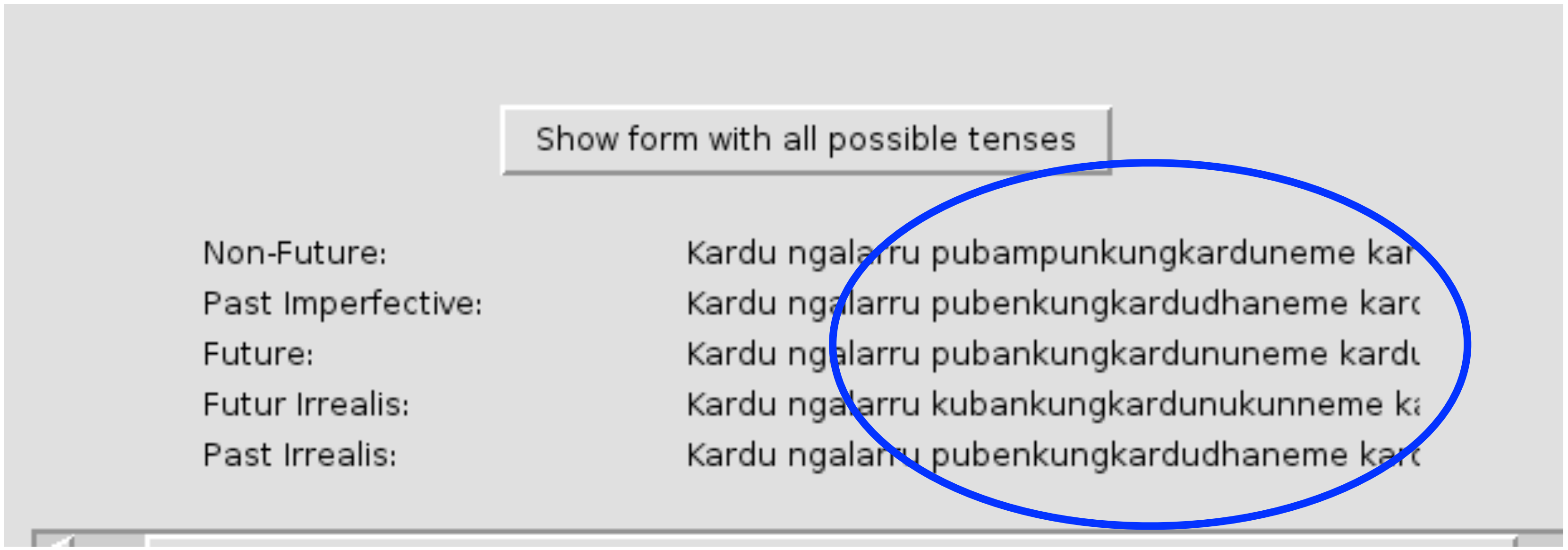

More Information

More information for 'kardu ngalarru pubampunkungkarduneme kardu kigay':

Get dictionary entry and examples
Get morphological information

```
Morphology for kardu ngalarru pubampunkungkarduneme kardu kigay:

 kardu     Noun Classifier

 ngalarru  Noun

 pubam     Classifier 13,
           Subject Information: 3. Person dual Sibling (they two Siblings),
           Tense: non-Future
punku      Direct Object Information: 3 Person dual Sibling (them, two siblings)
ngkardu    Lexical stem
neme       Object Information: few male non-Sibling (overwrites info)
           Subject: few female non-Siblings (overwrites info from classifier)
           Subject: few male non-Siblings (overwrites info from classifier)
neme       Subject: few male non-Siblings (overwrites info from classifier)
pldunFut3daucDOngkarduLSneme         Object Information: few male non-Sibling (overwrites info)
 kardu     Noun Classifier

 kigay     Noun
```

Show form with all possible tenses
Show form with all possible subject numbers
Show form with all possible object numbers



Non-Future:	Kardu ngalarru pubampunkungkarduneme kar
Past Imperfective:	Kardu ngalarru pubenkungkardudhaneme karc
Future:	Kardu ngalarru pubankungkardununeme kardu
Futur Irrealis:	Kardu ngalarru kubankungkardunukunneme ka
Past Irrealis:	Kardu ngalarru pubenkungkardudhaneme kar

Deep Grammars

- Deep Grammars have great potential with respect to NLP applications.
- Deep Grammars serve as a systematic tool for language documentation
- Why don’t more people use them?

Disadvantages

- Time consuming and expensive to write
  - shallow parsers can be induced automatically from a training set (but somebody has to construct that first)
- Brittle
  - shallow parsers produce something for everything
- Ambiguous
  - shallow parsers rank the outputs
- Slow
  - shallow parsers are very fast (real time)

Why pay attention now?

New Generation of Large-Scale Grammars:

- Robustness:
  - Integrated Chunk Parsers/Fragment Grammars
  - Bad input always results in some (possibly good) output
- Ambiguity:
  - Integration of stochastic methods
  - Optimality Theory used to rank/pick alternatives
- Speed: getting closer to shallow parsers
- Accuracy and information content:
  - far beyond the capabilities of shallow parsers.

XLE at PARC

- Platform for Developing Large-Scale LFG Grammars

- LFG (Lexical-Functional Grammar)
  - Invented in the 1980s
    (Joan Bresnan and Ronald Kaplan)
  - Theoretically stable ⇔ Solid Implementation

- XLE is implemented in C, used with emacs, tcl/tk
- XLE includes a **parser**, **generator** and **transfer** (XFR) component.

XLE at PARC

- Platform for Developing Large-Scale LFG Grammars

- LFG (Lexical-Functional Grammar)
  - Invented in the 1980s
    (Joan Bresnan and Ronald Kaplan)
  - Theoretically stable ⇔ Solid Implementation

- XLE is implemented in C, used with emacs, tcl/tk
- XLE includes a **parser**, **generator** and **transfer** (XFR) component.
- Extensive on-line documentation

Background Literature

- Bresnan, Joan & Ron Kaplan. 1982. *The Mental Representation of Grammatical Relations*. MIT Press.
- Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Blackwell.

- Dalrymple, Mary. 2001. *Lexical-Functional Grammar.* Academic Press.
- Butt, Miriam, Tracy Holloway King, Frédérique Segond & María-Eugenia Niño. 1999. *The Grammar Writer's Cookbook.* CSLI Publications.

XLE/LFG and ParGram

- XLE has been used to develop a number of grammars for very different languages as part of the **ParGram** (Parallel Grammar) collaborative effort.
- (see websites, ParGram Wiki for detailed information)

The Parallel in ParGram

- Analyze languages to a degree of abstraction that reflects the common underlying structure (i.e., identify the subject, the object, the tense, mood, etc.).

- Even at this level, there is usually more than one way to analyze a construction.
  - The same theoretical analysis may have different possible implementations.
  - ParGram agrees on common analyses and implementations (via regular meetings and a feature committee).

The Parallel in ParGram

- Analyses at the level of c-structure are allowed to differ (variance across languages)
- Analyses at f-structure are held as parallel as possible across languages (crosslinguistic invariance).
- **Theoretical Advantage**: This models the idea of universal principles underlying language structure.
- **Applicational Advantage**: machine translation is made easier; applications are more easily adapted to new languages.

ParGram Languages (so far)

- Arabic, Chinese, Danish, English, French, Georgian, German, Hungarian, Irish Gaelic, Indonesian, Japanese, Malagasy, Murrihn-Patha, Norwegian, Polish, Setswana, Tigrinya, Turkish, Urdu, Welsh, Wolof
- Loose organization: no common deliverables, but common interests.
- Recent Multilingual Effort: the ParGramBank
- Many grammars are available via the INESS website.

Sebastian Sulger, Miriam Butt, Tracy Holloway King, Paul Meurer, Tibor Laczkó, György Rákosi, Cheikh Bamba Dione, Helge Dyvik, Victoria Rosén, Koenraad De Smedt, Agnieszka Patejuk, Özlem Çetinoğlu, I Wayan Arka and Meladel Mistica: ParGramBank: The ParGram Parallel Treebank. *Proceedings of ACL 2013* (Long Papers), Sofia, Bulgaria. Association for Computational Linguistics.

Getting Started with XLE

- This concludes today's lesson.
- As the practical part of this lesson, you should now:
    1. Install XLE so that you can use it for the exercises in the course.
    2. Go to the INESS website where many of the ParGram grammars are available and experiment with the grammars and the representations.
    3. The English LFG grammar is particularly large and robust.
- Detailed information and all the necessary links are provided separately in the exercise file for the lesson.