Understanding LKB

Paul M. Heider

Department of Linguistics
University at Buffalo
The State University of New York

October 27, 2008
1 LKB Introduction

2 Options for Running LKB
   Loading the TTY System

3 Interpreting the Semantics

4 Adding to the Lexicon

5 Appendices

6 Appendices
Common Ground Terms

LKB  Linguistic Knowledge Builder
ERG  English Resource Grammar
MRS  Minimal Recursion Semantics
parser  e.g., constraint-based, statistical, CFG
lexicon
grammar
predicate
argument
syntax
semantics
1 LKB Introduction

2 Options for Running LKB
   Loading the TTY System

3 Interpretting the Semantics

4 Adding to the Lexicicon

5 Appendices

6 Appendices
System Requirements for GUI

1. Check the “LKB Installation Wiki” for full details

<---! Quick GUI Demo -->
System Requirements for TTY

1. Remember your CSE account password
2. `mkdir ~/tmp`
3. Find `alisp` (or case-insensitive upper equivalent)
Come talk to me later. This one is still messy.
1. LKB Introduction

2. Options for Running LKB
   Loading the TTY System

3. Interpreting the Semantics

4. Adding to the Lexicon

5. Appendices

6. Appendices
Loading the System

run-alisp
Loading the System

run-alisp

(load "~/projects/pmheider/567/load_LKB_with_ERG")
(load "~/projects/pmheider/567/load_LKB_utilities")

(see the appendix for more details on each file)
(read-cached-lex-if-available
 (list
  (lkb-pathname (parent-directory) "lexicon.tdl"))
(read-cached-lex-if-available
 (list
  (lkb-pathname (parent-directory) "lexicon.tdl")
  (lkb-pathname (parent-directory) "sneps.tdl")
  (lkb-pathname (parent-directory) "567.tdl")
)
... include: sneps.smi
include: 567.smi
Some Key Terms for LKB

**TFS:** Typed Feature Structure

**AVM:** Attribute-Value Matrix

**sign:** a pairing of structure (usually an AVM) and a string

**phrasal sign:** a well-formed TFS and a list of strings that can be mapped to it

**a parse:** building as many phrases as needed to connect the lexical signs to the start sign
ERG Files
An Quick Overview of ERG Files by Their Functions

Types and Constraints fundamentals, lextypes, syntax, lexrules, auxverbs, arboretum/mal-types, mtr
Lexical Entries lexicon, arboretum/mal-lex
Grammar Rules constructions, arboretum/mal-constructions
Lexical and Morphological Rules inflr, inflr-pnct, arboretum/mal-inflr
Start Symbol Description roots
Lexical Rules? lexrinst, arboretum/mal-lexrinst
Parse Node Description parse-nodes
1 LKB Introduction

2 Options for Running LKB
   Loading the TTY System

3 Interpreting the Semantics

4 Adding to the Lexicon

5 Appendices

6 Appendices
Copestake et al. [2005]
Minimal Recursion Semantics Introduction
Unique Features

• an E(lementary) P(redicate) will normally correspond to a single lexeme
Unique Features

- an E(lementary) P(redicate) will normally correspond to a single lexeme
- MRS is syntactically flat, i.e. one EP is never internal to another EP
Unique Features

- an E(lementary) P(redicate) will normally correspond to a single lexeme
- MRS is syntactically flat, i.e. one EP is never internal to another EP
- MRS was/is designed to integrate well with feature structure frameworks, e.g. HPSG
Unique Features

- an E(lementary) P(redicate) will normally correspond to a single lexeme
- MRS is syntactically flat, i.e. one EP is never internal to another EP
- MRS was/is designed to integrate well with feature structure frameworks, e.g. HPSG
- Assumes a non-floating top node (ltop for a phrase and gtop for the sentence) to ground the tree
Why Use Flat Semantics?

- Language often produces ambiguous or trivial bracketing that can only be resolved at a large computational cost
  
  (1)  
  a \quad \lambda x [\text{fierce}(x) \land (\text{black}(x) \land \text{cat}(x))] 
  
  b \quad \lambda x [\text{gato}(x) \land (\text{negro}(x) \land \text{feroz}(x))] 
  
  c \quad \lambda x [\text{cat}(x) \land (\text{black}(x) \land \text{fierce}(x))]

- But we can’t just reduce everything to a bag o’ semantics because then we lose scope
  
  (4)  
  a  
  \text{fierce black cat}  
  \text{fierce}(x), \text{black}(x), \text{cat}(x)

  b  
  \text{the beginning of spring arrives}  
  \text{the}(y), \text{beginning}(y, x), \text{def}(x), \text{spring}(x), \text{arrive}(e, y)
Every Big White Horse Sleeps I

First: create a standard tree

(8)  

a. Every big white horse sleeps.
b. every($x, \land (big(x), \land (white(x), horse(x))))., sleep(x))
c. every($x)

[Diagram of a tree structure]

$\land$

[Branches of the tree structure]
Every Big White Horse Sleeps I

- Next: Assume conjunction is n-ary

\[(9) \quad \text{every}(x, \land (\text{big}(x), \text{white}(x), \text{horse}(x)), \text{sleep}(x))\]
Every Big White Horse Sleeps III

- Because we always treat conjunction as n-ary and the base form, we can remove it explicitly from the formula

(10)

\[
\text{every}(x) \quad \text{big}(x), \text{white}(x), \text{horse}(x) \quad \text{sleep}(x)
\]

- Reify your like EP

(11)

\[
\text{every}(x) \quad \text{big}(x), \text{white}(x), \text{horse}(x) \quad \text{sleep}(x)
\]

- Finally: flatten the tree using the reified labels as handles for common EPs

(12) \ h0: \text{every}(x, h1, h2), h1: \text{big}(x), h1: \text{white}(x), h1: \text{horse}(x), h2: \text{sleep}(x)
<!-- TTY Demo -->
1. LKB Introduction
2. Options for Running LKB
   Loading the TTY System
3. Interpreting the Semantics
4. Adding to the Lexicon
5. Appendices
6. Appendices
dumbo := n__pn-msc_le &
[STEM < "dumbo">,
SYNSEM [LKEYS.KEYREL.CARG "dumbo",
PHON.ONSET con ]].
Some Generic Lexicon Entries

dumbo := n__pn-msc_le &
[ STEM < "dumbo" >,
  SYNSEM [ LKEYS.KEYREL.CARG "dumbo",
            PHON.ONSET con ] ].

canary_n1 := n__c_le &
[ STEM < "canary" >,
  SYNSEM [ LKEYS.KEYREL.PRED "_canary_n_1_rel",
            PHON.ONSET con ] ].
Some Generic Lexicon Entries

dumbo := n__pn-msc_le &
[ STEM < "dumbo" >,
  SYNSEM [ LKEYS.KEYREL.CARG "dumbo",
           PHON.ONSET con ] ].

canary_n1 := n__c_le &
[ STEM < "canary" >,
  SYNSEM [ LKEYS.KEYREL.PRED "_canary_n_1_rel",
           PHON.ONSET con ] ].

harbinger_n3052487 := n__c_le &
[ STEM < "harbinger" >,
  SYNSEM [ LKEYS.KEYREL.PRED "harbinger_n_1_rel",
           PHON.ONSET CON ] ].
Some Generic Predicate Entries

"harbinger_n_1_rel" : ARG0 x.
"harbinger_n_1_rel" : ARG0 x, ARG1 x.
"harbinger_v_1_rel" : ARG0 e, ARG1 p, ARG2 x.
<!-- Augmenting the Lexicon Demo -->
(defvar *lkb-directory* "~/projects/pmheider/delphin/lkb/")
(defvar *erg-directory* "~/projects/pmheider/delphin/erg_567/")
(defvar *sfy-directory* "~/projects/pmheider/567/")

(pushnew :tty *features*)

(load (concatenate 'string *lkb-directory* 
          "src/general/loadup"))

(pushnew :lkb *features*)

(load-system :lkb)

(lkb::read-script-file-aux (concatenate 'string *erg-directory* 
                                 "lkb/script"))

(setf lkb::*show-parse-p* nil)

(setf *sfy-parse-mode* "nested-lists")

(setf *sfy-quiet-parse* t)