Outline

1 A System Overview: LKB, MRS, HPSG
   LKB
   Copestake [2002]
   ERG
   MRS
   MRS Introduction

2 Bridging the Gap
   New Code and Formalisms

3 Demos
   Clyde

4 The Short List

5 Bibliography

6 Appendices
run-alisp

(load "~/projects/pmheider/sfy/load_LKB_with_ERG")
Loading the System

run-alisp

(load "/projects/pmheider/sfy/load_LKB_with_ERG")

:ld /projects/snwiz/bin/sneps
run-alisp

(load "~/projects/pmheider/sfy/load_LKB_with_ERG")

:ld ~/projects/snwiz/bin/sneps

(load "~/projects/pmheider/sfy/util.cl")

(Clyde is an elephant.)
run-alisp

(load "~/projects/pmheider/sfy/load_LKB_with_ERG")

:ld ~/projects/snwiz/bin/sneps

(load "~/projects/pmheider/sfy/util.cl")

(load "~/projects/pmheider/sfy/parse.cl")
run-alisp

(load "/projects/pmheider/sfy/load_LKB_with_ERG")

:ld /projects/snwiz/bin/sneps

(load "/projects/pmheider/sfy/util.cl")

(load "/projects/pmheider/sfy/parse.cl")

(sfy::parse "Clyde is an elephant.")
load LKB with ERG Details

(pushnew :tty *features*)
(load "~/projects/pmheider/delphin/lkb/src/general/loadup")
(pushnew :lkb *features*)
(load-system :lkb)
(lkb::read-script-file-aux "~/projects/pmheider/delphin/erg_test/lkb/script")
script Details

1. Types and Constraints
   (read-tdl-type-files-aux
      (list
       (lkb-pathname (parent-directory) "fundamentals.tdl")
       . . .))

2. Lexical Entries
   (read-cached-lex-if-available
      (list
       (lkb-pathname (parent-directory) "lexicon.tdl")
       (lkb-pathname (parent-directory) "sneps-lex.tdl")
       . . .))

3. Grammar Rules
   (read-tdl-grammar-file-aux
    (lkb-pathname (parent-directory) "constructions.tdl"))
1 Types and Constraints

(read-tdl-type-files-aux
 (list
   (lkb-pathname (parent-directory) "fundamentals.tdl")
   . . .))

2 Lexical Entries

(read-cached-lex-if-available
 (list
   (lkb-pathname (parent-directory) "lexicon.tdl")
   (lkb-pathname (parent-directory) "sneps-lex.tdl")
   . . .))

3 Grammar Rules

(read-tdl-grammar-file-aux
 (lkb-pathname (parent-directory) "constructions.tdl"))
Types and Constraints

(read-tdl-type-files-aux
 (list
  (lkb-pathname (parent-directory) "fundamentals.tdl")
  . . .))

Lexical Entries

(read-cached-lex-if-available
 (list
  (lkb-pathname (parent-directory) "lexicon.tdl")
  (lkb-pathname (parent-directory) "sneps-lex.tdl")
  . . .))

Grammar Rules

(read-tdl-grammar-file-aux
 (lkb-pathname (parent-directory) "constructions.tdl"))
4 Lexical & Morphological Rules

(read-morph-file-aux
   (lkb-pathname (parent-directory) "inflr.tdl"))

5 Valid Top Symbol(s)

(read-tdl-psort-file-aux
   (lkb-pathname (parent-directory) "roots.tdl"))

6 Tree Label Identifiers

(read-tdl-parse-node-file-aux
   (lkb-pathname (parent-directory) "parse-nodes.tdl"))
4 Lexical & Morphological Rules
(read-morph-file-aux
  (lkb-pathname (parent-directory) "inflr.tdl"))

5 Valid Top Symbol(s)
(read-tdl-psort-file-aux
  (lkb-pathname (parent-directory) "roots.tdl"))

6 Tree Label Identifiers
(read-tdl-parse-node-file-aux
  (lkb-pathname (parent-directory) "parse-nodes.tdl"))
Lexical & Morphological Rules
(read-morph-file-aux
   (lkb-pathname (parent-directory) "inflr.tdl"))

Valid Top Symbol(s)
(read-tdl-psort-file-aux
   (lkb-pathname (parent-directory) "roots.tdl"))

Tree Label Identifiers
(read-tdl-parse-node-file-aux
   (lkb-pathname (parent-directory) "parse-nodes.tdl"))
Outline

1. A System Overview: LKB, MRS, HPSG
   - LKB
     - Copestake [2002]
   - ERG
   - MRS
     - MRS Introduction

2. Bridging the Gap
   - New Code and Formalisms

3. Demos
   - Clyde

4. The Short List

5. Bibliography

6. Appendices
LKB: A Constraint-Based Parser
Copestake [2002]: Implementing Typed Feature Structure Grammars
Acronyms

**TFS**: Typed Feature Structure

**AVM**: Attribute-Value Matrix

**TDL**: Type Description Language

**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
Outline

1 A System Overview: LKB, MRS, HPSG
   LKB
   Copestake [2002]
   ERG
   MRS
   MRS Introduction

2 Bridging the Gap
   New Code and Formalisms

3 Demos
   Clyde

4 The Short List

5 Bibliography

6 Appendices
ERG: A General-Purpose Grammar
ERG Files
ERG Files Loaded and Their Functions

A long-term TODO includes grokking through these files to find elements we’ll frequently wish to imitate/duplicate.

**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
generic_trans_verb_bse := v_np*_bse-unk_le &
    [ STEM < *top* > ].

generic_trans_verb_pres3sg := v_np*_pr-3s-unk_le &
    [ STEM < *top* > ].

generic_trans_verb_presn3sg := v_np*_pr-n3s-unk_le &
    [ STEM < *top* > ].

generic_trans_verb_past := v_np*_pa-unk_le &
    [ STEM < *top* > ].

generic_trans_verb_psp := v_np*_psp-unk_le &
    [ STEM < *top* > ].

generic_trans_verb_prp := v_np*_prp-unk_le &
    [ STEM < *top* > ].

generic_sg_noun := n_-_c-sg-unk_le &
    [ STEM < *top* > ].

generic_mass_noun := n_-_m-unk_le &
    [ STEM < *top* > ].

generic_pl_noun := n_-_c-pl-unk_le &
    [ STEM < *top* > ].

generic_title_noun := n_-_c-tt-unk_le &
    [ STEM < *top* > ].

genericname := n_-_pn-unk_le &
    [ STEM < *top* > ].

generic_adj := aj_-_i-unk_le &
    [ STEM < *top* > ].

generic_adj_compar := aj_-_i-cmp-unk_le &
    [ STEM < *top* > ].

generic_adj_superl := aj_-_i-sup-unk_le &
    [ STEM < *top* > ].

generic_adverb := av_-_i-unk_le &
    [ STEM < *top* > ].
Outline

1. A System Overview: LKB, MRS, HPSG
   - LKB
     - Copestake [2002]
   - ERG
   - MRS
     - MRS Introduction

2. Bridging the Gap
   - New Code and Formalisms

3. Demos
   - Clyde

4. The Short List

5. Bibliography

6. Appendices
MRS: A Flat-Semantics Representation
Copestake et al. [2005]: Minimal Recursion Semantics
Introduction
Goals of the Paper

Definition
MRS stands for Minimal Recursion Semantics

Definition
EP stands for Elementary Predicate
Unique Features

- EP 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
Unique Features

- EP 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
Unique Features

- EP 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
Unique Features

- EP 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

\[
\begin{align*}
(1) & \quad a \quad \lambda x [\text{fierce}(x) \land (\text{black}(x) \land \text{cat}(x))] \\
& \quad b \quad \lambda x [\text{gato}(x) \land (\text{negro}(x) \land \text{feroz}(x))] \\
& \quad c \quad \lambda x [\text{cat}(x) \land (\text{black}(x) \land \text{fierce}(x))] 
\end{align*}
\]

- But we can’t just reduce everything to a bag o’ semantics because then we lose scope

\[
\begin{align*}
(4) & \quad a \quad \text{fierce black cat} \\
& \quad \text{fierce}(x), \text{black}(x), \text{cat}(x) \\
& \quad b \quad \text{the beginning of spring arrives} \\
& \quad \text{the}(y), \text{beginning}(y, x), \text{def}(x), \text{spring}(x), \text{arrive}(e, y)
\end{align*}
\]
Every Big White Horse Sleeps I

- First: create a standard tree

(8) a Every big white horse sleeps.
    b every(x, \( \land (\text{big}(x), \land (\text{white}(x), \text{horse}(x))), \text{sleep}(x) \))
    c

\[
\begin{align*}
\text{every}(x) \\
\land \\
\text{sleep}(x) \\
\land \\
\text{big}(x) \\
\land \\
\text{white}(x) & \quad \text{horse}(x)
\end{align*}
\]
Every Big White Horse Sleeps I

• Next: Assume conjunction is n-ary

(9) \( a \) every\( (x, \wedge (\text{big}(x), \text{white}(x), \text{horse}(x)), \text{sleep}(x)) \)

\( b \)

\( \wedge \)

\( \text{every}(x) \)

\( \wedge \)

\( \text{sleep}(x) \)

\( \text{big}(x) \)

\( \text{white}(x) \)

\( \text{horse}(x) \)
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

\[(10)\]

![Diagram of logical structure]

- Reify your like EP

\[(11)\]

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

\[(12)\] \(h0: \text{every}(x, h1, h2), h1: \text{big}(x), h1: \text{white}(x), h1: \text{horse}(x), h2: \text{sleep}(x)\)
MRS Structure

(12) \( h_0: \text{every}(x, h_1, h_2), h_1: \text{big}(x), h_1: \text{white}(x), h_1: \text{horse}(x), h_2: \text{sleep}(x) \)

**handle:** The tag associated with a reified EP

**label:** A handle used to tag a node on the semantic tree

**EP:** That which follows the label on a node

**QEQ:** \( h_1 \text{ qeq } h_3 \) is used to add the scopal information that either \( h_1 \) and \( h_3 \) are identical or close enough to not change the reading

<GTOP, LTOP, Bag o’ EPs, Scopal Restrictions>
MRS Structure

(12) \( h_0: \text{every}(x, h_1, h_2), h_1: \text{big}(x), h_1: \text{white}(x), h_1: \text{horse}(x), h_2: \text{sleep}(x) \)

**handle:** The tag associated with a reified EP

**label:** A handle used to tag a node on the semantic tree

**EP:** That which follows the label on a node

**QEQ:** \( h_1 \text{ qeq } h_3 \) is used to add the scopal information that either \( h_1 \) and \( h_3 \) are identical or close enough to not change the reading


&lt;GTOP, LTOP, Bag o’ EPs, Scopal Restrictions&gt;
Example Output from Clyde

(sfy::parse "Clyde is an elephant.")

<h1,e2,
  {h1:prpstmt_m(e2, h3, u4, u5),
   h6:proper_q(x8:-, h7, h9),
   h10:named(x8, "clyde"),
   h11:_be_v_id(e2, x8, x12:-),
   h13:_a_q(x12, h15, h14),
   h16:_elephant_n_1(x12)},
  {h3 qeq h11,
   h7 qeq h10,
   h15 qeq h16}>
Example Output from Clyde

(sfy::parse "Clyde is an elephant.")

<h1,e2,
   {h1:prpstmt_m(e2, h3, u4, u5),
    h6:proper_q(x8:-, h7, h9),
    h10:named(x8, "clyde"),
    h11:_be_v_id(e2, x8, x12:-),
    h13:_a_q(x12, h15, h14),
    h16:_elephant_n_1(x12)},
   {h3 qeq h11,
    h7 qeq h10,
    h15 qeq h16}>
1. \( \text{Ep}(\text{handle}0, \text{pred0}(\text{type1}, \text{handle}1), \ldots, \text{pred0}(\text{typen}, \text{handlen})). \)

2. \( \text{Ep}(\text{handle}0, \text{pred0}, \text{pred0}(\text{type1}, \text{handle}1), \ldots, \text{pred0}(\text{typen}, \text{handlen})). \)

3. \( \text{Mrs}(\text{handle}0, \text{Ep}(\text{pred0}, \text{handle}1, \ldots, \text{handlen})). \)

4. \( \text{EP}(\text{handle}0, \text{pred0}, \text{handle/argument}1, \ldots, \text{handle/argumentn}). \)
1. \( \text{Ep}(\text{handle}0, \text{pred}0(\text{type}1, \text{handle}1), \ldots, \text{pred}0(\text{typen}, \text{handlen})) \). 

2. \( \text{Ep}(\text{handle}0, \text{pred}0, \text{pred}0(\text{type}1, \text{handle}1), \ldots, \text{pred}0(\text{typen}, \text{handlen})) \). 

3. \( \text{Mrs}(\text{handle}0, \text{Ep}(\text{pred}0, \text{handle}1, \ldots, \text{handlen})) \). 

4. \( \text{EP}(\text{handle}0, \text{pred}0, \text{handle}/\text{argument}1, \ldots, \text{handle}/\text{argument}n) \).
1. $\text{Ep}(\text{handle}0,$
   $\text{pred}0(\text{type}1, \text{handle}1), \ldots, \text{pred}0(\text{type}n, \text{handle}n))$.

2. $\text{Ep}(\text{handle}0, \text{pred}0,$
   $\text{pred}0(\text{type}1, \text{handle}1), \ldots, \text{pred}0(\text{type}n, \text{handle}n))$.

3. $\text{Mrs}(\text{handle}0, \text{Ep}(\text{pred}0, \text{handle}1, \ldots, \text{handle}n))$.

4. $\text{EP}(\text{handle}0, \text{pred}0, \text{handle/argument}1, \ldots,$
   $\text{handle/argument}n)$.
1. Ep(handle0, 
   pred0(type1, handle1), . . ., pred0(typen, handlen)).

2. Ep(handle0, pred0, 
   pred0(type1, handle1), . . ., pred0(typen, handlen)).

3. Mrs(handle0, Ep(pred0, handlel1, . . ., handlen)).

4. EP(handle0, pred0, handle/argument1, . . ., handle/argumentn).
Information Flow

“Clyde ...” → LKB → ERG → Parse₁ → Scope₁ → Gold Standard → SNePSify
   Scope₂ →
   Scope₃ →
Parse₂ → Scope₄ →
Parse₃ → Scope₅ →
   Scope₆ →
Information Flow

“Clyde . . .” → LKB → ERG → Parse$_1$ → Scope$_1$ → Gold Standard → SNePSify
Scope$_2$ → Scope$_3$ →
Parse$_2$ → Scope$_4$ →
Parse$_3$ → Scope$_5$ →
Scope$_6$ →
“Clyde ...” → LKB → ERG → Parse_1 → Scope_1 → Gold Standard → SNePSify
    Scope_2 →
    Scope_3 →
    Parse_2 → Scope_4 →
    Parse_3 → Scope_5 →
    Scope_6 →
“Clyde . . .” → LKB → ERG → Parse₁ → Scope₁ → Gold Standard → SNePSify
Scope₂ →
Scope₃ →
Parse₂ → Scope₄ →
Parse₃ → Scope₅ →
Scope₆ →
Information Flow

“Clyde ...” → LKB → ERG → Parse$_1$ → Scope$_1$ → Gold Standard → SNePSify
  Scope$_2$ →
  Scope$_3$ →
Parse$_2$ → Scope$_4$ →
Parse$_3$ → Scope$_5$ →
Scope$_6$ →
“Clyde . . .” → LKB → ERG → Parse$_1$ → Scope$_1$ → Gold Standard → SNePSify
  Scope$_2$ →
  Scope$_3$ →
  Parse$_2$ → Scope$_4$ →
  Parse$_3$ → Scope$_5$ →
  Scope$_6$ →
"Clyde ..." → LKB → ERG → Parse₁ → Scope₁ → Gold Standard → SNePSify
  Scope₂ →
  Scope₃ →
  Parse₂ → Scope₄ →
  Parse₃ → Scope₅ →
  Scope₆ →
“Clyde ...” → LKB → ERG → Parse$_1$ → Scope$_1$ → Gold Standard → SNePSify
Scope$_2$ →
Scope$_3$ →
Parse$_2$ → Scope$_4$ →
Parse$_3$ → Scope$_5$ →
Scope$_6$ →
Information Flow: Revised

“Clyde . . .” → LKB → ERG → Parse$_1$ → Scope$_1$ → SNePSify → Gold Standard

Scope$_2$ →
Scope$_3$ →
Parse$_2$ → Scope$_4$ →
Parse$_3$ → Scope$_5$ →
Scope$_6$ →
Outline

1. A System Overview: LKB, MRS, HPSG
   - LKB
     - Copestake [2002]
   - ERG
   - MRS
     - MRS Introduction

2. Bridging the Gap
   - New Code and Formalisms

3. Demos
   - Clyde

4. The Short List

5. Bibliography

6. Appendices
1. A System Overview: LKB, MRS, HPSG
   - LKB
     - Copestake [2002]
   - ERG
   - MRS
     - MRS Introduction

2. Bridging the Gap
   - New Code and Formalisms

3. Demos
   - Clyde

4. The Short List

5. Bibliography

6. Appendices
Clyde, Dumbo, Tweety, and Opus
Clyde Data

- Clyde is an elephant.
- Dumbo is an elephant.
- Dumbo can fly.
- Tweety is a canary.
- Tweety can fly.
- Opus is a bird.
- Who can fly?
Additions to the SNePSLexicon

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

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

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

opus := n__pn-msc_le &
[ STEM < "opus" >,
  SYNSEM [ LKEYS.KEYREL.CARG "opus",
           PHON.ONSET voc ] ].
Additions to the SNePSLexicon

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

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

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

opus := n__pn-msc_le &
[ STEM < "opus" >,
  SYNSEM [ LKEYS.KEYREL.CARG "opus",
    PHON.ONSET voc ] ].
• (sfy::parse "Clyde is an elephant.")
• (sfy::parse "Dumbo is an elephant.")
• (sfy::parse "Dumbo can fly.")
• (sfy::parse "Tweety is a canary.")
• (sfy::parse "Tweety can fly.")
• (sfy::parse "Opus is a bird.")
• (sfy::parse "Who can fly?")
Existential Problems: To ‘be’ or not to ‘be’ I

Modifying

- Clyde is happy
- Clyde is white
- Clyde is tall

Co-referential

- Clyde is Dumbo
- The evening star is the morning star
Existential Problems: To ‘be’ or not to ‘be’ II

Weakly Co-referential

- Clyde is an elephant
- An elephant is a mammal

Existential

- Clyde is.
Existential Problems: To ‘be’ or not to ‘be’ III

Shell/Place Filler

• It is hot
• It is raining

Indefinite Balance Examples

• An elephant is in the room
• An elephant is dead
• An elephant is tall
• KR Related
  • Find standard MRSrepresentations for input/query frames
  • Create Act/Plans for interrogatives

• Vocabulary Related
  • List words from input unknown to LKB or known to LKB but unknown to SNePS
  • Import (said) words
  • Create dummy words on the fly

Available TTY Commands

show-type-spec-tty (type)
show-type-tty (type)
show-lex-tty (lex)
show-other-tty (other-id)
show-word-defs-tty (word-string)
show-words-tty (word-string)
show-grammar-rule-tty (name)
show-lex-rule-tty (name)
apply-lex-tty (lex lex-rule-name)
apply-lex-rules-tty (lex)
display-fs-tty (fs)
display-fs (fs title &optional id)
show-chart ()
display-fs-and-paths-tty (fs paths)
display-fs-and-parents-tty (fs parents)
Pierce’s of Code

**subtype-p.**
Returns true if X (or X’s type) is a subtype of TYPE.
X is mrs::rel or mrs::var
TYPE is symbol

**get-feature-value.**
Returns the value of feature F in feature-bearing object X, or
NIL if X has no value for F.
X is FVPAIR, EXTRAPAIR, VAR, or REL
F is a feature symbol or a path list (the last feature along this
path will be used)

***bindings*.**
A cache of the current MRS’ bindings in the form:
(((1 . 1) (3 . 3) (6 . 3) (5 . 5) (18 . 5) (9 . 9) (16 . 9))

**scope-mrs.**
Function that actually caches the bindings of a psoa