Grammar Engineering for Deep Linguistic Processing SS2012

Lecture 3: TDL

Yi Zhang

Department of Computational Linguistics & Phonetics
Saarland University

Language Technology Lab
German Research Center for Artificial Intelligence

May, 2012

Outline

1 Typed Description Language

Outline

1 Typed Description Language

TDL— A Type Description Language for Constraint-Based Grammars

- [Krieger and Schäfer, 1994]
- Originally used in PAGE system
- Simplification and extensions in LKB
  $\Rightarrow$ DELPH-IN reference formalism
- Fully compatible implementation in PET

TDL Syntax — Examples

- Type inheritance:

```
feat-struc := *top*.
```

or

```
feat-struc :< *top*.
```

- Type inheritance with attribute-value constraints:

```
agr-cat := gen-agr-cat &
[ PER per,
  NUM num,
  GEND gend ].
```

- Multiple inheritance and coreference:

```
head-feat-principle := grule & head-dtr-type &
[ SYNSEM [ HEAD #head ],
  H-DTR [ SYNSEM [ HEAD #head ] ] ].
```

TDL Syntax

Formal Description

- Type-def → Type Avm-def . | Type Subtype-def .
- Type → identifier
- Subtype-def → :< Type
- Avm-def → := Conjunction
- Conjunction → Term | Term & Conjunction
- Term → Type | string | Feature-term | Coreference
- Feature-term → [ ] | [Attr-val-list]
- Attr-val-list → Attr-val | Attr-val, Attr-val-list
- Attr-val → Attr-list Conjunction
- Attr-list → Attribute | Attribute.Attr-list
- Attribute → identifier
- Coreference → #identifier

TDL Syntax — Continued

- **top** is a built-in type in LKB
- Identifiers are composed of *{a-z, A-Z, 0-9, _, -, +, *, ?}*
  - Identifiers are case-insensitive
  - Conventionally, attributes in upper cases, types in lower cases
- Lisp-style comments:
  - Single line comments are started with ;
  - Multi-line comments are bracketed by #| |#

Lists

```
list :< *top*.
e-list :< list.
ne-list := list &
             [ FIRST *top*,
               REST list ].
```

List Abbreviations

- <a,b,c>

```
[ FIRST a,
  REST [ FIRST b,
         REST [ FIRST c,
                REST e-list ] ] ]
```

- <a,b,c,...>

```
[ FIRST a,
  REST [ FIRST b,
         REST [ FIRST c,
                REST list ] ] ]
```

- <a.b>

```
[ FIRST a,
  REST b ]
```

Difference Lists

- Allows more flexible list operation: concatenation, append, remove from end, . . . , simply using unification.
- Definition:

```
*diff-list* := *top &
[ LIST *list*,
  LAST *list* ].
```

  - `LIST` points to the beginning position
  - `LAST` points to the end position

Difference Lists — Continued

- Abbreviation: `<!a,b,c!>`

```
[ LIST [ FIRST a,
         REST [ FIRST b,
                REST [ FIRST c,
                       REST #last ] ] ],
  LAST #last ]
```

List & Diff-list

Formal Description

- Term $\rightarrow$ Type | string | Feature-term | Coreference | List | Diff-list
- Diff-list $\rightarrow$ $<!\ !>$ | $<!$ Conjunction-list $!>$
- Conjunction-list $\rightarrow$ Conjunction | Conjunction, Conjunction-list
- List $\rightarrow$ $<>$ | $<$ Conjunction-list $>$ | $<$ Conjunction-list, ... $>$ | $<$ Conjunction-list . Conjunction $>$

Lexical Entries

Formal Description

- Lexentry → LexID Avm-def .
- LexID → identifier

```
me_1 := pron-lxm &
[ ORTH "me",
  SYNSEM [ HEAD noun &
                  [ AGR non-3sing &
                        [ PER 1st ],
                    CASE acc ] ] ].
```

- Should not be confused with *types*

Grammar Rules

Formal Description

- Ruleentry $\rightarrow$ RuleID Avm-def .
- RuleID $\rightarrow$ identifier

```
binary-rule := rule &
                    [ ARGS < sign, sign > ].
```

- A means of constructing new signs
- Have a definite number of daughters
- Lexical rules are treated as unary grammar rules that may apply before affixation

Morphological Rules

Formal Description

- Mruleentry → RuleID Mgraph-spec-list Avm-def .
- Mgraph-spec-list → Mgraph-spec | Mgraph-spec Mgraph-spec-list
- Mgraph-spec → **%prefix** SPair-list | **%suffix** SPair-list
- SPair-list → SPair | SPair SPair-list
- SPair → (* Char-list) | (Char-list Char-list)
- Char-list → letter | Macro | letter Char-list | Macro Char-list
- Letterset → % (**letter-set** (Macro letters))
- Macro → !letter
- Irregentry → base Rulespec inflected

Morphological Rules

Letter sets

Formal Description

- Letterset → % (**letter-set** (Macro letters))
- Macro → !letter

```
%(letter-set (!c bdfglmnprstz))
%(letter-set (!s abcdefghijklmnopqrtuvwxyz))
%(letter-set (!t bcdfghjklmnpqrstvwxz))
%(letter-set (!v aeiou))
```

Morphological Rules

Inflectional rules

Formal Description

- Mruleentry → RuleID Mgraph-spec-list Avm-def .
- Mgraph-spec-list → Mgraph-spec | Mgraph-spec Mgraph-spec-list
- Mgraph-spec → **%prefix** SPair-list | **%suffix** SPair-list
- SPair-list → SPair | SPair SPair-list
- SPair → (* Char-list) | (Char-list Char-list)
- Char-list → letter | Macro | letter Char-list | Macro Char-list

```
past_verb_infl_rule :=
%suffix (* ed) (!ty !tied) (e ed)
lex_rule_infl_affixed &
[ NEEDS-AFFIX +,
  ARGS [ FIRST [ AFFIX past_verb ] ] ].
```

Morphological Rules

Irregular forms

Formal Description

- Irregentry $\rightarrow$ inflected Rulespec base

```
"
saw PAST-VERB see
seen PSP-VERB see
"
```

References I

Krieger, H.-U. and Schäfer, U. (1994).
TDL - a Type Description Language for HPSG.
Technical Report RR-94-37, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH.