L³XDG 0.9 Quick Reference

Michael Gasser
Indiana University, School of Informatics and Computing
hltdi
gasser@cs.indiana.edu
1 January, 2011

Background

L³XDG is a Python implementation of Extensible Dependency Grammar (Debusmann et al., 2004; Debusmann, 2007), which is built on the constraint satisfaction framework Mozart/Oz (Van Roy, 2004). It also includes small grammar fragments for English and Amharic within the content domain of WATER. This sketchy reference assumes familiarity with XDG.

Installation

1. Uncompress the file that you downloaded. This will yield a directory (folder) called `L3XDG-0.9`, which contains all of the files that you need to run L³XDG.
2. Go to the `L3XDG-0.9` directory (folder), and enter the following, making sure that you are running Python 3.0 or 3.1.

```
python setup.py install
```

Use

STARTING THE PROGRAM

The program can only be used within the Python interpreter. Start up the interpreter, again making sure that you are running at least Python 3.0, and type the following to load the program.

```
import l3xdg
```

FUNCTIONS

XDG(*sentence*, *source_language*)
  Options: target=[], grammar='water', verbosity=0

This function, the constructor for the XDG class, returns an XDG object (a kind of constraint satisfaction problem, CSP), consisting of a set of nodes representing the words in the input sentence and the variables and constraints resulting from lexicalizing the words. The parameter `sentence` is a

string representation of the input sentence. The parameter `source_language` is a string abbreviation of the language of the input sentence (`'en'` or `'am'`). The optional parameter `target` is a list of languages to translate into. Other optional parameters include `grammar`, which names a particular grammar and lexicon for the language(s) and defaults to the `'water'`, and `verbosity`, which prints out various messages when it is greater than 0.

```
>>> l3xdg.XDG('the water froze', 'en')
<CSP the water froze .>

>>> l3xdg.XDG('አስቴር ውሃውን አጠለለችው', 'am')
<CSP አስቴር ውሃውን አጠለለችው ።>
```

`solve``()`
Options: `prop_verbosity=1, dist_verbosity=0`

This XDG method runs constraint satisfaction on the XDG object, returning a list of solutions in the form of Multigraph objects. The two optional parameters, when non-zero, print out messages concerning propagation or distribution, the two steps in constraint satisfaction.

```
>>> enam1 = l3xdg.XDG('Esther filtered water', 'en', target=['am'])

>>> enam1.solve()
[<MG b/1 Esther filtered water .>, <MG a/1 Esther filtered water .>]
```

`pprint``()`

This Multigraph method prints out information about the Multigraph. For each arc dimension (interface dimensions are not listed), each node's daughters, agreement feature values, and target language position are shown.

```
>>> enam_mg = l3xdg.XDG('Almaz filtered water', 'en', target=['am']).solve()

>>> enam_mg[0].pprint()
<MG b/1 Esther filtered water .>

Dimension en-lp
  &2water
  &3.
    Daughters:: root:{&1filtered}
  &1filtered
    Daughters:: mf2:{&2water} vf:{&0Esther}
  &0Esther
Dimension am-syn
  &2water
    Agrs:: png:(3, 1, 1) def:(0,) pos:n
    Position:: 0
  &3.
    Daughters:: root:{&1filtered}
    Position:: 3
  &1filtered
    Daughters:: sb:{&0Esther} ob:{&2water}
    Agrs:: sub:(0,) der:(3, 1) ob:(3, 1, 1) pos:v sb:(3, 1, 2) defob:(0,)
    Position:: 2
  &0Esther
    Agrs:: png:(3, 1, 2) prp:(1,) def:(1,) pos:n
    Position:: 1
```

```
Dimension sem
  &2water
    Agrs:: def:(0,)
  &3.
    Daughters:: root:{&1filtered}
  &1filtered
    Daughters:: arg1:{&0Esther} arg2:{&2water}
  &0Esther
    Agrs:: def:(1,)
Dimension en-id
  &2water
    Agrs:: pn:(3, 1) def:(0,)
  &3.
    Daughters:: root:{&1filtered}
  &1filtered
    Daughters:: sb:{&0Esther} ob:{&2water}
    Agrs:: tns:(1,)
  &0Esther
    Agrs:: pn:(3, 1) def:(1,)
```

`io()`

This Multigraph method prints out the input sentence and sentences in any target languages.

```
>>> for mg in enam_mg: mg.io()
...
Input (English): Esther filtered water .
Output (Amharic): ውሀ አስቴር አጠለለች .
Input (English): Esther filtered water .
Output (Amharic): አስቴር ውሀ አጠለለች .
```

References

Debusmann, R. (2007). *Extensible Dependency Grammar: a modular grammar formalism based on multigraph description.* Ph.D. Thesis. Universität des Saarlandes.

Debusmann, R., Duchier, D., and Kruijff, G-J. M. (2004). Extensible Dependency Grammar: a new methodology. *Proceedings of the COLING 2004 Workshop on Recent Advances in Dependency Grammar.*

Van Roy, P (Ed.) (2005). *Multiparadigm Programming in Mozart/Oz, Second International Conference: revised, selected, and invited papers. Lecture Notes in Computer Science 3389*. New York: Springer.