GraphLang: A DMRS graph description language

Alexander Kuhnle

Computer Laboratory
University of Cambridge

DELPH-IN Summit, 2016
GraphLang
A DMRS graph description language

```
proper_q  named("Kim")  _eat_v_1  udef_q  _cake_n_1
```

Motivation:
▶ Succinct and easily read/writeable representation for DMRS
▶ DMRS formalism similar to MRS formalisms like Oepen et al. (2004)
▶ Useful if one wants to quickly specify a DMRS graph, e.g. for debugging
GraphLang
A DMRS graph description language

Motivation:
▶ Succinct and easily read-/writeable representation for DMRS
▶ DMRS formalism similar to MRS formalisms like Oepen et al. (2004)
▶ Useful if one wants to quickly specify a DMRS graph, e.g. for debugging
GraphLang
A DMRS graph description language

proper_q --> subj:named(Kim) x[3s+] <-1- _like_v_1 e(ppi--);
:_like_v_1 -2h-> _eat_v_1 e[pui--] -2-> _cake_n_1 x[3s__] <-- udef_q;
:_eat_v_1 -1-> :subj

Features:

- Sortinfo syntax (short form): e[pui--], texttttx[3s__]
- Node identifier via colon prefix: subj:named
- Referring back to nodes via leading colon: :_like_v_1, :subj
Matching & mapping

Search

node <-1- _eat_v_1 e? -2-> _?obj_n_1 x?

Features:

▶ Underspecification of nodes or parts of their properties
▶ Identifier suffix for querying
Matching & mapping

Search

Features:

- Underspecification of nodes or parts of their properties
- Identifier suffix for querying

Query tool:

```bash
> python3 dmrs.py "Kim likes to eat cake."
    | python3 query.py "node <-1- _eat_v_1 e? -2-> _?obj_n_1 x?"
    | {'obj': 'cake'}
```
Matching & mapping

Replace

Features:
- Node identifier (with square brackets) for mapping alignment
Matching & mapping

Replace

Features:
- Node identifier (with square brackets) for mapping alignment

Paraphrase tool:

> python3 paraphrase.py paraphrases.txt "Kim likes to eat cake."

Kim eats cake.
More specialised concepts

**Optional node:** “at (long) last” → “finally”

Search:

1. \[1] \texttt{:at} \texttt{p} \texttt{e[pui--]} -2- \rightarrow \texttt{.last.n.1 x[3s.+]} \leftarrow \texttt{idiom.q.i};
2. \[2] \texttt{:long.a.1 e[pui--]} =1- \rightarrow \texttt{.last.n.1}

Replace:

1. \[1] \texttt{:final.a.1 e[pui--]}
More specialised concepts

Optional node: “at (long) last” → “finally”
Search: [1]:at.p e[pui--] -2-> _last_n_1 x[3s_] <-- idiom_q.i;
    (2):_long_a_1 e[pui_] =1=> :_last_n_1
Replace: [1]:_final_a_1 e[pui--]

Subgraph node: “Kim eats apple cake.” → “What does Kim eat?”
Search: *[1]:v e[p????] -2-> {2}:node
Replace: *[1]:v e[q????] -2-> [2]:thing x <-- which_q
More specialised concepts

**Optional node:** “at (long) last” → “finally”

Search: \[1\]: at.p e[pui--] -2-> _last.n_1 x[3s+] <-- idiom.q.i;
(2): long.a_1 e[pui_] =1=> :_last.n_1
Replace: [1]: final.a_1 e[pui--]

**Subgraph node:** “Kim eats apple cake.” → “What does Kim eat?”

Search: *[1]:_v e[p????] -2-> {2}:node
Replace: *[1]:_v e[q????] -2-> [2]:thing x <-- which_q

**Equality constraint:** “I think I will go.” → “I am thinking of going.”

Search: [1]:node=1 <-1- [2]:_think.v_1 e[????-] -2h-> [3]:_v e[pfi--];
:3 -1-> node=1
Replace: [1]:node <-1- [2]:_think.v.of e[????+] -2-> nominalization x;
Udef_q --> :nominalization =1h=> [3]:_v e[pui+]
Applications

- Robust text query, e.g. for ontology extraction from WikiWoods
- Paraphrasing (examples in pydmrs)
- Sentence simplification/normalisation
- Machine translation, similar to the MRS transfer formalism of e.g. Bond et al. (2011) or Oepen et al. (2004)
- Mapping between graph formalisms or to/from simplified “DMRS graphs”, e.g. Guy’s robot language
- Other ideas?