A Comparative Introduction to XDG: Adding the Deep Syntax Dimension

Ralph Debusmann

and

Denys Duchier

Programming Systems Lab, Saarland University, Saarbrücken, Germany

and

Équipe Calligramme, LORIA, Nancy, France

This presentation

- adding the Deep Syntax (ds) dimension to the example grammar
- new:
  - type definitions
  - one-dimensional principles (dag, valency)
  - multi-dimensional principles (climbing, linking, linking^{-1})
  - lexical classes

Defining the new types

- edge labels:

```
deftype "ds.label" {detd subjd objd vcd partd root}


deflabeltype "ds.label"
```

- lexical entries:

```
deftype "ds.entry" {in: valency("ds.label")
                    out: valency("ds.label")}


defentrytype "ds.entry"
```

Instantiating the ds principles

- re-used from the other dimensions (id, lp):
  - class of models: graph principle
  - deep subcategorization: valency principle
- new:
  - class of models: dag principle

Class of models, deep subcategorization

```
useprinciple "principle.graph" {
  dims {D: ds}}

useprinciple "principle.dag" {
  dims {D: ds}}

useprinciple "principle.valency" {
  dims {D: ds}
  args {In: _.D.entry.in
        Out: _.D.entry.out}}
```

Extending the multi dimension

- add lexical attributes for multi-dimensional principles:

```
defentrytype {%% id/lp multi-dimensional principles
              blocks_lpid: set("id.label")
              %% ds/id multi-dimensional principles
              link2_dsid: map("ds.label" iset("id.label"))
              link2_idds: map("id.label" iset("ds.label"))}
```

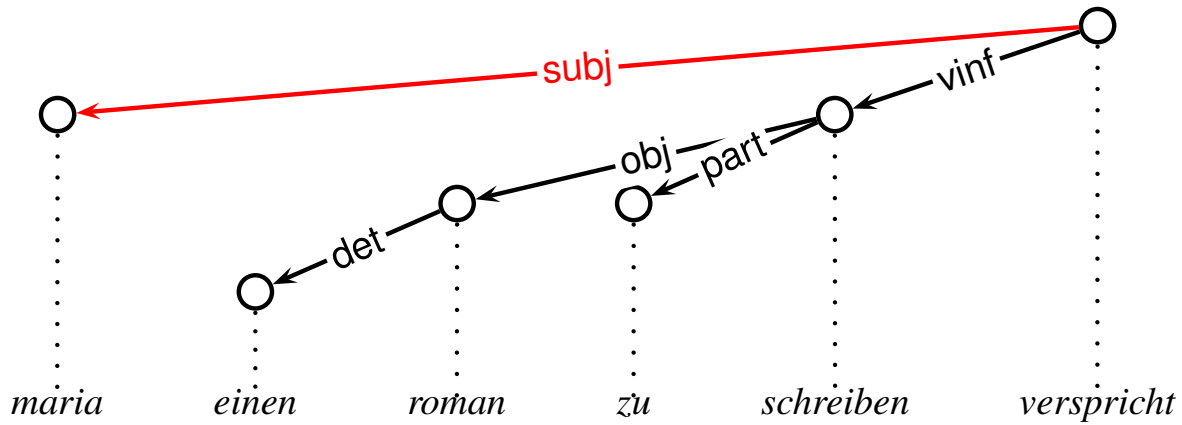
- instantiate multi-dimensional principles:
  - restrict the class of models: climbing principle
  - realize deep by surface arguments: linking principle
  - surface arguments realize deep arguments: linking principle ($^{-1}$)

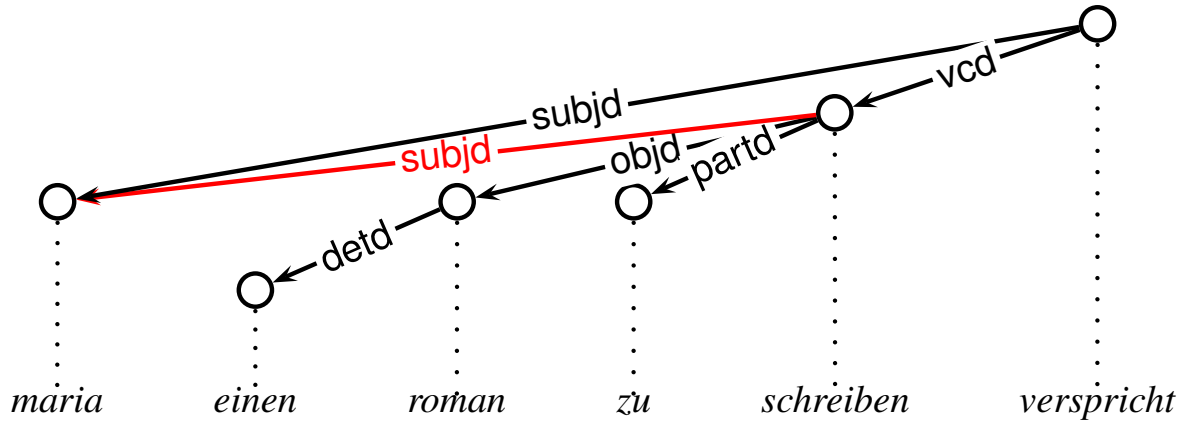
Restricting the class of models

```
useprinciple "principle.climbing" {
  dims {D1: id
        D2: ds}}
```

- parameters:
  - dimensions: `D1`, `D2` (here: `id`, `ds`)
  - deep syntactic arguments can emancipate and be realized by surface syntactic arguments higher up

Realizing deep by surface arguments





Realizing deep by surface arguments contd.

```
useprinciple "principle.linking" {
  dims {D1: ds
        D2: id
        Multi: multi}
  args {Link1: {}
        Link2: _.Multi.entry.link2_dsid}}
```

- linking from ds to id dimension
- declarative semantics (end point):

$$h \xrightarrow{l}_1 d \quad \Rightarrow \quad l'' \in F_2(l) \wedge \xrightarrow{l''}_2 d$$

- declarative semantics (start point and end point):

$$h \xrightarrow{l}_1 d \quad \Rightarrow \quad (F_1(l) \neq \emptyset \Rightarrow l' \in F_1(l) \wedge h \xrightarrow{l'}_2 \ldots \rightarrow_2 d) \wedge (l'' \in F_2(l) \wedge \xrightarrow{l''}_2 d)$$

Surface arguments realize deep arguments

```
useprinciple "principle.linking" {
  dims {D1: id
        D2: ds
        Multi: multi}
  args {Link1: {}
        Link2: _.Multi.entry.link2_idds}}
```

- linking^{-1}
- from id to ds dimension

Lexicon

- lexical classes
  - new lexical classes to specify ds and id/ds properties
  - update existing lexical classes to inherit from them
- lexical entries
  - apply the updated lexical classes

Defining new lexical classes: cnoun_ds

```
defclass "cnoun_ds" {
  dim ds {in: {subjd* objd? root?}
          out: {detd!}}}
```

- *a common noun can be the deep subject of arbitrary many nodes, or deep object of a node, or it can be a root, and it requires a deep determiner*

Defining new lexical classes: intransitive

```
defclass "subjdc" {
  dim ds {out: {subjd!}}}

defclass "intransitive" {
  "subjdc"}
```

- *an intransitive verb requires a deep subject*
- also non-finite intransitives require a deep subject (though not a surface subject)

Defining new lexical classes: subjsubj

```
defclass "subjsubj" {
  dim multi {link2_idds: {subj: {subjd}}}}
```

- *the surface subject must be realized by a deep subject on the ds dimension*
- for subject raising and subject control

Defining new lexical classes: objsubj

```
defclass "objsubj" {
  dim multi {link2_idds: {obj: {subjd}}}}
```

- *the surface object must be realized by a deep subject on the ds dimension*
- for object raising and object control

Defining new lexical classes: vcdLabel

```
defclass "vcdLabel" Label {
 dim id {out: {Label!}}
 dim ds {out: {vcd!}}
 dim multi {link2_dsid: {vcd: {Label}}}}
```

- *require a deep verbal complement (`vcd`) realized by a non-finite verb with id label Label*
- `Label` *is either* `vbse`, `vprt` *or* `vinf`

Applying the updated lexical classes: subjraising

```
defclass "subjraising" {
  "vcdLabel" {Label: vinf}
  "subjsubj"}
```

- *a subject raising verb requires an infinitive and surface subject realizes a deep subject*

Applying the updated lexical classes: subjcontrol

```
defclass "subjcontrol" {
  "subjraising"
  "subjdc"}
```

- *a subject control verb is just like a subject raising verb, and in addition it requires a deep subject for itself*

Applying the updated lexical classes: objcontrol

```
defclass "objcontrol" {
  "vcdLabel" {Label: vinf}
  "objsubj"
  "subjdc"
  "objdc"}
```

- *an object control verb requires an infinitive, its surface object realizes a deep subject, and it requires a deep subject and a deep object*

Applying the updated lexical classes: raising

```
defentry {
  "subjraising"
  "mainverb" {Word1: "scheint"
              Word2: "scheinen"
              Word3: "geschienen"}}
```

Applying the updated lexical classes: control

```
defentry {
  "subjcontrol"
  "mainverb" {Word1: "versucht"
              Word2: "versuchen"
              Word3: "versucht"}}

defentry {
  "objcontrol"
  "mainverb" {Word1: "ueberredet"
              Word2: "ueberreden"
              Word3: "ueberredet"}}
```