

Abduction and Preferences in Linguistics: Extended Abstract

Kathrin Konczak and Ralf Vogel

{Institut für Informatik, Institut für Linguistik}, Universität Potsdam, Postfach 90
03 27, D-14439 Potsdam, {konczak@cs.uni-potsdam.de, rvogel@ling.uni-potsdam.de }

Abstract. We associate optimality theory with abduction and preference handling. We present linguistic problems that appear in the study of dialects as new application of abduction and preference handling. Differences in German dialects originate from different rankings of linguistic constraints which determine the well-formedness of expressions within a language. We introduce a framework for analyzing differences in German dialects by abduction of preferences. More precisely, we will take the perspective of a linguist and reconstruct dialectal variation as abduction problem: Given an observation that a sentence is found as grammatically correct, abduct the underlying constraint ranking. For this, we give a new definition for the determination of optimal candidates for total orders with indifferences. Additionally, we give an encoding for the diagnosis front-end of the DLV system.

1 Background

We assume a basic familiarity with logic programs, answer set programming (ASP) [5], abduction (within ASP), and diagnosis [2, 7, 8, 4].

In this work we want to find preference structures in a linguistic framework as explanations of an abduction problem. A linguistic grammar is a model of the implicit knowledge that guides linguistic behavior. This knowledge is usually conceived as a system of rules and/or well-formedness constraints which determine for a given language which expressions are well-formed and which are not. Our example is the verb order in 3-verb clusters of German dialects. A standard German clause with such a 3-verb cluster looks as follows:

Maria glaubt, dass sie das Lied singen müssen wird.

Maria thinks that she the song sing must will.

Swiss and Standard German follow different ordering strategies for the verbs:

Default verb cluster orders

Standard German: *singen müssen wird*

Swiss German: *wird müssen singen*

Syntactic structures are composed recursively by *complementation*. The object, here: “*ein Lied*” is the complement of its governing head, here: the predicative verb “*singen*”. This verb phrase, “*ein Lied singen*”, is the complement of the modal verb “*müssen*”, and this modal verb phrase, in Standard German: “*ein*

Lied singen müssen”, is the complement of the temporal auxiliary verb, “*werden*”. The differences between the languages and variants can now be described in terms of complement-head order:

Default complement-head orders	
Standard German:	All complements precede their heads: “ <i>ein Lied singen müssen wird</i> ”
Swiss German:	Noun complements precede their heads, verbal complements follow them: “ <i>wird müssen ein Lied singen</i> ”
English	All complements follow their heads: “ <i>will have to sing a song</i> ”

The differences between the three languages can be reconstructed within Optimality Theory (OT) using the following three constraints:

H-Comp A complement follows its head.

Comp-H A complement precedes its head.

H-VComp A verbal complement follows its head.

Constraint rankings are indicated with “ \gg ”, meaning “has higher priority than”. The three rankings that conform to the observations are the following:¹

Standard German: $Comp-H \gg H-Comp (H-VComp)$

Swiss German: $H-VComp \gg Comp-H \gg H-Comp$

English: $H-Comp \gg Comp-H (H-VComp)$

The exact rank of $H-VComp$ can only be determined in Swiss German. While its effects are completely subsumed by the high rank of $H-Comp$ in English, in Standard German all that is necessary is that $Comp-H$ has highest priority, while the relative order of $H-Comp$ and $H-VComp$ is irrelevant because of the low rank of these two constraints. Grammars are usually, but not necessarily, strict total orders of constraints. The rankings given here are only the crucial ones. For those which are left open, any order will be compatible with the observations.

2 Optimal candidates

A linguistic grammar predicts the well-formedness of expressions. Optimality theory grammars do so by establishing a competition between different candidate expressions which are evaluated on a hierarchy of well-formedness constraints. An OT grammar is an input-output mapping. The input defines what is to be expressed. We then have a set of candidate output expressions. The candidates incur different constraint violations. Each candidate is evaluated on the basis of the constraint hierarchy, and the candidate that performs best in this evaluation is the winner, the optimal, hence, grammatical expression.

In the following, we will consider the determination of optimal candidates wrt well-formedness of expressions. For this, let \mathcal{X} be a set of candidates (sentences), \mathcal{C} be a set of constraints, $\delta : \mathcal{X} \times \mathcal{C} \rightarrow \mathcal{N}$ be a violation function, where $\delta(x, c)$ denotes the degree of violation of $x \in \mathcal{X}$ wrt $c \in \mathcal{C}$, and \ll be a (strict) total order on \mathcal{C} . Then, we call $\mathcal{L} = (\mathcal{X}, \mathcal{C}, \delta, \ll)$ a linguistic framework.

¹ These rankings were found out by empirical linguistic studies.

In Section 1, we have taken an example for the dialectal variation in German 3-verb clusters. In the following we will elaborate this example. For the 3-verb cluster we have the following possible word orders:²

Maria glaubt, dass sie das Lied ...

(321) singen müssen wird. (231) müssen singen wird.

(123) wird müssen singen. (132) wird singen müssen.

(312) singen wird müssen. (213) müssen wird singen.

Our set of candidates $\mathcal{X} = \{321, 231, 123, 132, 312, 213\}$ is constituted by these possible word orders.³ $\mathcal{C} = \{H-VComp, Comp-H, H-Comp\}$ is our set of constraints. The degree of violation denotes how well a sentence fulfills a constraint. It is represented by the number of asterisks (*), as in the following table.

	<i>H-VComp</i>	<i>Comp-H</i>	<i>H-Comp</i>
321	***		**
231	**	*	*
123		**	
132	*	*	*
312	**	*	**
213	*	**	*

Next, we want to clarify when a sentence is a best candidate wrt a given constraint ranking [1].

Definition 1. Let $\mathcal{L} = (\mathcal{X}, \mathcal{C}, \delta, \ll)$ be a linguistic framework, where \ll is a strict total order on \mathcal{C} . Then, candidate $x \in \mathcal{X}$ is a winner if there does not exist a candidate $y \in \mathcal{X}, x \neq y$ such that there exists a constraint $c \in \mathcal{C}$ where

1. for all $c' \in \mathcal{C}$ where $c \ll c'$ we have that $\delta(x, c') \geq \delta(y, c')$, and
2. $\delta(x, c) > \delta(y, c)$.

In our example, for the constraint order $H-VComp \gg Comp-H \gg H-Comp$ (Swiss German), we get 123 as winner. For constraint ranking $Comp-H \gg H-VComp \gg H-Comp$ (Standard German), candidate 321 is a winner.

3 Abduction of constraint rankings

In Optimality theoretic terms, linguists observe that a candidate is determined by a speaker as a *winner*, expressing that the sentence is grammatically correct. The problem is that the observer does not know which underlying constraint ranking the speaker has. Here, abduction comes into play. Given a linguistic framework $\mathcal{L} = (\mathcal{X}, \mathcal{C}, \delta)$ with an unknown constraint ranking \ll and an observation that candidate $x \in \mathcal{X}$ wins, we want to abduct \ll which explains x .

The set of candidates is given by rules (1) $cd(x) \leftarrow$ for each $x \in \mathcal{X}$, the set of constraints by (2) $cst(c) \leftarrow$ for each $c \in \mathcal{C}$, and the violation degrees by (3) $viol(x, c, \delta(x, c)) \leftarrow$ where $\delta(x, c)$ is the degree of violation of $x \in \mathcal{X}$ wrt $c \in \mathcal{C}$.

² The numbers signal the hierarchical position of the verb. Verb 1 is the temporal auxiliary (*werden*), verb 2 the modal (*müssen*), and verb 3 the predicative (*singen*).

³ Due to OT we have to consider all possibilities.

According to Definition 1, a winner, can be determined by the following rules:

- (4) $winner(X) \leftarrow cd(X), not\ defeated(X)$
- (5) $defeated(X) \leftarrow cd(X), cd(Y), Y \neq X, better(Y, X)$
- (6) $better(Y, X) \leftarrow cd(X), cd(Y), Y \neq X, cst(C), wins(Y, X, C), not\ hp(X, Y, C)$
- (7) $hp(X, Y, C) \leftarrow cd(X), cd(Y), Y \neq X, cst(C), pref(C_1, C), wins(X, Y, C_1)$
- (8) $wins(X, Y, C) \leftarrow cd(X), cd(Y), cst(C), viol(X, C, NX), viol(Y, C, NY), NX < NY$
- (9) $pref(X, Z) \leftarrow pref(X, Y), pref(Y, Z)$
- (10) $\leftarrow pref(C, C), cst(C)$
- (11) $\leftarrow cst(C_1), cst(C_2), unranked(C_1, C_2), C_1 \neq C_2$
- (12) $unranked(C_1, C_2) \leftarrow not\ pref(C_1, C_2), not\ pref(C_2, C_1), cst(C_1), cst(C_2)$

Our logic program Π consists of the rules (1)–(12).⁴ An observation is that exactly one candidate is observed as a winner, but the other candidates not [1]:

$$O(x) = \left\{ \begin{array}{l} winner(x) \leftarrow \text{for } x \in \mathcal{X} \\ defeat(y) \leftarrow \text{for all } y \in \mathcal{X}, y \neq x \end{array} \right\}$$

Our hypothesis is the set of all possible pairwise preferences: $H = \{pref(c, c') \leftarrow | c, c' \in \mathcal{C}, c \neq c'\}$. Then, an explanation $\Delta \subseteq H$ for $\langle \Pi, H, O \rangle$ gives us a possible strict total order among the constraints such that $\Pi \cup \Delta$ explains O . More precisely, $\Delta \subseteq H$ is an explanation if $O(x) \subseteq S$ for some answer set S of $\Pi \cup \Delta$.

In our linguistic example we have additionally the background knowledge that constraint *Comp-H* is strictly higher preferred than *H-Comp*. Hence, we have $pref(comp, hcomp) \leftarrow$ additionally in Π . Then, our hypotheses is $H = \{pref(comp, hvcomp) \leftarrow, pref(hvcomp, hcomp) \leftarrow, pref(hvcomp, comp) \leftarrow, pref(hcomp, hvcomp) \leftarrow\}$, which gives us together with $pref(comp, hcomp) \leftarrow$ all possible constraint rankings. For computing explanations, we use DLV [3, 4] with the command `-FD` for abductive diagnosis. As a result, we get that 321 has two explanations, $\Delta_1 = \{Comp-H \gg H-Comp \gg H-VComp\}$ and $\Delta_2 = \{Comp-H \gg H-VComp \gg H-Comp\}$. 123 has one explanation, $\Delta = \{H-VComp \gg Comp-H \gg H-Comp\}$, and for observing other candidates as winners, we get no explanation. This means that the constraints are not sufficient for explaining these candidates as a winner. Candidate 321 yields two explanations, $H-Comp \gg H-VComp$ and $H-VComp \gg H-Comp$. This supposes that *H-Comp* and *H-VComp* can be ranked equally. Since Def. 1 is only valid for (strict) total orders, we have to extend it for total (pre-)orders.

Definition 2. Let $\mathcal{L} = (\mathcal{X}, \mathcal{C}, \delta, \preceq)$ be a linguistic framework, where \preceq is a total order on \mathcal{C} . Then, candidate $x \in \mathcal{X}$ is a winner if there exists no $y \neq x$ such that there exists a $c \in \mathcal{C}$ such that

1. for all $c' \neq c$ such that $c' \approx c$ or $c' \succ c$ we have $\delta(c', x) \geq \delta(c', y)$, and
2. $\delta(c, y) < \delta(c, x)$.

The encoding for Definition 1 is adapted to Definition 2 in a straight forward way. Our hypotheses are now the set of all pairwise strict preference relations and the set of all possible indifferences: $H = \{pref(c, c') \leftarrow | c, c' \in \mathcal{C}, c \neq c'\} \cup \{prefeq(c, c') \leftarrow | c, c' \in \mathcal{C}, c \neq c'\}$. From this set of hypotheses, all possible total orders are constructible. Coming back to our example, for candidate 321, we additionally get the supposed explanation that $H-VComp \approx H-Comp$.

⁴ Note that the abducted preference ordering must be total, as required in OT.

4 Discussion and Further Work

We have associated OT with abduction and preference handling. Abduction and preference handling were studied in many other issues, e.g. in [6]. But, as far as the authors know, abduction and preferences were not yet linked to OT before.

We have shown that abduction within ASP is a useful knowledge reasoning tool for linguistic problems, here: dialectic studies, where the abducted explanations match the empirical results found out by the linguists. We have taken the perspective of a linguist and have reconstructed dialectal variation as abduction problem: Given an observation that a sentence is found as grammatically correct, abduct the underlying constraint ranking of the dialect. Furthermore, we have provided an encoding (within ASP) for the diagnosis front-end of DLV.

Regarding linguistic studies, there is an ongoing debate about how unique the rule systems of language are in human cognition, as well as in biology in a very broad sense. The reconstruction of grammatical regularities with abduction and preference handling has consequences for this debate: if grammars can be modeled this way, then they share core properties with other non-linguistic rule systems. This supports a position that does not make special assumptions about the nature of linguistic rule systems.

Further work is to study results when observing non-unique optimal candidates and to study abduction of partial orders instead of total orders, since partial orders may be enough for explaining the observations.

Acknowledgments. The first author was supported by the German Science Foundation (DFG) under grant SCHA 550/6, TP C and by the EC under project IST-2001-37004 WASP. The second author was supported by the DFG under grant FOR-375/2-A3.

References

1. P. Besnard, G. Fanselow, and T. Schaub. Optimality theory as a family of cumulative logics. *Journal of Logic, Language and Information*, 12(2):153–182, April 2003.
2. M. Denecker and A. Kakas. Abduction in logic programming. In A. Kakas and F. Sadri, editors, *Computational Logic. Logic Programming and Beyond*, volume 2407 of *Lecture Notes in Computer Science*, pages 402–436. Springer-Verlag, 2002.
3. DLV. <http://www.dbai.tuwien.ac.at/proj/dlv/>.
4. T. Eiter, W. Faber, N. Leone, and G. Pfeifer. The diagnosis frontend of the DLV system. *AI Communications*, 12(1-2):99–111, 1999.
5. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. pages 1070–1080. The MIT Press, 1988.
6. K. Inoue and C. Sakama. Abducing priorities to derive intended conclusions. *IJCAI-99*, pages 44–49. Morgan Kaufmann Publishers Inc., 1999.
7. A. Kakas, R. Kowalski, and F. Toni. The role of abduction in logic programming. In *Handbook of logic in Artificial Intelligence and Logic Programming*, volume 5, pages 235–324. Oxford University Press, 1998.
8. A. C. Kakas and P. Mancarella. Generalized stable models: A semantics for abduction. *ECAI'90*, pages 385–391, 1990.