Chapter 9

Automatic online writing support for L2 learners of German through output monitoring by a natural-language paraphrase generator

Karin Harbusch and Gerard Kempen

Students who are learning to write in a foreign language, often want feedback on the grammatical quality of the sentences they produce. The usual NLP approach to this problem is based on parsing student-generated text. Here, we propose a generation-based approach aiming at preventing errors ("scaffolding"). In our ICALL system, the student constructs sentences by composing syntactic trees out of lexically anchored "treelets" via a graphical drag & drop user interface. A natural-language generator computes all possible grammatically well-formed sentences entailed by the student-composed tree. It provides positive feedback if the student-composed tree belongs to the well-formed set, and negative feedback otherwise. If so requested by the student, it can substantiate the positive or negative feedback based on a comparison between the student-composed tree and its own trees (informative feedback on demand). In case of negative feedback, the system refuses to build the structure attempted by the student. Frequently occurring errors are handled in terms of "malrules." The system we describe is a prototype (implemented in JAVA and C++) which can be parameterized with respect to L1 and L2, the size of the lexicon, and the level of detail of the visually presented grammatical structures.

Motivation and preview

Many foreign-language learners, especially students at the level of secondary or tertiary education who are learning to write in the target language, want feedback on the grammatical quality of the sentences they produce. This raises the question how *ICALL* systems (*Intelligent Computer-Assisted Language Learning*) can provide feedback on the grammatical structure of their L2 sentences—for instance, in essay writing exercises. The usual *NLP* (*Natural Language*

Processing) approach to this problem is based on *parsing*. After the student has typed a sentence, the parser evaluates it and provides feedback on the grammatical quality. However, the more errors a sentence contains, the less accurate the feedback tends to be: A parser working with a large lexicon and a rich grammar usually finds many correction options but has no criteria to select the option that fits the message the student wishes to express. A related problem is caused by ambiguity. Hardly any sentence can be parsed unambiguously (cf. the proverbial *Time flies like an arrow,* for which Wikipedia lists no less than seven different interpretations). Hence, it is notoriously difficult to produce highly reliable feedback based on the parsing results.

We propose a *generation*-based approach aiming at the *prevention* of errors ("*scaffolding*"). Students construct sentences incrementally, and the ICALL system intervenes immediately when they try to build an ill-formed structure. We use a natural-language sentence and paraphrase generator—briefly called *paraphraser*—with a graphical drag & drop user interface. In our system, the student drags words into a workspace where their grammatical properties are displayed in the form of syntactic "treelets" as defined in the lexicalized *Performance Grammar* formalism (PG; Kempen & Harbusch, 2002, 2003; Harbusch & Kempen, 2002). The treelet(s) associated with a word express(es) conditions on the syntactic environment(s) in which the word can occur (subcategorization restrictions). In the workspace, the student can combine treelets by moving the root of one treelet to a foot (i.e., a non-lexical leaf) of another treelet. In the generator, this triggers a unification process that evaluates the quality of the intended structure. If the latter is licensed by the generator's syntax, the tree grows and a larger tree is displayed. In case of licensing failure, the generator informs the student about the reason(s). This feedback follows directly from the unification requirements. The level of detail of the feedback can be parameterized with respect to the assumed proficiency level of the student. At any point in time, the stu-

dent can issue a request for grammatical information—not only about syntactic rules, but also about the structure under assembly: *informative feedback on demand.*

The system presented here monitors the process of combining words and word groups into clauses and sentences (including coordinate and subordinate structures). The current prototype focuses on constituent order in German as L2 and checks correctness of attempted orderings. Feedback is based on the correctly applied L2 ordering rules. The paraphrase generator can provide the student with the correct ordering(s) on demand. Additionally, typical errors due to intrusions from L1 (currently English) are handled by *malrules*.

The chapter is organized as follows. First, we outline the state of the art in ICALL systems for essay writing based on NLP techniques. In subsequent sections, we sketch the Performance Grammar (PG) formalism, illustrate how it represents contrasts between well-formed and ill-formed structures, and describe the prototype of our generation-based L2-learning system called COMPASS-II:[1] the generator that monitors the sentence construction process, the user interface, and various types of feedback. In the final section, we take stock and discuss desiderata for future work.

ICALL writing tools: state of the art

Computer-supported learning of how to write grammatically correctly in L1 and L2 figures prominently in the ICALL literature. Here, we cursorily review systems based on *natural-language processing (NLP)* techniques that provide students with online support in writing novel sentences that are grammatically well-formed.

[1] COMPASS-II (see also Harbusch, Kempen & Vosse, 2008) is an acronym for COMbinatorial and Paraphrastic Assembly of Sentence Structure, version II. It is an improved version, implemented in JAVA and C++, of the COMPASS system described by Harbusch, Kempen, van Breugel & Koch (2006).

Virtually the entire literature on NLP applications to the syntactic aspects of first- and second-language teaching is based on parsing technology (Heift & Schulze, 2003). A *parser* computes the syntactic structure of input sentences, possibly in combination with their semantic content (provided that all words in the sentence are in the vocabulary, that the grammar available to the system covers all constructions mastered by the student, and that the input does not contain any errors). However, as indicated above, these systems struggle with ungrammatical input and need special measures preventing the parsing quality from getting unacceptably poor. For example, in the *FreeText* system (L'haire & Vandeventer Faltin, 2003), the syntactic–semantic analysis is supplemented with *constraint relaxation* and *sentence comparison.* Other systems invoke matches with corpus texts (Granger, 2004). Yet another option is the addition of malrules to cover frequent errors (Fortmann & Forst, 2004).

Probably the first *generator*-based[2] software tool capable of evaluating the grammatical quality of student output was developed by Zamorano Mansilla (2004), who applied a sentence generator (KPML; Bateman, 1997) to the recognition and diagnosis of writing errors ("fill-in-the-blank" exercises). Zock & Quint (2004) converted an electronic dictionary into a drill tutor. Exercises were produced by a goal-driven, template-based sentence generator, with Japanese as the target language. More recently, Harbusch, Itsova, Koch & Kühner (2008, 2009) developed the "Sentence Fairy"—an interactive tutoring system for German-speaking elemen-

[2] A *generator* produces a sentence or a set of paraphrases from an abstract representation of the content, often called *logical form* (see Reiter & Dale, 2000, for an authoritative overview of sentence and text generation technology). In the case of *paraphrase generation*, the generator delivers all possible ways of linguistically realizing the input logical form, given the lexicon and the grammar rules. Virtually all recent natural language generation systems work in a *best-first* manner, i.e., produce only one output sentence rather than the set of all paraphrases. As it is not easy to change the control structure of such a system, the choice of generators is very limited. The paraphrase generator deployed in COMPASS-II does not take logical forms as input but a set of "lexical treelets" as defined in PG, which are connected via dependency links. It delivers all possible sentences licensed by the grammar (see next section).

tary schoolers who are about 10 years old, which supports writing little stories in L1. The pupils perform limited tasks such as combining simple clauses into compound or complex sentences. A sentence generator (described in Harbusch, Kempen, van Breugel & Koch, 2006; see also next section) calculates all correct paraphrases, and an avatar (the Sentence Fairy) provides feedback.

Both the Sentence Fairy and the COMPASS-II system presuppose a minimum level of explicit grammatical knowledge in the student. Without it, the feedback information provided by the systems would be incomprehensible. Hence, systems of this type—but also parsing-based systems that are able to elucidate the parse trees they deliver—can only be used in the context of courses where the necessary grammatical concepts, structures and rules have been, or are being, explained. Although this requirement entails a restriction on the range of potential users, in view of the increasing *grammatical awareness* in present-day language instruction (cf. Levy, 1997; Roehr, 2007), we believe this drawback is a minor one.

Performance Grammar

The Performance Grammar (PG) formalism distinguishes three aspects of the structure of sentences: *dependency* relations, *constituent* structure, and *linear* order. The dependency relations and the constituent structure together form the *hierarchical* (or *dominance*) structure. The dependency relations include functional relations (subject, direct and indirect object, head, complement, determiner, modifier, etc.). The constituent structure comprises word categories (parts of speech) and word groups (the various types of phrases and clauses). As (a subset of) these concepts and structures are taught in many grammar courses, PG structures are relatively easy to apprehend—easier than the structures defined in many other formalisms. This advantage is enhanced by the fact that PG does not make use of movement transformations. Where certain other formalisms invoke such transformations, PG uses word order rules that assign constituents to

their final positions in one go. PG's hierarchical structures can be visualized as rather flat unordered trees. The application of linear order rules may give rise to structures that can be depicted as ordered trees with crossing branches (graphs). Taken together, given the fact that PG's theoretical apparatus is rather close to what the students learn in pedagogical grammars, and that the structures it generates can be visualized in a transparent manner, we believe that PG is attractive as an ICALL formalism.

We now turn to some key technical aspects. PG's key operation is *Typed Feature Unification*—widely used in theoretical and computational linguistics (e.g. in HPSG; Sag, Wasow & Bender, 2003). Moreover, PG is *lexicalized*, i.e. every constituency rule is associated with a *lexical anchor* consisting of at least one word (form).

Figure 9.1 (a) illustrates an *elementary treelet* (also called *lexical frame)* for the German word form *Junge* 'boy'. The rightmost branch specifies the lexical anchor of the treelet: *Junge* is a n[oun] functioning as the h[ea]d of a N[oun]P[hrase]. The second layer of nodes represents grammatical functions: det[erminer], q[uantifier], mod[ifier], etc. The third layer consists of phrasal nodes that specify which types of constituents are allowed to fulfill the function above them (the slash '/' separates alternative options). For example, the modifier role can be played by a P[repositional]P[hrase], an A[djectival]P[hrase], or a S[entence] (more precisely, a relative clause). One node in the third layer specifies the word category of the head, i.e. the lexical anchor (here n[oun]).

Every node of a lexical treelet has associated with it a set of morphosyntactic features. They are specified in the lexicon of word forms[3]. A feature is a combination of a property and a value specification. The latter may be a single term (which holds for the features of the noun *Junge*, with the feature-value pairs: wordform=Junge, lemma=Junge, gender=masculine, person=3rd, case=nominative, and number=singular) but it may also be a disjunctive set of alternative value options. For instance, the word form *Jungen* (for 'boy' or 'boys') has the same treelet associated with it, except for the leaf node *Jungen*. However, the feature structure for the noun *Jungen* expresses the fact that *Jungen* can have *genitive* or *dative* or *accusative case* if and only if its *number* is *singular* whereas it can have *nominative, genitive, dative* or *accusative case* if and only if its *number* is *plural.* In disjunctive feature structures, the alternative value options are enumerated within curly brackets (the logical inclusive OR), and square brackets enclose an AND enumeration. The feature specification for the word form *Jungen* at node n[oun] now looks as follows:

[wordform=Jungen AND
lemma=Junge AND
gender=masculine AND
person=3rd AND
{[case={gen OR dat OR acc} AND number=singular] OR
[case={nom OR gen OR dat OR acc} AND number=plural]}]

[3] Word forms are members of an inflectional paradigm. For instance, *Junge* and *Jungen* both belong to the same paradigm: the paradigm of the "lemma" *Junge*. Lemmas are referred to by one member of the paradigm—here the wordform *Junge*.

Phrasal leaf nodes (*foot nodes*) can be expanded by an appropriate treelet whose root node carries the same label, thus forming more complex phrases. This operation (technically called *unification*) merges a foot node of one treelet with the root node of another treelet. In Figure 9.1 (b), the D[eterminer]P[hrase] foot node has been expanded by the DP root node dominating the appropriate masculine definite article *der*, and the ADJ[ective]P[hrase] root node dominating the word form *kleine* 'small' expands the foot node of a mod[ifier][4] branch. Whether a root and a foot node can be merged ("unified") or not, depends not only on their label but also on the associated features. The feature specifications are used by the unification operation to select legal expansions. For instance, the fact that S-type modifiers within NPs should be relative clauses (rather than, say, main clauses) is controlled by features. Similarly, other features control the selection of the inflected word form *kleine* instead of the uninflected *klein*. For details of the unification process, in particular on how it deals with phenomena of grammatical agreement, we refer to the papers quoted above.

Associated with every treelet is a *topology*. Topologies serve to assign a linear order to the branches of lexical frames. Here, we only illustrate the topologies associated with lexical frames for verbs ("clausal treelets"). A topology is a left-to-right sequence of slots which can be occupied by one or more constituents. In the current PG grammar for German, clausal topologies comprise nine slots, grouped into three "fields": one slot in the Forefield (slot F1), six slots in the Midfield (slots M1 through M6), and two Endfield slots (E1 and E2) The terminology derives from the *Topologische Felder* in German structural linguistics. Every grammatical function

[4] Except for modifiers, every grammatical function in an elementary treelet occurs there at most once. Some of them are *obligatory*, like subjects of finite verbs and direct objects of transitive verbs, whereas others are *optional* (e.g., many indirect objects). To allow more than one modifier, when a branch of this type is expanded by a unification partner, another exemplar is added immediately.

(subject, head, direct object, complement, etc.) has a small number of placement options (slots) in the topology associated with its "own" clause, i.e. within the verb's lexical frame. Here are some of the slot fillers:[5]

F1: Subject, topic or focus in a declarative main clause (one constituent only); a wh-constituent (a phrase including an interrogative pronoun) in an interrogative main clause; a wh-constituent in a complement clause

M1: Finite verb in a main clause; the complementizer *dass* 'that' of a complement clause

M2-M5: Non-wh subject, direct object, indirect object, non-finite complement clause

M6: Finite verb, possibly preceded by particle and pre-infinitival *zu* 'to', in a subordinate clause

E1-E2: Nonfinite complement preceding finite complement

Example sentence (1) shows PG's linear order system at work. The hierarchical structure is depicted in Figure 9.1 (c). The root S-node of the verb treelet associated with the word form *sage* 'say' in the complement clause has been unified with the complement (cmp) S-node of the word form *will* 'wants' of the verb in the main clause.

(1) Was will der kleine Junge dass ich sage?
what wants the little boy that I say
'What does the little boy want me to say?'

Each of the verbs instantiates its own topology**.** Constituents fulfilling a "major" grammatical function (i.e., a function immediately dominated by an S-node) receive a position in accordance with the above slot assignment rules (cf. Figure 9.1 (d)). In the main clause, the subject (which is neither a focused nor a wh-constituent) goes to M2; the verb is assigned M1,

[5] The description in this paper conflates the individual slot positions M2–M5 and E1–E2, respectively. The more differentiated PG rules allow simple but finegrained word order specifications. For instance, an indirect object in the form of a personal pronoun is allowed to precede a full (i.e. non-personal-pronoun) subject NP.

(which is neither a focused nor a wh-constituent) goes to M2; the verb is assigned M1, and the entire complement clause ends up in E1–E2. At the subordinate clause level, the direct object—a wh-constituent—goes to F1, the subordinating conjunction *dass* 'that' goes to M1, the subject to M2, and the verb to M6, as prescribed by the rule for subordinate clauses.

How is the direct object NP *was* 'what' "extracted" from the complement clause and "promoted" into the main clause? "Movement" of phrases between clauses is due to *lateral topology sharing*. If a sentence contains more than one verb, each lexical frame instantiates its own topology. In certain syntactic configurations (not to be defined here; but see Harbusch & Kempen, 2002), the topologies of two verbs are allowed to *share* one or more identically labeled lateral (i.e. left- and/or right-peripheral) slots. Sentence (1) embodies such a configuration. *After two slots have been shared, they are no longer distinct; in fact, they are unified and become token-identical.* In (1), the embedded topology shares its F1 slot with the F1 slot of the matrix clause. This is indicated by the dashed borders of the lower F1 slot of Figure 9.1 (d). Sharing the F1 slots effectively causes the embedded direct object *was* to be *preposed* into the main clause (black dot in F1 above the single arrow in Figure 9.1 (d)). The dot in slot E1–E2 in the main clause topology above the double arrow marks the position selected by the remainders of the finite complement clause.

Figure 9.1 (e) shows the linearly ordered structure after slot assignment. It also includes details concerning the linear order assignment to nodes within nonclausal constituents. For, not only clauses but in fact all constituents have—usually very simple—topologies associated with them. For instance, NP topologies have five slots, labeled NP1 through NP5, for determiner, quantifier, prenominal modifier, head, and postnominal modifier, respectively. The line connecting the S-node below "E2:cmp" and the node labeled "F1:dobj" represents the promotion of the

wh-constituent *was* 'what' from the subordinate clause into the main clause (see also the F1 slots in Figure 9.1 (d)): The promoted element fulfills a function in the subordinate clause but surfaces in the main clause.

"Scaffolded" sentence construction based on natural-language generation

In this section, we describe how COMPASS-II lets students compose sentences in PG format while the generator is monitoring this process and provides online feedback. This is followed by a sketch of the user interface and its parameterization options.

Student actions and feedback by the system

The student drags word forms one-by-one from an online lexicon into a *workspace*. The dragging actions are continually monitored by the generator. Each time a word form is entered into the workspace, the system reacts by depicting the lexical treelet associated with that word. As soon as the workspace is populated by more than one word, the student can combine them by dragging the root of one treelet over one foot of another treelet.[6] The system then checks whether root and foot node can be unified, and if so, pretty-prints the resulting larger tree (hierarchical structure) in the workspace. Furthermore, it provides a *positive feedback* message. If unification fails, *negative feedback* is provided (see next subsection). By pressing a button at the bottom of the workspace, the student can undo any action even after the system has accepted them (unrestricted undo). No constraints are imposed on the order in which the student performs

[6] The mouse handling need not be very precise. The root of a tree(let) gets selected by a mouse click anywhere within the tree(let). In order to connect the root node of the currently selected tree to a foot node of another tree(let), the student only needs to drag the former tree toward the targeted foot node of the latter. The nearest foot node calculated by the system is highlighted. Releasing the mouse triggers a unification attempt for the root and foot nodes involved. If the student made a mistake and initiates an undo action, the system returns to the previous state of the workspace.

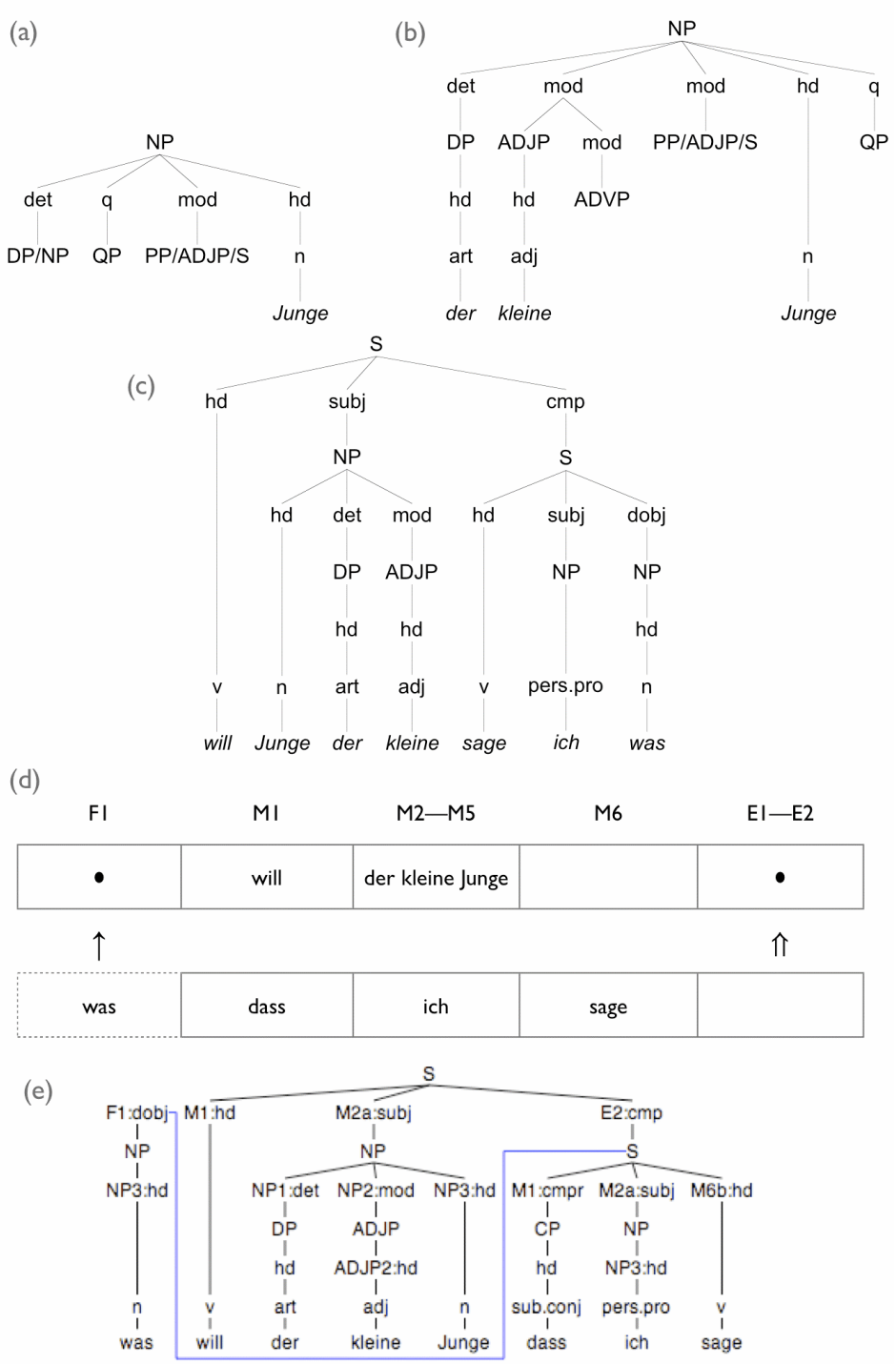

Figure 9.1. (a) Elementary treelet for the noun *Junge*. (b) *Junge* treelet unified with a determiner and an adjective. (c) Hierarchical structure of example (1), with arbitrary word order. (d) Topology slot assignments of the major constituents of main and subordinate clause of sentence (1). (e) Linearly ordered tree spelling out the final topological slot positions of the major constituents of sentence (1).

the actions. For instance, all noun phrases can be built prior to selecting a verb; and all NPs can be assigned a grammatical function without spelling out their linear order. Clauses can be combined into more complex sentences by linking them via coordinating or subordinating conjunctions. We call this way of composing sentences "*scaffolded writing*" as it prevents the students from constructing wrong sentences. At any point in time during the sentence composition process, the student can *query* the system by clicking on any node of a tree(let) in the workspace. In response, the system provides *informative feedback* by displaying the morphosyntactic features of that node (or a subset thereof).

The student actions described so far lead to the construction of *hierarchical* structures for partial or complete sentences. In order to specify a possible linear order for the branches of a hierarchical structure, the student can drag nodes (and the subtrees they dominate) to a position left or right of one of its siblings. When the node is released, the workspace is updated and the system pretty-prints the branches in the new left-to-right order. Because several drag & drop actions may be needed before the student is satisfied with the tentative linear order of constituents, the systems checks well-formedness of the current order only when explicitly requested to do so. When during a linear order check the generator notices that an obligatory constituent is missing (e.g., the subject of a finite verb or the direct object of a transitive verb), the system asks the student to expand the obligatory node before word order checking takes place. This is necessary because the generator needs the focus and wh-features of that constituent in order to determine its slot position.

Importantly, the positive or negative feedback supplied by the system in response to composition actions is not just a "correct" or "incorrect" signal. Positive feedback is accompanied by a summary of the linguistic action just performed, and its effect. Negative feedback in-

cludes a statement of the reason(s) why the unification or ordering attempt failed. Notice that the content of such feedback is conceived by the generator itself, in response to concrete unification or ordering attempts by the student.

The user interface and its parameterization options

When starting up, COMPASS-II initializes four windows: to the left a window where the lexicon is displayed; to the right a window for feedback messages; in the upper central region of the screen a window for linear order manipulations on word strings; and in the lower central region a large window serving as workspace. Special push buttons at the bottom of the workspace enable the following system actions: get word order in selected tree, erase optional branches in selected tree, delete selected tree, and undo last tree manipulation, respectively. The upper central region includes a button labeled "Check word order."

All windows allow manipulation by the student, except for the right-hand window which is reserved for system feedback. The student can select word forms from the left-hand window. The upper central window can display the terminal leaves of a tree selected in the workspace; to this purpose, the student can push the button labeled "Get word order in selected tree" (see next section).[7]

The user interface of COMPASS-II can be *parameterized* in the following respects:

1. Size of the lexicon
2. Level of detail concerning the visible hierarchical structure and associated features
3. Level of detail concerning the feedback
4. L1-specific malrules.

[7]For a guided tour through the system, see http://www.uni-koblenz.de/~harbusch/COMPASSII-guided-tour.html.

Ad (1) The size of the lexicon can be tailored to a specific task, i.e. to the limited vocabulary addressed in a lesson. However, the full range of CELEX word forms (Baayen, Piepenbrock & Gulikers, 1995) is available to the paraphraser; hence, in another parameterization, advanced students can freely formulate and check the sentences they want to write in L2. New lemmas and their word forms can be added by hand.

Ad (2) The student drags words into the workspace in order to build a phrase or a sentence. This action is monitored online by the paraphraser. How many grammatical details known to the paraphraser become visible to the student, is a matter of parameterization. Showing treelets and feature structures in PG notation is the default parameterization. The grammatical terminology used in the feedback messages can be tailored to the vocabulary the learner is familiar with (e.g., L1 terms for elementary school children vs. international terms for advanced learners). However, the system requires a lower bound on the level of visualized grammatical detail. Selected word forms are often ambiguous, i.e. have several readings, of which the student might not be aware. By displaying all alternative readings, the system forces the student to select the reading to be used in the construction process. Such confrontations with syntactic facts can serve to raise the student's grammatical awareness.

Ad (3) The level of detail of the feedback messages is determined as follows. As outlined earlier, every system action is associated with a feedback message. Actually, this message has the form of a template with placeholders in PG terminology. The placeholders get automatically instantiated as terms in the student's grammatical vocabulary. Moreover, the set of templates can be adapted to teacher preferences (e.g., as was done in the Sentence Fairy system; see Harbusch et al., 2008/2009).

How often syntactic nodes are queried, is completely in the student's hands. Simply

moving the mouse over a node of a tree in the workspace triggers the presentation of the morphosyntactic features of that node. Thus, students can verify their guesses as regards the features of the selected word form, or simply learn which features characterize a word form they have not used before. In the example of NP *den kleinen Jungen*, they might be insecure about its number and case features (as mentioned earlier, *Jungen* can be singular or plural). By checking these features, they can predict whether a desired unification action will work properly (e.g., moving the treelet to the direct or indirect object NP foot node of a verb will yield a successful unification whereas the subject option causes unification failure). At any point in time, the student can query other nodes or resume the sentence construction task.

While performing the sentence construction task, every composition action is commented in terms of positive or negative feedback as indicated at the end of the previous section. When the student attempts to merge the root node of one treelet with a foot node of another treelet, this triggers a unification process that evaluates the well-formedness of the intended structure. If the unification is licensed by the paraphraser's syntax, the feedback window flashes in green and displays a text saying that the node labels and the feature structures match (the student needs not pay attention to the text; the green color is a signal to go on).

In case of unification failure, the background of the feedback window turns red, informing the students that the intended unification is not executed, and inviting them to read the explanatory text. In addition to the reason(s) of unification failure, this text may provide small hints on how to continue, e.g. a list of other word forms belonging to the same inflectional paradigm as the offending word.

Ad (4) The paraphraser can run *malrules* that derive from typical errors users make in L2, given their L1. For instance, the erroneous string *der kleiner Junge* is "accepted" by the system

but triggers a negative feedback message (the correct string is *der kleine Junge*; the confusion may arise from the correct *ein kleiner Junge* 'a little boy'). In COMPASS-II, the malrules can be parameterized for different L1s.

Constructing a sentence in COMPASS-II

The following example illustrates how students can construct sentences in their personally preferred manner. Let us assume a student wants to compose sentence (2).

(2) *Heute baut Anja eine Rakete weil ihr Freund morgen zum Mond fliegen will*
Today builds Anja a rocket because her friend tomorrow to-the moon fly wants-to
'Today Anja builds a rocket because tomorrow her friend wants to fly to the moon'

She is allowed to perform the various subtasks in any order. For instance, main and subordinate clauses may be constructed, inclusive of their internal linear order, before they get combined. Alternatively, she may first concentrate on the overall structure of the sentence as a whole. Anyway, only during the final steps can she determine the ultimately correct word order based on her (implicit or explicit) knowledge of the L2 linear order rules.

Suppose the student moves the determiner *einen*$_{\text{ACC,MASC}}$ 'a' to the foot node of the det[erminer] branch of the feminine noun *Rakete* 'rocket'. The feedback window now turns red and requires the student to pay attention to the gender mismatch. The two trees snap back to their original positions and the system refuses to perform the erroneous action. The student may now inspect the features in detail (by querying some nodes) in order to pick up ideas for further actions. Suppose the student now decides to erase the treelet for *einen* by pushing the "Erase selected tree" button and to select the word form *eine* (in response to a hint in the feedback window). Moving the determiner *eine* 'a' to the foot of the det[erminer] branch of the *Rakete* 'rocket' treelet elicits positive feedback. The feedback window turns green and the two treelets

merge to form one overall dominance structure. The feedback text summarizes the individual steps taken by the paraphraser on its way to the tree structure displayed in the workspace.[8]

The remainder of the example illustrates how our student can determine word order. Orderings of lexical leaves of any hierarchical structure can be changed by dragging subtrees horizontally and releasing them to the left or right of sister subtrees, whereupon the resulting tree is pretty-printed. The paraphraser does not immediately check whether the resulting linear order of words (lexical leaves) is grammatically well-formed; instead, it waits until the student issues an "order check". This check is executed in two steps. A press on the "Get word order in selected tree" button below the workspace causes the system to copy the lexical leaves of a selected tree into the word order window above the workspace. Then, when the button "Check word order" is pressed, the paraphraser determines whether the copied leaf string (i.e., the word group or sentence) is in the list of well-formed linear orders, and provides feedback accordingly. Additionally, or alternatively, the student can reorder the words of the sentence in a cut & paste manner, followed by an order checking request. In case of positive feedback, the system shows the hierarchical structure with topology slot assignments (as illustrated in Figure 9.1 (e)). In case of negative feedback (e.g., caused by the ill-formed string **heute Anja baut eine Rakete*.), the system shows a list of correct orderings in the feedback panel. This list can be queried by selecting one of the alternative orders, whereupon the system shows the corresponding tree with slot assignments (informative feedback).

In order to demonstrate the effect of *malrules*, we show how COMPASS-II reacts to two

---

[8] Furthermore, the PG grammar specifies which branches are optional or obligatory in terms of a feature on grammatical function nodes. The student can remove any optional branch by selecting a tree, and then pressing the button labeled "Remove optional branches in selected tree" at the bottom of the workspace.

typical word order errors by L2 learners of German whose L1 is English. In German subordinate clauses, the finite verb goes to a clause-final position whereas in main clauses it is "verb-second". Let us assume that our student has violated both rules and has already produced sentence (3).

(3) **Heute Anja baut eine Rakete weil morgen ihr Freund will zum Mond fliegen*

A malrule is a special grammar rule that "allows" the paraphraser to build ungrammatical structures but simultaneously triggers an error message. When analyzing the main clause of (3), where the finite verb is verb-third rather than verb-second, the paraphraser "accepts" the sub-string *heute Anja baut* but immediately provides negative feedback and prints the content of the malrule. Another malrule reacts to the incorrect verb-second position of the finite verb *will* 'wants' in the subordinate clause introduced by the subordinating conjunction *weil* 'because'; here, the finite verb should occupy a position at the end of the midfield (cf. the above topology rules). Additionally, the system lists all correct orderings in the default feedback mode. They can be queried as yet another piece of informative feedback.

Discussion

We view the current version of the COMPASS-II as the prototype of an "engine" that can drive the automatic evaluation and diagnosis of sentences produced by L2 students of German. The system is far from complete and not yet usable in the classroom. Several software aspects are in need of improvement, in particular the robustness of the system and the way feedback information is couched in nontechnical terms. We hope, however, that the foregoing description rouses the interest of the (I)CALL community in the great potential of generator-based systems as providers of online L2 writing support to students whose knowledge and understanding of

sentence grammar is at high-school or beginning-university level.

Profitable deployment a COMPASS-II-type tool in the classroom requires embedding it in a tutoring system tailored to the requirements imposed by specific student populations and by specific L2 courses and exercise types. The resulting system should be evaluated with real students under realistic conditions. Of particular interest will be empirical studies that pit a generator-based writing support tool like COMPASS-II against a parser-based or a traditional (template-based) tool.

In the absence of pertinent supporting empirical data, we speculate that an important asset of systems like COMPASS-II is the fact that they do not impose upon the student any specific learning strategy (exploration, trial and error, drill and practice). Via the embedding tutoring system, they can be adapted to the strategy preferred by student or teacher. Another advantage —emphasized in the above sections—is the prospect of enabling effective feedback: feedback that, in line with the notion of scaffolding, is immediate, reactive and assistive.

We are keenly aware that COMPASS-II makes heavier demands on the student's explicit grammatical knowledge than many another writing support tool. However, in quite a few languages, rules for spelling and other aspects of writing presuppose that the writer is able to explicitly recognize detailed syntactic properties of the sentence under construction. Well-known examples are morphosyntactic distinctions that got lost in pronunciation but are maintained in spelling—e.g., the distinction between *dass* (subordinating conjunction) and *das* (determiner or pronoun) in German, and numerous inflectional suffixes in French and Dutch. We suggest that COMPASS-II-type tools can be employed fruitfully in integrated courses writing and grammar courses.

A particularly useful approach to teaching grammar and writing in an integrated fash-

ion—one that is relatively easy to implement in COMPASS-II—is to focus on an interrelated set of syntactic constructions and the rules controlling their shape. An example concerns coordinate structures and their elliptical forms: *forward conjunction reduction, gapping, right node raising*, etc. Recently, we have laid the PG-oriented linguistic and computational groundwork for these constructions, which have very high usage frequencies (Kempen, 2009; Harbusch & Kempen, 2006, 2007). One of the topics we might address in the near future is to build a COMPASS-II application based on this groundwork.

Acknowledgement

We are indebted to Theo Vosse for his collaboration in general, and in particular for making available his C++ software for word order checking.

References

Baayen, R.H., Piepenbrock, R., & Gulikers, L. (1995). *The CELEX lexical database (release 2.5, CD-ROM)*. Linguistic Data Consortium: University of Pennsylvania, PA.

Bateman, J.A. (1997). Enabling technology for multilingual natural language generation. *Natural Language Engineering, 3,* 5–55.

Delmonte, R., Delcloque, P., & Tonelli, S. (Eds.) (2004). *Proceedings. of the InSTIL/ICALL2004 Symposium,* Venice (Italy). Retrieved from http://www.isca-speech.org/archive/.

Granger, S. (2004). Computer learner corpus research: Current status and future prospects. In Connor, U., & Upton, T. (Eds.). *Applied Corpus Linguistics: A Multidimensional Perspective* (pp. 123–145). Amsterdam: Rodopi.

Fortmann, C., & Forst, M. (2004). An LFG Grammar Checker for CALL. In Delmonte et al*.,* 2004 (pp. 59–61).

Harbusch, K., Itsova, G., Koch, U., & Kühner, C. (2008). The Sentence Fairy: A natural-language generation system to support children's essay writing. *Computer Assisted Language Learning, 21,* 339–352.

Harbusch, K., Itsova, G., Koch, U., & Kühner, C. (2009). Computing accurate grammatical feedback in a virtual writing conference for German-speaking elementary-school children: An approach based on natural language generation. *CALICO Journal, 20,* 626–643.

Harbusch, K., & Kempen, G. (2002). A quantitative model of word order and movement in English, Dutch and German complement constructions. In Tseng, S.-C. (Ed.), *Proceedings of the 19th COLING*, Taipei, ROC (pp. 328–334). San Francisco, CA: Morgan Kaufmann. Retrieved from http://www.aclweb.org/anthology/C/C02.

Harbusch, K., & Kempen, G. (2006). ELLEIPO: A module that computes coordinative ellipsis for language generators that don't. In *Proceedings of the 11th EACL*, Trento, Italy, (pp. 115–118). East Stroudsburg, PA: ACL. Retrieved from http://www.aclweb.org/anthology/E/E06/#2000.

Harbusch, K., & Kempen, G. (2007). Clausal coordinate ellipsis in German. In Nivre, J., Kaalep, H.-J., Muischnek, K., & Koit, M. (Eds.), *Proceedings of the 16th NODALIDA*, Tartu, Estonia (pp. 81–88). Retrieved from http://dspace.utlib.ee/dspace/handle/10062/2683.

Harbusch, K., Kempen, G., van Breugel, C., & Koch, U. (2006). A generation-oriented workbench for Performance Grammar. In Colineau, N., Paris, C., Wan, P., & Dale, R. (Eds.), *Proceedings of the 4th INGL,* Sydney, Australia (pp. 9–11). Retrieved from http://www.aclweb.org/anthology/W/W06/W06-14.pdf.

Harbusch, K., Kempen, G., & Vosse, T. (2008). A natural-language paraphrase generator for on-line monitoring and commenting incremental sentence construction by L2 learners of

German. In Koyama, T., Noguchi, J., Yoshinari, Y., & Iwasaki, A. (Eds.), *Proceedings of WORLDCALL,* Fukuoka, Japan (pp. 190–193). Retrieved from http://www.j-let.org/~wcf/proceedings/proceedings.pdf.

Heift, T., & Schulze, M. (Eds.) (2003). Error diagnosis and error correction in CALL. *CALICO Journal, 20(3)*.

Kempen, G. (2009). Clausal coordination and coordinate ellipsis in a model of the speaker. *Linguistics, 47,* 653–696.

Kempen, G., & Harbusch, K. (2002). Performance Grammar: A declarative definition. In Theune, M., Nijholt, A., & Hondorp, H. (Eds.), *Computational Linguistics in the Netherlands 2001* (pp. 146–162). Amsterdam: Rodopi.

Kempen, G., & Harbusch, K. (2003). Dutch and German verb constructions in Performance Grammar. In Seuren, P.A.M., & Kempen, G. (Eds.), *Verb Constructions in German and Dutch* (pp. 185–222). Amsterdam: Benjamins.

Levy, M. (1997). *CALL: context and conceptualization.* Oxford: Oxford University Press.

L'haire, S., & Vandeventer Faltin, M. (2003). Error diagnosis in the FreeText project. *CALICO Journal, 20*(3), 481–496.

Reiter, E., & Dale, R. (2000). *Building applied natural language generation systems.* New York: Cambridge University Press.

Roehr, K. (2007). Metalinguistic knowledge and language ability in university-level L2 learners. *Applied Linguistics, 29,* 173–199.

Sag, I.A, Wasow, T., & Bender, E. (2003). *Syntactic Theory: A Formal Introduction (Second edition).* Stanford, CA: CSLI Publications.

Zamorano Mansilla, J.R. (2004). Text generators, error analysis and feedback. In Delmonte et al., 2004, pp. 87–90.

Zock, M., & Quint, J. (2004). Converting an Electronic Dictionary into a Drill Tutor. In Delmonte et al., 2004 (pp. 41–44).