CHAPTER 1

Intelligent Language Tutoring Systems

1.1 Introduction

Computer-Assisted Language Learning (CALL) is the result of the convergence of several fields of research addressing the use of computers in language processing. While the influence of computational linguistics and machine translation may be indirect, modern CALL systems draw heavily upon the findings of these two areas. The handling of natural language by the computer contributes greatly to the fluency of interaction between the human user and the machine.

We find a direct influence on CALL in experiments in Programmed Instruction (PI). In the late sixties, CALL systems were primarily developed on large-scale, mainframe systems in universities where computer sessions were intended to replace classroom instruction. Programmed Instruction...
proposed the view that the best way of learning a task is to split it into small units, where the successful completion of one building-block leads to the next.

The generally sound pedagogical principle of dividing a large learning task into conceptually smaller units was, however, distorted by an overemphasis on rote memorization manifested in repetitive drills, multiple choice answers, and uninformative feedback. For example, in a typical drill exercise, the student was asked to type in the answer which was checked as each character was entered. If any error resulted, the computer assumed control and typed out the word “wrong” [Barker & Yeates 1985].

CALL questioned the effectiveness of such systems. A program with a simple letter to letter match is incapable of differentiating types of errors: not only is it, therefore, incapable of providing any valuable, evaluative feedback, but, in ignoring the source of the error when selecting another problem, it relies upon an inflexible, program-centered, rather than student-centered, definition of difficulty. While, in considering the early systems, one should make allowances for the limitations of the then current technology, the fact remains that, although PI has been refined over the years, it has never achieved a high degree of popularity [Price 1991]. The original linear programs (representing fixed sequences of instruction) improved, becoming more sophisticated branching programs; most, however, remained based on multiple choice answers. Inasmuch as the new is often an offshoot of the old, as among teaching approaches which emulate some procedures of the previous approach while rejecting others, CALL is historically related to PI, adopting however a concern for individualized learning, self-pacing, and immediate feedback.

Due to an increased emphasis of current teaching approaches on the student as an active participant in the learning process, increasingly more
attention is being paid to the interactive aspects of CALL systems [Burns, Parlett, and Redfield 1991]. As a consequence, CALL research has shifted its focus from drill-and-practice to tutoring systems. In tutoring programs, the computer is still the “judge-of-the-right-answer”. But as opposed to drill-and-practice applications, the path to the right answer involves a fair amount of student choice, control, and in particular, student-computer interaction [Warschauer 1996].

Marshall [1988] identifies the significant interactive qualities of Computer-Assisted Language Learning as one advantage of implementing the computer into the language classroom. True interaction, however, requires intelligent behaviour on the part of the computer. Without intelligence, the system is merely another method of presenting information, one not especially preferable to a static medium like print. Instead of multiple choice questions, relatively uninformative answer keys, and gross mainstreaming of students characteristic of workbooks, modern CALL is aiming at interactive computer systems possessing a high degree of artificial intelligence and capable of processing Natural Language input [Holland et al. 1993]. For a program to be intelligent, however, it must emulate the way a language teacher evaluates a student response.

A language teacher first examines the sentence and locates the error. Once the error has been identified, s/he decides on the reason for the error. In some instances, the source of an error might not be easily determined due to error ambiguity. For this reason, a teacher’s evaluation of the error will take into account further factors. These might include the student’s previous performance history and/or the frequency and difficulty of a particular grammatical construction. The final result of this process is error-contingent feedback$^1$ suited to learner expertise.
The goal of this dissertation is to design a model of a language instructor by evaluating and responding to student performance on foreign language exercises. The steps taken by a language instructor in the correction process are achieved by the computational analysis described and implemented, by way of example, in the German Tutor, an Intelligent Language Tutoring System (ILTS) for German.

The first step in the evaluation process of students’ input is the sentence analysis. In ILTSs, this is handled by the parser and the grammar. The parser analyzes a sentence according to the knowledge of the language encoded in the grammar. In section 1.1.1, I will discuss the importance of Natural Language Processing (NLP) to ILTSs. In section 1.1.2, I will introduce existing techniques for parsing ill-formed student input, a challenging task not unique to ILTSs.

1.1.1 Natural Language Processing

The strength of NLP is that it allows for a sophisticated error analysis where student tasks can go beyond multiple-choice questions and/or fill-in-the-blanks. Simple drills are based on string matching algorithms, that is, the student response is compared letter for letter against an answer key. However, one obviously cannot enter the arbitrarily many sentences required for meaningful practice into memory for purposes of comparison. NLP provides the analytical complexity underpinning an ILTS.

The pedagogical goal behind an ILTS is to provide error-contingent feedback suited to learner expertise. For example, if a student chooses an incorrect article in German the error might be due to incorrect inflection for

---

gender, number, or case. In such an instance the program must be capable of distinguishing between the three error types. For cognitive learning to occur, instructional feedback must address the different sources of an error [Rumelhart & Norman 1975, Venezky & Osin 1991].

The error analysis performed by the computer system forms the basis for error-contingent feedback. Garrett [1987] describes four kinds of feedback:

1. "presents only the correct answer
2. pinpoints the location of errors on the basis of the computer’s letter-by-letter comparison of the student’s input with the machine-stored correct version (pattern markup)
3. based on analysis of the anticipated wrong answers, error messages associated with possible errors are stored in the computer and are presented if the student’s response matches those possible errors (error-anticipation technique)
4. most sophisticated, uses an intelligent, Natural Language Processing (NLP) approach such as the “parsing” technique in which the computer does a linguistic analysis of the student’s response, ..."2

Studies [Nagata 1991, 1995, 1996, van der Linden 1993] addressing the question of what kind of feedback a computer program should give have shown that not only do students appreciate the more sophisticated feedback made possible by NLP, but also perform better on the language skills being taught. The fact that students learn better provides the rationale for employing parsers in Computer-Assisted Language Learning.

1.1.1.1 Syntactic Parsers

Natural Language parsers show great promise in the syntactic analysis of students’ input. They are best employed for ILTSs in introductory/intermediate language courses where the focus is primarily on form rather than on content. At the advanced level where a stronger emphasis is placed on content, syntactic parsers become less useful because they are less reliable.

Inevitably, grammatical constructions become more elaborate and ambiguous at the higher levels which causes difficulties for parsers in general.³

A number of studies have proven the usefulness of syntactic parsers in language learning. For example, a study conducted by Juozulynas [1994] at Miami University showed that only 20% of errors in the essays of second-year students of German are of a semantic nature.⁴ Juozulynas collected 349 students compositions. In all, 360 pages (313 essays) were included in the study. The error distribution in his study was:

- syntax: 28.6%
- morphology: 24.4%
- punctuation: 12.3%
- spelling: 14.7%
- semantics: 20%

A study by Rogers [1984] who collected 26 German essays with an average length of 559 words revealed similar results. Her distribution of errors is:

- syntax: 35%
- morphology: 34.5%
- lexical: 15.6%
- orthography: 9.5%
- complete transfer of English expression: 5.4%

Adjusting Rogers’ error classification to match Juozulynas⁵, 30% of errors are of semantic origin. The higher percentage of semantic errors in Rogers’ study might be due to the fact that “the Miami University student samples were from second-year students, while the students in Rogers’ study were...

---

³ There are a number of ILTSs which focus on communicative language learning and thus emphasize fluency over accuracy. See Holland et al. [1995a] and Swartz & Yazdani [1992].
⁴ Juozulynas classified semantic errors as errors of meaning, such as wrong word choice, “made-up” words, and errors in pronoun reference.
⁵ For an even comparison, Juozulynas took Rogers’ classification and assigned her categories lexical errors, complete transfer of English expression, and some types of syntactic and morphological errors to the semantic category.
were advanced, with at least four years of learning German in a formal environment, in many cases supplemented by visits to Germany.‘6

Given the outcomes of these studies, syntactic parsers can treat a large percentage of student errors and thus are a powerful tool in second language learning.7 The following section, section 1.1.2, will discuss the techniques employed by existing ILTSs in handling ill-formed input.

1.1.2 Parsing Ill-Formed Input in Intelligent Language Tutoring Systems

Although parsing ill-formed input has been a challenge for all NLP applications, ILTSs differ fundamentally from other NLP applications in how and why they handle ill-formed input. In most applications, the goal is to successfully parse and analyze a sentence, despite any errors. In ILTSs, however, the focus lies on tracing student’s language knowledge rather than on the linguistic analysis of well-formed input. Such systems are inherently prescriptive; for error-contingent feedback, they need to analyze where the student deviated from the expert knowledge. In this respect, ILTSs have a more difficult task. Fortunately, however, they usually do not deal with the relatively large input domain found in other NLP applications.

In ILTSs, errors occur, not because the student’s knowledge is a strict subset of the expert knowledge, but because the learner possesses knowledge potentially different in quantity and quality. For other NLP applications, Weischedel & Sondheimer [1983] analyze errors as either absolute or relative. Absolute errors refer to errors in the language output. Examples are tense and number mistakes, or word order problems. Relative errors address

7. See also Sanders [1991].
grammatical structures which are correct but are beyond the scope of the grammar.

Although this definition of absolute errors addresses the concerns of an ILTS, there is nonetheless a different goal behind the two conceptions: ILTSs aim at providing the student with error-contingent feedback. Thus ILTSs need to analyze student errors, and not simply withstand them. The error analysis performed by an ILTS provides the source of an error and thus enables the student to learn grammatical constructions of the target language. Simply withstanding errors fails the purpose of a parser-based ILTS.

Relative errors refer to language the user possesses that is beyond the system’s knowledge, which is rarely the case with an Intelligent Language Tutor. ILTSs focus on a language subset as described in students’ grammar books. It is a diagnostic tool for language learners to improve their second language grammar skills. The system responds to a student’s work on the basis of any errors the student may have made and the model of the student which the system has constructed to that point.

1.1.2.1 Anticipating Ill-Formed Student Input

ILTSs augment grammars that parse grammatical input in one of three ways. Each method overcomes some obstacles; however, all have concomitant disadvantages which will be discussed in the context of specific implementations in section 1.2. The following is a brief description of the three general approaches.

In the first technique, ILTSs may augment rules such that if a particular rule does not succeed, specific error routines, that is meta-rules\(^8\), force application of the rule by systematically relaxing its constraints. Meta-
rules are commonly employed for agreement errors. An example of a meta-
rule as defined by Chen and Kurtz [1989] is given in (c):

(1a) sentence(s(Np, Vp, Num)) --> noun_phrase(Np, Num),
    verb_phrase(Vp, Num).
(b) verb_phrase(vp(v(V), Num) --> intrans_verb(V, Num).
(c) intrans_verb(V, Num, Relax_flag) -->
    it_verb(V, Num);
    Relax_flag == true, it_verb(V, WrongNum),
    WrongNum == Num,
    error_flagging(X, subject_verb_agreement,
    [V, Num, WrongNum]).

The grammar rules given in (1a) and (b) specify that a sentence
consists of a subject and an intransitive verb which agree in number. If the
noun and the verb do not agree in number, the parse would simply fail. The
meta-rule given in (c), however, relaxes the constraint on number and will
allow the parse to proceed. The predicate error_flagging will add an error
message and the student can be informed that a mistake in number has
occurred.

In a second approach, ILTSs may augment the grammar with rules
which are capable of parsing ill-formed input (buggy rules)\textsuperscript{10} and which apply
if the grammatical rules fail. Buggy rules are distinct from meta-rules in that
they do not force application of the same rule, but rather, provide a distinct
rule for every ill-formed construction. For example, in German the past
participle always occurs sentence-finally in a main-clause, given in example
(2):

\begin{quote}
8. The term meta-rule used in this dissertation does not refer to schema for other context-free
rules as used in grammar formalisms such as GPSG.
9. The meta-rule given has been adapted from Chen & Kurtz [1989:58-9] who provide a rule
for a transitive verb.
\end{quote}
(2) Der Mann hat das Buch gelesen.
The man has read the book.

Due to native language interference\textsuperscript{11}, English learners of German will tend to place the past participle in the position which resembles English word order, that is, between the auxiliary and the noun phrase. A buggy rule would anticipate precisely this error by describing a structure where the past participle incorrectly occurs in the position derived from the English norm, successfully parsing the ungrammatical sentence and providing error-contingent feedback.

Finally, with feature grammar formalisms, ILTSs may also alter the unification algorithm itself such that in the event of conflicting feature values the parse does not fail, but instead applies a different set of procedures.

Parsers designed for language instruction typically contain components which anticipate or search for errors in the event that the grammatical rules are not successful, buggy rules being a common instance. Searching for errors, however, presents a number of problems.

First, due to the vast error scope and unpredictability of some errors, ill-formed input can only be partially anticipated [Yazdani & Uren 1988]. The fewer errors anticipated, the smaller the error coverage. Sentences containing errors which have not been anticipated by the designer cannot be processed by the system, inevitably resulting in generic rather than error-contingent feedback, and failing to achieve one of the primary goals of an ILTS.

\textsuperscript{11} Native language interference, or interlingual transfer, refers to the use of elements from one language while speaking/writing another [Richards 1971]. In learning a second language, the learner maps onto previously existing cognitive systems, which usually is the native language. The school of Contrastive Analysis [Lado 1957] tried to explain all errors through reference to the native language of the learner. However, this view was rejected by later work and studies in Error Analysis [Corder 1967, Selinker 1972]. See also Larsen-Freeman & Long [1991].
Second, anticipating errors lacks generality. Most systems base the search for errors on the native language of the student. Any system which anticipates errors requires that the same error in the target language is presented in each different source language. The fewer source languages considered the more user-limited the system. For example, a system which anticipates errors made by English learners of German, will tend not to handle errors specifically made by French learners of German.

For the reasons of error coverage, generality, and user scope, the methods employed in handling incorrect student input in ILTSs should be those that provide the most general treatment of errors, and that make minimal use of error anticipation.

The methods available to detect errors are determined by the programming language and the grammar formalism. ILTSs have been implemented in both procedural and declarative programming languages. Programming language to some extent determines the choice of grammar formalism, which in turn determines what detection methods can be applied, and how errors are perceived in the system [Matthews 1993, 1994, Sanders & Sanders 1989].

The following discussion will focus on ILTSs which have been implemented in either Augmented Transition Networks [Woods 1970] or declarative representations. The systems will be discussed with respect to the method used to detect errors and their error scope.

12. ATNs, in their original design, were conceived as parsing rather than grammar formalisms. However, they have developed into a grammar-like concept, especially in computational linguistics [Shieber 1986].
13. The systems discussed all place a pedagogical focus on form rather than content.
1.2 Augmented Transition Networks

Transition Networks (TNs) consist of a set of states, movement between which is controlled by rules according to the next element in the input. States are referred to as nodes; transitions between states are referred to as arcs. TNs are inherently procedural.

Augmented Transition Networks (ATNs) additionally employ registers which hold grammatical information. ATNs also allow actions to be associated with each arc, for instance, the setting of a register, or the calling of another network.

The Augmented Transition Network illustrated in Figure 1.1 parses sentences by following the arcs, and accepting, from the input string, constituents that are on the arc labels. The ATN can either accept a word, a proper noun for example, or it can call another entire subnetwork, VP. Parsing proceeds until a node is reached in which parsing can stop, indicated by the double circle. In addition, the arcs set and test register NUM to control number agreement between the subject and the verb.

The ATN given in Figure 1.1 parses sentences such as Mary sleeps by going through the following steps:

- Start in state 1 of S network.
- Accept NP (Mary, NUM = singular).
- Set register NUM of NP = singular.
- Set register NUM of S = singular.
- Proceed to state 2 of S network.
- Call VP network.
- Start in state 1 of VP network.
- Accept V (sleeps, NUM = singular).
- Set register NUM of VP = singular.
Proceed to state 2 of VP network and exit.
Test that NUM of S = NUM of VP
(the test succeeds since both have the value singular).
Proceed to state 3 of S network and exit.

While the Augmented Transition Network will successfully parse Mary sleeps, note, however, that the sentence Mary sleep will fail since the register on number agreement is violated. Violations of this kind, however, are common among second language students. An ILTS needs to parse such sentences, and analyze the errors to provide error-contingent feedback.

Meta- and buggy rules are the most common procedures employed for handling students’ errors in ILTSs. In ATN systems, agreement constraints are commonly relaxed with meta-rules, while buggy-rules are used for errors in word order. Buggy rules might also make up a complete second grammar, the student’s native language. In this case, meta-rules are used for analyzing overgeneralization errors\(^\text{14}\).

The pioneer work in ILTSs was developed by Weischedel et al. [1978]. The system is a prototype German tutor implemented as an Augmented
Transition Network with a semantic and syntactic component. The authors address four classes of errors:

1. spelling,
2. agreement,
3. frequently occurring errors due to native language interference, and
4. ambiguity.

Spelling errors are not treated within the system; instead a spelling correction algorithm is applied before the sentence is analyzed by the parser. The remaining three error classes will be discussed in the following sections.

**Agreement**

For agreement errors (subject-verb agreement, inflected noun phrase endings, word order among adverbial elements) the system uses predicates on the constituents. The predicates check whether the forms are correct. If a predicate evaluates to false, an error message is added. An example of a predicate might be “CASE-FAILED?”. If the predicate evaluates to true, a meta-rule is instantiated which loosens the constraint on case agreement.

While relaxing constraints will lead to a successful parse, the strategy does not necessarily lead to the desired pedagogical outcome. It is a technique for permitting ill-formed structures rather than for analyzing ill-formedness. For error-contingent feedback, a system needs to diagnose the source of an

14. Overgeneralization errors are the effects of particular learning strategies on items within the target language, that is, overgeneralization of target language rules. For example, a student might choose for the verb schlafen the regular past tense suffix -te *schlafte as opposed to the correct irregular form schlief. According to Richards [1971] such learning strategies appear to be universally employed when a learner is exposed to second language data; this is confirmed by the observation that many errors in second language communication occur regardless of the background language of the speaker. Error Analysis [Corder 1967, Selinker 1972] states that equal weight can be given to interlingual and overgeneralization errors.
error rather than merely relaxing the constraint. Consider example (3a), cited by Schwind [1990a, 1995]:

(3a) *Der Götter zürnen.
(b) Die Götter zürnen.
The gods are angry.

The subject of the sentence is Der Götter and thus requires nominative case. Der Götter, however, is inflected for genitive plural and the feedback would be The determiner is incorrectly inflected for case. However, the mistake is ambiguous and thus depends on the context. In example (3a), it is very unlikely that the student constructed a subject assigning genitive case which is the least common and thus hardest of the four cases in German [Schwind 1995]. The desired feedback should be The determiner is incorrectly inflected for number since Götter is plural and der is singular. This analysis is not possible by simply relaxing the constraint on case. It is the context in which the particular error occurs which needs to be considered to achieve the desired feedback.

Errors like the one given in example (3a) cause difficulty because the analysis performed by the parser is correct from a computational point of view but the feedback is pedagogically unsound. The parser and grammar will not flag der Götter as an error in number because the determiner and the noun make up a correct noun phrase in the genitive plural. However, when the verb combines with the noun phrase a case error is recorded because the verb requires nominative case.

15. Articles in German are ambiguous. Depending on where the error occurs der, for example, could be a mistake in case, gender, or number.
Frequently Occurring Errors due to Native Language Interference

For frequently occurring errors in word order due to native language interference, Weischedel et al. [1978] add each incorrect form as a buggy rule. An example of a native language interference error concerns the position of the past participle in German, illustrated in example (4a).

(4a) *Er hat gesehen das Kind.
(b) Er hat das Kind gesehen.
He has seen the child.

ATNs do not allow for a very general treatment of errors, particularly errors in word order. Each individual error has to be anticipated and a buggy rule needs to be implemented. Overgeneralization errors in word order, for example, although not implemented in Weischedel’s system, would need to be treated in a similar way as language transfer errors in ATNs, that is, by ad hoc rules.  

Similar failings, due to the need to anticipate errors can be found in Liou [1991]. That ATN system consists of an expert model, containing correct grammatical constructions and a bug model which anticipates the ill-formed structures. Liou’s program covers seven types of errors: det-noun phrases, conjunctions, verb morphology, subject verb disagreement, capitalization, although... but, and no matter... combinations. The error scope is fairly small and an extension would involve increasing the bug model by painstakingly encoding a buggy rule for each likely error.

16. Larsen-Freeman & Long [1991] list examples for errors due to factual misconception which they label simplification and induced errors. In addition there are mistakes as opposed to errors, a distinction made by Corder [1967]. Mistakes are random performance slips. In such cases, the student is aware of the lexemes and grammar rules of the target language but due to a lack of typing skills, concentration, and/or fatigue mistakes occur. All of these error types would require ad hoc rules.
Another example of the same heuristics is found in ALICE-chan, a multimedia foreign language learning environment for Japanese developed by Levin & Evans [1995]. The system covers grammatical topics typically taught in a first-year Japanese course and includes about 100 rules. The limitations are again that special rules need to be designed to parse errorful structures. Thus their treatment is not very general.

**Ambiguity**

The final error class Weischedel addresses is ambiguous readings. According to Weischedel et al. [1978] the intended meaning in example (5) is The man is giving the girl a hat rather than The girl is giving the man a hat.

\[(5) \text{Dem Mädchen gibt der Mann einen Hut.} \]

neuter, dative             masc., nom     masc., accusative

The heuristic Weischedel’s system uses to determine the interpretation intended by the student is to select the parse that gives the fewest errors. Ambiguous readings are rarely addressed by ILTSs although they are a common phenomenon with highly inflected languages such as German.  

Weischedel’s method is a computationally effective way to resolve ambiguous readings. However, from a pedagogical perspective “a more sophisticated routine would take into account the likelihood of various kinds of errors”[19]. The likelihood of an error is determined by the learner level and the frequency of a grammatical construction. Consider example (6):

---

17. The system also includes a Spanish module.
18. Levin and Evans [1995] resolve ambiguity through interaction with the user. Covington & Weinrich [1991] use the same heuristics as Weischedel et al. [1978]. However, they express their doubts about the usefulness of the method employed.
(6) Sie liebt er.
    fem., nom or fem., acc masc., nom
It's her he loves.

In example (6) the intended meaning could either be It's her he loves or She loves him. According to Weischedel's approach, the former interpretation would win since the sentence contains no errors, while the latter has an error associated with the direct object. However, a beginner student of German most likely intended to write she loves him, not knowing or overlooking the fact that lieben assigns accusative case to its object. The interpretation It's her he loves is certainly grammatical if the object receives stress. But in writing, even native speakers would probably underline Sie indicating the emphasis of the direct object. An even more puzzling example is given in (7):

(7) * Sie dankt er.
    fem., nom or fem., acc masc., nom
It's her he thanks or She thanks him.

Choosing the parse with the fewest errors, would not be sufficient for example (7). The verb danken assigns dative case. The two possible readings are either It's her he thanks or She thanks him but in both interpretations there is one mistake: either sie, nominative is used instead of ihr, dative or er, nominative instead of ihm, dative, respectively. In such an instance, relying on the parse with the fewest errors makes the choice of parse arbitrary, even though the student most likely intended Sie dankt ihm. A better approach to treating ambiguous readings will be provided in Chapter 3.

Augmented Transition Networks are limited in their application to ILTSs. In procedural programs, the data and its implementation are interwoven which makes a system less general and makes it more difficult to provide a large coverage of errors, because errors need to be anticipated and encoded similar to a pattern-matching mechanism.20
Furthermore, all the ATN systems described above are tied to a specific native language of the student, in Weischedel’s system, for example, English. For students with different native languages, the systems would require even more buggy rules since the error anticipation is derived from the native language of the student. Examples (8) - (10) illustrate the problem:

(8) German: Ich habe es gesehen.
(9) French: J'ai vu.
(10) English: I have seen it.

In German, the direct object appears in between the auxiliary and the past participle, the auxiliary always being in second and the past participle in final position in main clauses. A French learner of German is likely to commit errors concerning the position of the auxiliary, while the English learner of German will have problems with the position of the past participle. For an ATN to deal with the error caused by either native language, buggy rules of each error would have to be implemented.

A further shortcoming of procedural systems is that they are not modular. Such systems cannot be easily altered and applied to another language. The whole system has to be rewritten, when preferably one could replace only the language-dependent grammar within a language-independent shell.

In distinction to Augmented Transition Networks, declarative systems contain feature grammar formalisms which are based on unification. In addition to meta- and buggy rules, ill-formed input can be processed by

---

20. Johnson’s evaluation of general ATN systems can be well applied to ILTSs: “Like any procedural model, an ATN gives extensive possibilities for optimization and tight control over details of parsing strategy. The price to be paid in return is a danger that as the system increases in size, its logic becomes obscure, modification progressively more risky, and debugging more and more problematic.” Johnson [1983], p.72. See also Loritz [1992, 1995] who further discusses the implementation of ATNs in ILTSs.
altering the unification algorithm itself. Section 1.3 will discuss ILTSs employing declarative formalisms.

1.3 Intelligent Language Tutoring Systems Using Declarative Formalisms

The bulk of ILTSs are implemented as declarative representations. The main types of grammar formalisms used in ILTSs are Government Binding [Chomsky 1981, 1986] and Logic Grammars (Colmerauer’s Metamorphosis Grammar [1978] and Definite Clause Grammar [Pereira & Warren 1980]). In addition to the main types, HPSG as part of the GPSG family [Gazdar et al. 1985, Gazdar 1982] has been implemented in an ILTS by Hagen [1994].

There are three common methods of analyzing ill-formed input used in declarative-based ILTSs. In the first approach, in addition to the target grammar, the student’s native language is explicitly encoded to cover errors due to native language interference. Depending on the error scope of the system, meta-rules may be added for overgeneralization errors. The second approach does not rely on the native language of the student, instead relying on meta-rules to relax the constraints of the target language. In the third approach, the unification algorithm is altered in such a way that the system, despite clashing features, performs a parse and keeps track of the conflicting features.

An example of the first approach is Schuster’s [1986] VP² system for Spanish learners of English. The system focuses on the acquisition of English verb-particle and verb-prepositional phrase constructions. The program
contains two grammars, Spanish and English. An obvious shortcoming of the system is that it can only deal with native language interference errors. Overgeneralization errors, another significant source of students’ errors, are ignored and cannot be recognized by the system.

Catt & Hirst [1990] overcome the shortcoming in Schuster’s system by implementing meta-rules for overgeneralization errors. Their system Scripsi consists of three grammars: English as the target language, French and Chinese as the native languages of the learners. Whenever the input cannot be recognized by English rules alone, the system applies the rules of French or Chinese.21 Although the system represents an extension of Schuster’s system, nonetheless, an important error class passes unrecognized: errors in word order can only be detected if due to native language transfer. A further, rather apparent, limitation of the authors’ approach lies in its lack of generality. First, the same error in the target language has to be encoded in each source grammar, French and Chinese. Second, identical grammatical structures are encoded in all three grammars leading to redundancies.

Wang & Garigliano [1992, 1995] designed a Chinese Tutor for English native speakers. In their design, they attempted to avoid the kind of redundancies found in Catt and Hirst’s system. The Chinese Tutor models only the fragments of the grammatical rules in the native language which are different from the corresponding rules in the target language. Additionally, the authors address only those transfer errors justified by empirical data they collected. They found that 78% of the errors made by English learners of Chinese are due to transfer. The remaining 22% are ignored in their system.

21. It is not quite clear from the description in Catt & Hirst [1990] whether a sentence would actually run through three grammars or whether the students possibly identify their native language when using the program and required switches are set.
The percentage of transfer errors is naturally closely dependent on the two languages compared.\textsuperscript{22}

The deficiencies of the systems employing native grammars are either poor coverage of errors, as in the case of neglecting errors due other than to native language interference, and lack of generality. The systems also overtly rely on a particular native language of the user and are thus limited in their scope. This limitation becomes more problematic due to the increase in multi-culturalism and on-line Distance Education over the Internet. Any overt presumption of the native language of the student is severely restrictive.

A variation of error handling is found in other declarative systems. For example, Kurtz, Chen, and Huang [1990] developed a DCG system XTRA-TE for Chinese learners of English. The authors only use correct grammar rules with meta-rules that relax constraints at multiple levels. Level 1 contains all grammatical constraints, Level 2 relaxes syntax, Level 3 semantics, and Level 4 both syntax and semantics. In addition, for translation exercises the system performs a literal word-by-word translation. The authors' system is more general in the sense that they do not anticipate errors on the basis of the native language of the student. However, the system cannot handle errors in word order. The only constraints which can be relaxed are agreement errors, both syntactic and semantic.

A similar pitfall can be found in a system for learning English developed by Covington & Weinrich [1991]. The program is written in Gulp, an extended form of Prolog which includes a notation for feature structures

\textsuperscript{22} Weinreich [1953] asserted that the greater the difference between the two languages, i.e. the more numerous the exclusive forms and patterns in each, the greater is the learning problem and the potential area of interference.
The system focuses on agreement, insertion, and omission errors. Insertion and omission errors are due to a violation of the subcategorization list of the verb. For example, a student might use an inherently transitive verb without an object. The authors relax constraints by implementing a lenient rule for each rule which parses well-formed input. However, errors in word order pass unrecognized.

Meta-rules, by definition, cannot address errors in word order. They can only relax constraint violations of grammatical features. All systems which handle word order errors anticipate these by implementing buggy rules. However, in all cases the result is very user-specific since the systems rely on a particular native language of the student. The ones which implement more than one native language lack generality. The same error in the target language is presented in each source language.

The third approach in error detection within declarative systems makes use of the feature grammar formalism itself. In declarative implementations, the grammar makes heavy use of the operation on sets of features called unification. Unification-based grammars place an important restriction on unification, namely that two categories A and B fail to unify if they contain mutually inconsistent information [Gazdar & Pullum 1985, Knight 1989]. However, this inconsistent information constitutes exactly the errors made by second language learners. For example, if the two categories A and B do not agree in gender a parse will fail. A system designed to accommodate the errors of second language learners' requires either a modification of the unification algorithm itself or feature specifications capable of overcoming the gender restriction.

23. Covington & Weinrich [1991] use features to describe the particular error but their error detection mechanism is based on buggy rules rather than on the feature grammar formalism.
Hagen [1994] developed an ILTS, GrammarMaster, for French as a second language. The program is driven by an object-oriented, unification-based parser written in HyperTalk. The system addresses three particular grammatical constructions: conjunctions, reflexive binding, and dislocated, missing, and superfluous constituents.

The goal of the system is to show an implementation of some “thorny problems of complex grammatical constructions.” Hagen’s system borrows analyses from a number of grammar formalisms: Head-Driven Phrase Structure Grammar (HPSG), Generalized Phrase Structure Grammar (GPSG), and Government Binding (GB). Missing constituents are handled with the SUBCAT feature of HPSG, reflexive binding makes use of the foot feature principle (GPSG), and superfluous constituents are controlled by thematic assignment (GB).

Hagen relaxes the unification algorithm to block parsing failure such that if the parser “detects contradictory features like [genM] and [genF], it preserves the features on the major part of speech (in the case of noun phrases, the noun) and inserts the contradictory feature on a minor part of speech like an article into an error stack along with a corrective message.”

Hagen’s implementation of the principles from GPSG, HPSG, and GB allow for a general treatment of the errors considered. The errors do not need to be anticipated. If any of the grammar principles does not hold, the constituent is flagged and an error message is attached. However, Hagen addresses a very small range of errors; the system also lacks a uniform syntactical framework and seems linguistically rather than pedagogically motivated. In addition, altering unification implies inserting procedural

techniques in an inherently declarative algorithm. Thus the analysis becomes closely tied to its implementation, that is, the grammar only works if unification is changed in a certain way.

Schwind [1990a, 1995] developed a system for German based on Colmerauer’s Metamorphosis Grammar [Colmerauer 1978]. Her system covers a broad range of errors: agreement, syntactic and semantic errors. Her method of analyzing agreement errors is based solely on the feature grammar formalism. The class of agreement errors covers errors in gender, number, person, and case of noun phrases. The system does not anticipate these errors. Instead, each lexeme is specified for every possible grammatical occurrence.\(^{27}\)

In addition, the unification-based algorithm builds up two sets of feature structures: “… unify (a,b,r,e) holds whenever r is the result of the unification and e is the set consisting of all the pairs of values for which a and b could not be unified, together with all the symbols contained in the symmetrical difference between a and b.”\(^{28}\) Thus Schwind’s definition of unification differs from the usual one [Karttunen 1984] in the sense that the parse does not fail but instead records the elements which do not unify.

Schwind’s analysis also allows for discriminating among errors that are phonetically identical, but differ in their source. These errors occur within

26. Grammatical constructions which are linguistically and computationally complex do not necessarily present difficulties for the language learner. For example, the errors second language learners make with conjunctions are limited to subject/verb number agreement and case assignment of the subject or verb complement. While conjunctions present a challenge from a computational point of view since it is of utmost importance for the linguist that the parser displays the correct analysis, for the language learner it is the feedback that outweighs the linguistic and computational analysis [Farghali 1989].
27. In German, there are three genders, four cases, and two numbers. While this results in 16 potentially distinct lexical forms, there are only six forms for definite articles, which are phonetically distinct. For example, the definite article der can be [genitive, plural], [genitive, feminine, singular], [nominative, singular, masc], [dative, feminine, singular]. Schwind [1995] uses only one lexical entry for der, see also [McFetridge & Heift 1995].
noun phrases. The problem with noun phrases in German derives from the fact that there are three features (gender, number, and case) any of which can be wrong.\(^{29}\) Consider examples (11a) - (13a), as cited by Schwind [1990a, 1995]:

\[(11a) \text{*Der Kind spielt.}\\ (b) \text{Das Kind spielt.}\\ \text{nominative}\\ \text{The child is playing.}\]

\[(12a) \text{*Er gibt der Kind Milch.}\\ (b) \text{Er gibt dem Kind Milch.}\\ \text{dative}\\ \text{He is giving the child milk.}\]

\[(13a) \text{*Sie kennt der Kind.}\\ (b) \text{Sie kennt das Kind.}\\ \text{accusative}\\ \text{She knows the child.}\]

In examples (11a) - (13a), the error occurs with \text{der Kind}. According to Schwind, the desired feedback in example (11a) and (12a) is The determiner is incorrectly inflected for \textit{gender}, while in example (13a) it is The determiner is incorrectly inflected for \textit{case}. In the three examples, although the surface error \text{der} is identical, the sources of the errors are distinct. This is due to the distinct case requirements of the three examples and the ambiguity of the determiner \textit{der}. In example (11a), the source of the error is gender since \text{der} is a possible article in the nominative (\text{der, die, das}). However, since \text{der} is not a possible accusative article (\text{den, die, das}) as required in example (13a), the source of the error lies in case. Schwind [1995] applies case filtering to achieve the desired feedback. Her system first checks whether the article is a possible determiner for the required case. If it is, the system responds with a gender.

\(^{29}\) These three features do not cover the declension of adjectives. Adjectives also inflect according to whether or not they are preceded by a definite or indefinite article.

\(^{30}\) Schwind [1995], p. 312.
error; if not, the source of the error is case. The shortcoming of her analysis is that the system assumes that students in general are more likely to know grammatical case than gender. This is apparent in example (12a) which Schwind [1995] analyzes as an error in gender as opposed to in case. The error is ambiguous due to the grammatical ambiguity of der. If the student assumed that Kind is feminine, the error is due to wrong gender. However, if the student knows the correct gender but did not assign the correct case, the error source is wrong case. Ambiguous errors will be further discussed in Chapter 2.

Schwind uses the same analysis for semantic errors. The system recognizes the violation of semantic restrictions on verbs and their complements. For this, she provides a semantic network where semantic constraints are expressed as features. For example, while computer is [inanimate], woman is marked as [animate, human]. The analysis is again very general, that is, the errors do not have to be anticipated.

One significant deficiency of Schwind’s system is its handling of high-level syntax errors. In syntax, Schwind makes a distinction between low- and high-level. Low-level syntax errors are errors of insertion or omission while high-level errors refer to errors in word order. Schwind uses one buggy rule for each constituent of insertion and omission. High-level syntax errors, however, must be anticipated and according to Schwind [1995] their treatment is not general.

Feature grammars are more promising than ATNs for Intelligent Tutoring Systems. The data and implementation are kept separate which allows for easier expansion. Feature grammars also accommodate more

31. see also Weischedel et al. [1983]. The authors present a very similar analysis of semantic networks. However, it is not as complete as the one by Schwind [1990b].
methods of error detection. In addition to meta- and buggy-rules, they allow for alteration of the unification algorithm itself. Altering the unification algorithm has its pitfalls as mentioned; however, a more general treatment of errors than with meta- and buggy rules alone can be achieved.

Feature relaxation presents an alternative to existing methods and is the method adopted in this dissertation. Unlike existing ILTSs, the analysis described does not seek or anticipate errors, but instead emulates the way a language instructor evaluates a student’s response. The computational analysis reflects the pedagogical bias of the system.

1.4 Evaluation of Intelligent Language Tutoring Systems

The foregoing discussion illustrates two developmental stages in ILTSs that primarily focus on form rather than on content. The early ILTSs, as the one by Weischedel et al. [1978], concentrated solely on tackling the computational problems of parsing ill-formed input as opposed to embedding pedagogic considerations into such systems. For example, while Weischedel’s system considers ambiguous readings, they are addressed from a computational rather than a pedagogic point of view. This is evident in the algorithm used to handle such errors. Selecting the parse with the fewest errors is a computationally effective method, but it does not address the likely cause of error. A pedagogic system would consider the performance level of the learner and/or the language learning task in order to address ambiguous readings.
The second phase of ILTSs, while still extending the parsing capabilities of ill-formed input, also attempts to emphasize pedagogic considerations. For example, Schwind [1995] and Holland [1994] seek a wider error coverage in a computationally more general way, and also address language learning pedagogy in considering ambiguous errors. However, their pedagogic focus is limited. While the algorithm addresses ambiguous errors to some extent, contingent errors cannot be handled adequately.

The discussion of existing ILTSs shows that neither phase has been completed. From a computational point of view, some errors such as errors in word order, for example, still need to be anticipated and thus are not addressed in a general way. The algorithms are based on a particular native language of the student and the systems are thus limited in error coverage and user scope. From a pedagogical perspective, ambiguous and contingent errors are important to language instruction.

The analysis presented in this dissertation addresses both developmental phases in ILTSs. From a computational point of view, a wider class of errors is addressed in a general way, thus not sacrificing error coverage and user scope for errors in word order. From a pedagogical point of view, the analysis considers the learner by emulating the ways a language instructor evaluates and responds to a student’s performance in language learning tasks. As a result, ambiguous and contingent errors, as well as multiple sentence readings, are treated with a strong pedagogic focus. In addition, instructional feedback is suited to learners’ expertise.

The following section will outline the German Tutor, which illustrates the analysis described in this dissertation by way of example.
1.5 The German Tutor

The NLP component of the German Tutor parses students' answers to introductory German exercises and returns structures which specify student model updates and possible feedback of different levels of granularity correlated with students' expertise. The concept of granularity has been previously applied to an Intelligent Tutoring System for LISP programming [see Greer & McCalla 1989]. For ILTSs, granularity is particularly important in framing responses to learners' errors. Inexperienced students require detailed instruction while experienced students benefit best from higher level reminders and explanations [LaReau & Vockell 1989].

The design of an ILTS implemented in the German Tutor consists of five components: the Domain Knowledge, the Licensing Module, the Analysis Module, the Student Model, and the Filtering Module. The five components, given in Figure 1.2 correspond to the steps a language instructor applies in evaluating and responding to students' errors.

The Domain Knowledge represents the knowledge of the language. It consists of a parser with a grammar which parses sentences and phrases to produce sets of phrase descriptors [McFetridge & Heift 1995]. A phrase descriptor is a model of a particular grammatical phenomenon such as case assignment or number agreement. A phrase descriptor records whether or not the grammatical phenomenon is present in the input and correctly or incorrectly formed.

Due to structural ambiguity found in language, a parser produces more than one parse for many sentences. The Licensing Module selects a
Figure 1.2: The German Tutor

Student Input →

Domain Knowledge
knowledge of the language encoded in the grammar

→ sends phrase descriptors of all parses

Licensing Module
selects the desired parse

→ sends all phrase descriptors of the selected parse

Analysis Module
returns student model updates and produces instructional feedback of increasing abstraction

→ sends a student model update and instructional feedback

Student Model
keeps learner model updates and decides on the student level

→ sends instructional feedback suited to student expertise

Filtering Module
decides on the order of instructional feedback

Error-contingent Feedback Suited to Learner Expertise
parse by taking into account factors a language instructor considers in evaluating a student’s response. In determining the most likely sentence reading, a language instructor considers the level of instruction, the frequency of the grammatical construction, and thus the likelihood of the error.

The Analysis Module incorporates a language instructor’s knowledge in pinpointing the precise source of an error. The Analysis Module takes a phrase descriptor as input and generates sets of possible responses to the learner’s input that the instruction system can use when interacting with the student. The level of the learner, either expert, intermediate, or novice according to the current state of the Student Model, determines the particular feedback displayed. The Student Model records mastery of grammatical structures as well as structures with which the learner has problems.

The Filtering Module determines the order of the instructional feedback displayed to the learner. The system displays one message at a time so as not to overwhelm the student with multiple error messages. In addition, the Filtering Module takes into account contingent errors, a class of multiple errors to be discussed in Chapter 4.

1.5.1 The Domain Knowledge

The parser analyzes students’ input according to the knowledge of the language encoded in the grammar. The parser and the grammar thus provide the linguistic analysis of students’ input. The grammar for the German Tutor is written in ALE (The Attributed Logic Engine), an extension of the computer language Prolog. ALE is an integrated phrase structure parsing and definite clause programming system in which grammatical information is expressed as typed feature structures. Typed feature structures combine type
inheritance and appropriateness specifications for features and their values [Carpenter & Penn 1994].

The grammar formalism used is derived from Head-driven Phrase Structure Grammar [Pollard & Sag 1987, 1994, Nerbonne et al. 1994]. This theory is one of a family which share several properties. Linguistic information is presented as feature/value matrices. Theories in this family are to varying degrees lexicalist, that is, a considerable amount of grammatical information is located in the lexicon rather than in the grammar rules. For example, Figure 1.3 illustrates a minimal lexical entry for geht. The subcategorization list of the verb, notated with the feature \textit{subj}, specifies that \textit{geht} takes a subject which is minimally specified as a singular noun. Rules of grammar specify how words and phrases are combined into larger units according to the subcategorization list. In addition, there are principles which govern how information such as the head features is inherited.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{geht_lexical_entry.png}
\caption{Partial Lexical Entry for \textit{geht}}
\end{figure}
The philosophy of the analysis described differs from other systems designed to parse ill-formed input in ILTSs in that it does not seek or anticipate errors, but instead records whether or not grammatical constraints are met, the goal being to analyze students’ language input rather than the more modest aim of recognizing ill-formed constructions. The analysis models the steps taken by a language instructor in the correction process of students’ input.

The grammar itself is written as a set of constraints. Any constraint such as the one in Figure 1.3 that the subject of the verb geht must be a singular noun may block parsing if not successfully met. The terminology used in Figure 1.3 will be described in more detail in Chapter 2.

In HPSG, the information flow is modeled by structure-sharing and not by transformation or movement. Two or more distinct attributes (or paths) within a feature structure are said to be structure-shared if their values are specified by one and the same feature structure [Pollard & Sag 1987]. Structure-sharing in feature-value matrices is indicated by multiple occurrences of a coindexing box labeling the single value. For example, in the lexical entry for the verb geht, given in Figure 1.3, the number feature of the verb and its subject share one common value, sg. If this condition is not met, that is, either the subject or the verb is not singular, unification fails. In ILTSs, however, sentences containing such inconsistent information need to parse successfully so that the error can be discovered, analyzed, and reported to the student.

The constraint on number agreement can be relaxed, however, by changing its structure so that, rather than checking that the noun is singular, the system records whether or not the subject of geht is in the singular. To achieve this, the noun is no longer marked as [num sg], but instead the path
num|sg terminates with the values error or correct. For example, for a singular noun phrase, the value of the path num|sg is correct, while it is error for a plural noun phrase. The two partial lexical entries are given in Figure 1.4 and Figure 1.5, respectively.

The verb geht records the value of sg from its subject (Figure 1.6). If the value of the path num|sg is correct, the subject is in the singular. In case of a plural noun, geht records the value error for number agreement.

The goal of the parser and the grammar is the generation of phrase descriptors. A phrase descriptor is implemented as a frame structure that models a grammatical phenomenon. Each member of the frame consists of a name followed by a value. For example, subject-verb agreement in number is modeled by the frame [number, value] where value represents an as yet
uninstantiated value for number. If the grammatical phenomenon is present in the student’s input, the value is either correct or error depending on whether the grammatical constraint has been met or not, respectively. If the grammatical constraint is missing, the feature value is absent. Consider examples (14a) and (b):

(14a) *Er gehen.
(b)  Er geht.
    He is leaving.

The phrase descriptor for subject-verb agreement in number in example (14a) is [number, error], while that for the sentence in (b) is [number, correct]. For either sentence, (14a) or (b), the information will be recorded in the Student Model. A system presented with (14a), however, will also instruct the learner on the nature of subject-verb agreement in number.
In addition to the grammatical features defined in HPSG the grammar uses a type descriptor representing the description of the phrase that the parser builds up. This type is set-valued and is initially underspecified in each lexical entry. During parsing, the values of the features of descriptor are specified. For example, one of the members of descriptor, vp_num in Figure 1.6, records the number agreement of subject-verb in a main-clause. Its value is inherited from the sg feature specified in the verb geht. Ultimately, descriptor records whether the sentence is grammatical and what errors were made.

1.5.2 The Licensing Module

The Licensing Module determines the most likely reading of a sentence. Due to language ambiguity, parsers generally produce more than one syntactic structure. Feature relaxation leads to an even greater number of parses. Selecting the first parse or the parse with the fewest errors is inappropriate for the design of an ILTS described. Neither solution takes into account the likelihood of an error, possibly resulting in misleading feedback. The design of the Licensing Module reflects a language instructor’s decision in selecting the most likely reading of a sentence by considering the source language of the learner, the level of instruction, and the frequency of a grammatical construction. This is discussed in detail in Chapter 3.

1.5.3 The Analysis Module

The third component of the system is an Analysis Module which encodes knowledge of the language instructor to determine the exact source of an error. The Analysis Module takes phrase descriptors as input and generates possible responses that the instruction system can use when
interacting with the learner. A response is a pair that contains a message the
system will use to inform the learner if a phrase descriptor indicates there
has been an error and a Student Model update. The Student Model update
contains the name of a grammar constraint in the Student Model along with
an instruction to increment or decrement the corresponding cumulative total.

The Analysis Module generates sets of instructional feedback of
increasing abstraction. As an example consider the ungrammatical sentence
in (15a). An inexperienced student should be informed that Mädchen is a
neuter noun, that the verb danken is a dative verb and that the determiner
das is incorrect. A student who has mastered case assignment (as indicated
by the Student Model) may be informed only that the case of the object is
incorrect.

(15a) *Der Mann dankt das Mädchen.
(b) Der Mann dankt dem Mädchen.
The man thanks the girl.

The Analysis Module is implemented in DATR [Evans and Gazdar
1990], a language designed for pattern-matching and representing multiple
inheritance. The Sussex version of DATR is implemented in Prolog.

Nodes in DATR are represented by the name of the node followed by
paths and their values. A partial representation of the node which
corresponds to the phrase descriptor that records the position of a finite verb
in a main clause is given in Figure 1.7.

The paths in a node definition represent descriptions of grammatical
constraints monitored by phrase descriptors. The matching algorithm of
DATR selects the longest path which matches left to right. Each path in a
node is associated with atoms on the right of ‘==’. There are three learner
levels considered: expert, intermediate, and novice each of which has four atoms. The first atom in each set refers to a grammar node, followed by an increment, a Boolean value, and a message.

For example, the node `posmainclfin`, is responsible for monitoring finite verb position in a main clause. If there has been an error identified by the parser, the parse will generate the phrase descriptor `[main_clause [position_mainclause [finite error]]]`. This will match the path `<main_clause position_mainclause finite error>`. This clause will concatenate the name of a slot in the Student Model `posmainclfin` and its increment with a message. The message provides the student feedback, and the remaining information is used to adjust the Student Model.

32. A phrase descriptor that contains the value `correct` indicates that the grammatical constraint has been met in which case no feedback message is specified. A phrase descriptor that contains the value `absent` is ignored by the system. Thus the list to the right of `==` is empty.
33. The Analysis Module will be explained explicitly in Chapter 4.
34. The Boolean values indicate whether it is an increment or decrement: true for increment, false for decrement.

Figure 1.7: DATR Code Listing for a Finite Verb in a Main Clause
If the phrase descriptor is [main_clause [position_mainclause [finite correct]]] and thus there has been no error, no message is associated with the name of the slot in the student model and its decrement. The information, however, still contributes a Student Model update in order to record the success.

Finally, if the grammatical phenomenon was not present in the student input, the phrase descriptor [main_clause [position_mainclause [finite absent]]] is generated. However, the phrase descriptor is uninstantiated and thus no information is associated.

The final result of the Analysis Module is a student model update and a list of possible learner responses from a coarse to fine grain size. The information is further processed by the Student Model.

1.5.4 The Student Model

The Student Model keeps track of the learner history and provides learner model updates. There are three learner levels considered in the system: novice, intermediate learner, and expert. The Student Model passes instructional feedback suited to learner expertise to the Filtering Module.

The Student Model consists of 79 grammar constraints which are equally distributed among main, subordinate and coordinate clauses. The distinction between the three clause types is necessary for two reasons. First, if not kept distinct the system would overwrite the descriptors instantiated by one clause type once it parses the next. Second, in some instances, there are two grammar constraints monitoring noun phrases practiced in isolation, one for verb-initial position, and one for the entire sentence. A complete list of the grammar constraints is provided in the Appendix.

35. Each clause contains 25 grammar constraints per clause. In addition, there are two grammar constraints monitoring noun phrases practiced in isolation, one for verb-initial position, and one for the entire sentence. A complete list of the grammar constraints is provided in the Appendix.
different grammatical rules which apply to main and coordinate as opposed to subordinate clauses. For example, in a subordinate clause in German the finite verb appears in final position while in a main clause it is always the second constituent. To capture this distinction and to provide error-contingent feedback suited to learner expertise the system needs to identify in what kind of clause the constraints were met or violated.

Each of the grammar constraints represents a grammatical phenomenon; the phrase descriptors report which constraints have been met or not. Each student starts out at the intermediate level. Once a student uses the system, the Student Model adjusts the counter of each grammar node accordingly. Setting the initial level to intermediate is arbitrary but reasonable, and the model will quickly adapt for various profiles.

On the basis of the dynamic learner model, the system decides on the granularity of the instructional feedback. The central idea is that an expert requires less detailed feedback than a beginner learner [Fischer & Mandl 1988]. For each grammar node, the Student Model checks which level the student has achieved and sends the appropriate instructional feedback to the Filtering Module.

1.5.5 The Filtering Module

The final component of the design is the Filtering Module which decides on the order of the instructional feedback displayed to the learner. In a language teaching situation a student might make more than one error within an exercise. In a typical student-teacher interaction, however, the language instructor does not overwhelm the student with multiple error reports. Instead, a language instructor reports one error at a time by considering the salience of an error. This pedagogy has been transferred to the analysis of this
dissertation. The grammar constraints produced by the phrase descriptors are hierarchically organized. An Error Priority Queue determines the order in which the instructional feedback is displayed by considering the importance of an error in a given exercise and the dependency between syntactically higher and lower constituents. Once the student makes the required correction, the whole evaluation process repeats.

Contingent errors, a class of multiple errors which presents a particular problem for existing systems, are taken into consideration. For instance, example (16a) illustrates that denken subcategorizes for the preposition an, while von is a dative preposition requiring the dative pronoun dir. Ignoring the dependency of errors, the feedback to the student would be This is the wrong preposition and This is the wrong case of the pronoun. The feedback is correct with regards to von, however, the pronoun dich is not incorrect if the student correctly changes the preposition von to an. Depending on the order of the feedback, the student might end up changing von to an and dich to dir and wind up utterly confused because the case of the pronoun had been flagged as an error in the original sentence.

(16a) * Ich denke von dich.

<table>
<thead>
<tr>
<th></th>
<th>dat prep.</th>
<th>acc pronoun</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>Ich denke</td>
<td>von</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dich.</td>
</tr>
</tbody>
</table>

(b) Ich denke an dich.

<table>
<thead>
<tr>
<th></th>
<th>acc prep.</th>
<th>acc pronoun</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>Ich denke</td>
<td>an</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dich.</td>
</tr>
</tbody>
</table>

I am thinking of you.

The error in the pronoun is correctly flagged by the system from a purely logical point of view. However, from a pedagogical perspective reporting the error in the pronoun is redundant and even misleading. In such instances, only the error in the preposition is reported while the error in the pronoun is recorded in the Student Model. Once recorded in the Student Model the information can be used for assessing and remediating the student.
1.6 Summary

ILTSs aim at modelling the human tutor. Their goal is to provide error-contingent feedback and to allow for an individualization of the learning process. From a computational point of view, the challenging task lies in processing ill-formed input.

The existing systems discussed all depend on buggy and meta-rules, and/or an alteration of the unification-based algorithm itself. Even in the most general systems, errors in word order need to be anticipated and thus cannot be treated in a very general way. The search for particular errors is also to a large extent based on the native language of the student. Any system which anticipates errors requires that the same error in the target language be presented in each different source language. The fewer source languages considered the more user-limited the system. In addition, in a number of systems the data and its implementation are interwoven complicating their design, maintenance, and ultimately restricting their error coverage.

The strengths of the analysis of this dissertation are manifold: first, the design contains a pedagogic component and thus it emulates a language instructor by evaluating and responding to students’ language input. The result of the entire process is error-contingent feedback suited to learner expertise. Second, the grammar is sufficiently general that it treats grammatical and ungrammatical input identically for the phenomena it is designed to handle. This generality has the advantage of reducing the number of rules required by the grammar. Third, the decoupling of the parsing system from the analysis of whether or not the input is grammatical has the practical advantage that development of each can proceed independently. Fourth, the
analysis described can successfully handle ambiguous and contingent errors, a capability beyond that found in existing systems.

The modular design of the entire system also allows for easier adaptation to other languages. Although the grammar itself is language-dependent, the system does not anticipate specific errors and thus is not built entirely from the perspective of a particular native language. The Licensing Module, the Analysis Module, the Student Model, and the Filtering Module are language-dependent only to a trivial degree. While the phrase descriptors and thus the instructional feedback will vary from one language to another, the overall architecture of these modules can be applied to languages other than German.