Department of Computer Science,
University of Otago

Technical Report OUCS-2005-02

A system for generating teaching initiatives in a computer-aided language learning dialogue

Author:
Nanda Slabbers
Department of Computer Science, University of Twente, the Netherlands
(Intern, Artificial Intelligence Group, Department of Computer Science, University of Otago)

Department of Computer Science,
University of Otago, PO Box 56, Dunedin, Otago, New Zealand

http://www.cs.otago.ac.nz/research/techreports.html

A system for generating teaching initiatives in a computer-aided language learning dialogue

Nanda Slabbers

Department of Computer Science
University of Otago
New Zealand
nanda@cs.otago.ac.nz

Supervisor: Alistair Knott
alik@cs.otago.ac.nz

Department of Computer Science
University of Twente
The Netherlands
n.slabbers@student.utwente.nl

Supervisor: F. M. G. de Jong
f.m.g.dejong@ewi.utwente.nl

February, 2005

Abstract

This document describes an extension made to a dialogue-based CALL system, to allow the system to take initiatives in the dialogue. The initiative module is triggered when the user passes the initiative to the system during the course of a dialogue. The system will then generate a set of "possible initiatives" and decide which one is best based on a number of criteria.

The initiative module supports two different goals. The first one is the formal goal of generating an initiative which is appropriate in the context. Furthermore the dialogue system is used for second language learning and therefore the module also supports the substantive goal of teaching the student a number of syntactic constructions.

Contents

Introduction

Learning a second language requires much time and practice. People who decide to learn the second language in a classroom will follow many lessons in class. In such a classroom students often learn grammar and vocabulary, but to encourage the student to actually use the language a more focussed interaction is often needed, where the student talks individually to a single teacher or tutor.

Dialogue is a useful environment for second language learning for a number of different reasons. First of all, teachers' questions give immediate feedback about student comprehension, because the teacher can use the student's response to detect difficulties. Furthermore learning a language by means of a dialogue forces the student to produce language instead of just processing the language. Finally students and teachers can both shape the learning process; not only the teacher can decide what to teach, but also the student can ask questions.

An automated human-computer dialogue system can stimulate the learner to produce the language and can therefore improve the students' learning process. Building such a dialogue system requires modelling the teacher's behavior and his questioning behavior in particular. This is a challenging task because teachers have to make innumerable choices which often depend quite subtly on the current context.

The CALL (Computer-Aided Language Learning) system which is described in this report is part of the Te Kaitito system, a bilingual dialogue system for the English and Māori languages. The CALL system is meant for people who are learning Māori (the indigenous language of New Zealand) as a second language. In the dialogue system the user can enter facts and ask questions. The system will respond with a simple 'ok' message if the user enters a fact which it successfully interprets, and will try to answer the question if the user enters a question. In this way the user always has the initiative himself, but he can also decide to pass the initiative to the system. He can do so by simply pressing the Enter key without entering other input. At that point the initiative module is called which will generate an initiative from scratch. This report describes what happens when the system generates an initiative.

The plan of this report is as follows. First some related literature is dealt with in Chapter 1. Then the design of the module is described in Chapter 2. Chapter 3 is on the implementation of the different steps of the algorithm. In Chapter 4 the results are presented using some example dialogues. Chapter 5 concludes with a summary and some suggestions for future extensions.

Chapter 1

Literature

Creating initiatives in a language teaching environment is a rather new field of inquiry. Therefore there is not much literature on exactly this subject, but there are some related subjects with relevant literature which will be described in this chapter.

In section 1.1 we will look at the Te Kaitito system, the system in which the initiative module is embedded. This will contain a general description of the system, an overview of the architecture and an explanation of some relevant topics used by the system. In section 1.2 second language learning and in particular second language teaching are dealt with. Section 1.3 describes some existing teaching systems, some of which are dialogue-based, others involving language teaching. Finally, using this literature a somewhat more detailed goal description is given in section 1.4.

1.1 Te Kaitito

1.1.1 Te Kaitito system

Te Kaitito is a bilingual system for the English and Māori language, see (Knott et al., 2002), (Knott et al., 2003) and (Vlugter et al., 2004). The system consists of two main components: a sentence-to-sentence translator and a dialogue system. The translator simply translates sentences from one of the languages to the other language. The system is bidirectional which means that every sentence that can be parsed can also be generated, in either one of the languages. In the dialogue system the user can have a conversation with the system in either English or Māori. In this dialogue system the meanings of the user's utterances are stored by the system and are used to answer the user's questions.

The system has a number of different functions. First of all the system is used as a platform for developing computational models of syntax, semantics and discourse, and particularly as a training ground for students working in these areas. Secondly, the system contains some useful natural language applications, such as the aforementioned translator, a natural language front-end

for a database and a language-teaching tool. Finally, given that the number of people who can speak Māori fluently is decreasing, building applications which use the Māori language can increase the number of people interested in the language.

1.1.2 Architecture

In figure 1.1 the architecture of the Te Kaitito system is displayed.

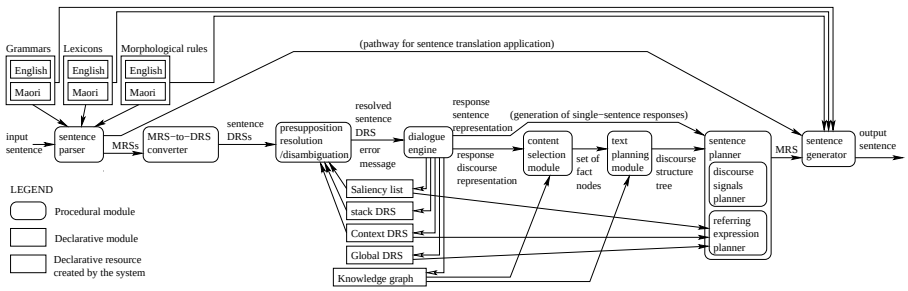


Figure 1.1: Te Kaitito architecture

When the student enters a sentence the parser generates a set of possible syntactic structures each with a semantic representation of the sentence. The parser used is the LKB system (Copestake and Flickinger, 2002) which works with HPSG-like grammars. Te Kaitito currently supports two grammars: the English Resource Grammar or ERG (A and Flickinger, 2000) and the Maori-English grammar or MEG (Bayard et al., 2002). The parser generates 'flat' semantic structures in Minimal Recursion Semantics (MRS) form (Copestake et al., 1999) which are turned into Discourse Representation Structures (DRSs) (Kamp and Reyle, 1993). The relevant aspects of MRS and DRS are described in the next section. The DRSs are then passed to the disambiguation module which disambiguates the sentence (i.e. finds referents for the anaphora and other referring expressions). Next the input's dialogue act is determined (e.g. question, assertion etc) and then the dialogue manager updates the dialogue context and decides how to respond to the user input. This process generates a response in the same MRS form as the user's input. Finally the sentence generator transforms this MRS into a sentence in natural language.

1.1.3 MRS and DRS

The first two parts of this section describe MRSs and DRSs. The last subsection describes the common ground and the structures used by Te Kaitito's dialogue manager.

Minimal Recursion Semantics

When the student enters a sentence the parser generates semantic representations in MRS form (Copestake et al., 1999). MRS is a meta-level language describing semantic structures using a flat structure.

Until recently, most parsers generated either trees or propositions in first order logic, but for various reasons many modern parsers generate underspecified semantic representations. What is left unspecified is the scope of the quantifiers in the semantic representation. For instance, the sentence "Every man loves a woman" has two possible interpretations, one in which 'every' outscopes 'a' (so that each man can love a different woman), and one in which 'a' outscopes 'every' (so that each man loves the same woman). In MRS underspecified representations are provided by representing a proposition as a simple set of relations which are not allowed to be embedded. This can be done by giving every relation in the tree a handle and using these handles to refer to the relations. Scopal relations will have handles as arguments which show the order in which the branches of the tree are to be built. An example of a sentence with scopal relations is "Every big white horse is old". The MRS form of the sentence is $\{h0 : every(x, h1, h2),\ h1 : big(x),\ h1 : white(x),\ h1 : horse(x),\ h2 : old(x)\}$. One of these relations (e.g. $h0 : every(x, h1, h2)$) is called an elementary predication and each elementary predication refers to one lexeme (an individual entry in the lexicon).

Using these handles sentences can be left underspecified. For example the ambiguous sentence "Every dog chases some white cat" has the following two MRSs:

- The reading $some(y,\ white(y) \wedge cat(y),\ every(x,\ dog(x),\ chase(x, y)))$ results in the MRS: $\{h1 : every(x, h3, h4),\ h3 : dog(x),\ h7 : white(y),\ h7 : cat(y),\ h5 : some(y, h7, h1),\ h4 : chase(x, y)\}$
- The reading $every(x,\ dog(x),\ some(y, white(y) \wedge cat(y),\ chase(x, y)))$ results in the MRS: $\{h1 : every(x, h3, h5),\ h3 : dog(x),\ h7 : white(y),\ h7 : cat(y),\ h5 : some(y, h7, h4),\ h4 : chase(x, y)\}$

The only difference between these MRSs is in the handles for the body arguments of the quantifiers. Therefore it is possible to represent both MRSs as one single MRS by representing which parts of the trees are constant and composing some constraints on how to combine those parts. The MRS which represents both MRSs looks like this: $\{h1 : every(x, h3, hA),\ h3 : dog(x),\ h7 : white(y),\ h7 : cat(y),\ h5 : some(y, h7, hB),\ h4 : chase(x, y)\}$ The only two solutions which result in a correct tree are the ones created by replacing hA by $h5$ and hB by $h4$ or by replacing hA by $h4$ and hB by $h5$.

In the aforementioned MRSs the elements like $h1 : every(x, h3, h4)$ are called elementary predications. An EP contains the following components:

- a handle which is the label of the EP
- a relation

- a list of zero or more ordinary variable arguments of the relation
- a list of zero or more handles corresponding to scopal arguments of the relation

This is written as $handle : relation(arg_1, \ldots, arg_n, scarg_1, \ldots, scarg_n)$ like the example given before.

An MRS structure is then a tuple $\langle T, L, C \rangle$ where T is a handle, L is a bag of EPs and C is a bag of handle constraints. An example of an MRS is $\langle h0, \{h1 : every(x, h3, h4), h3 : dog(x), h7 : white(y), h7 : cat(y), h5 : some(y, h7, h1), h4 : chase(x, y)\}, \{\}\rangle$. In this example the bag of handle constraints is empty because there is exactly one possible reading. If an MRS is used like the one with the handles hA and hB the bag contains the constraints on how to link those handles to other handles in the MRS. It goes too much into detail to describe the handle constraints, but it is all described in (Copestake et al., 1999).

In order to understand everything in this report it is necessary to be familiar with the following terminology. An elementary predication is called a *relation*, the relation's name is called the *predicate* and all of the arguments are called *referents*. There are different types of referents of which the most important ones are the following:

- e-vars refer to so-called 'events', treating them as objects. Every proposition has a 'main' event which is referred to by an event variable of this kind.
- x-vars refer to entities like dogs, people, houses etc.
- h-vars refer to handles and are used to represent the scopal relationships between the different relations.

Furthermore every relation has a handle which shows where the relation is to be embedded in the corresponding tree structure. Associated relations (relations which belong to the same node in the tree) have the same handle. In the sentence "The green dog barked" the relations 'green' and 'dog' belong to the same node and therefore they have the same handle.

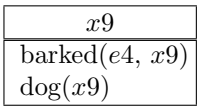
Discourse Representation Structures

After generating an MRS it is transformed into a DRS (Kamp and Reyle, 1993). A DRS is a structure with the following two fields:

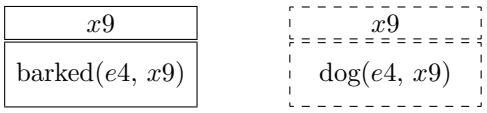
- a list of referents: the entities which have been introduced into the context
- a list of conditions: the relations which are known to hold of these referents.

Each of the referents in the 'referents' field represents a unique entity; if there are several dogs in the current context each of the dogs has a different referent.

DRSs can be drawn as split boxes with the referents at the top and the conditions below. An example for the sentence "A dog barked" is the following:



This sentence contains the indefinite determiner 'a' which means that a new entity is introduced. However, sentences often contain entities which are already in the context. For example, the sentence "The dog barked" presupposes there is a dog in the current discourse context. This presupposition can also be represented as a DRS. A sentence consists then of a single assertion DRS and a *set* of presupposition DRSs. The assertion DRS is represented as the one in the previous example and the presupposition DRSs are represented as dashed boxes. The previous example has no presuppositions and therefore the DRS consists of a single assertion DRS and an empty set of presupposition DRSs. The sentence "*The* dog barked" does have a presupposition DRS and the whole sentence is represented as:



Common ground and update structures

Finally the DRS is stored in the common ground which contains all facts uttered in the dialogue. The common ground is represented by one large DRS, also containing a list of referents and a list of conditions. The referents are all entities ever mentioned by the user or the system and the conditions contain the other information that could be derived from the utterances in the dialogue. The common ground can be seen as the union of all DRSs corresponding to these utterances. When the user asks a question the system will look at the common ground and will try to answer the question using the relations stored there. In appendix A an example of a common ground is given containing the two sentences "A dog chased a cat" and "The cat was afraid".

The dialogue manager which updates the dialogue context does not use MRSs or DRSs, but uses *updates* instead. These are similar structures, but they include somewhat more information. An update structure consists of the following elements: the act of the update (e.g. assertion or question), the speaker, the addressee, the message and a list of bindings for each of the referents used in the MRS. The message can be either a proposition or a question. A proposition structure is similar to a DRS structure: the nucleus is the asserted part of the assertion and the presups are the presuppositions of the assertion. A question looks like a proposition except that it has a (possibly empty) set of parameters. Examples of a number of propositions and a question are given in appendix A. In these examples you can see that an entity using the determiner 'a' is stored in the nucleus and an entity using the determiner 'the' is stored in a presup.

1.2 Second language learning and teaching

There are many different ways to learn a second language. First of all language learning can be done naturally, via 'immersion', for example by moving to a country where the target language is spoken. In this way the learner is exposed to the language most of the day. Another way is following lessons in a classroom setting which can be subdivided into traditional instructional settings and communicative instructional settings. The traditional instructional setting is used at most schools and the teacher focuses mainly on grammar and vocabulary. In this setting the focus is on the language itself, rather than on the information which is carried by the language. The goal of the learners is most often passing an exam instead of using the language in everyday life. In the communicative instructional setting the learners also follow lessons, but those lessons focus more on interaction, conversation and language use than on learning *about* the language. The goal of the learners in this setting is to get things done using the second language rather than accuracy.

There are many differences between the two ways of teaching a second language which are described by (Lightbown and Spada, 1993). The most important difference is that the traditional way focuses on language and its form and the communicative setting focuses on meaning. Another difference is the teacher's questioning behavior which is described by (Shomoossi, 2004). She studied the effect of teachers' questioning behavior in a *traditional* instructional setting. First of all she distinguishes between display questions and referential questions. Display questions are questions to check if the student understands everything that has been said in the classroom. This kind of questions is seldom used in a normal conversation, but can be very useful when learning a second language. Referential questions are questions whose answers are not already known to the teacher.

The most important results of her study in the traditional instructional setting were the following. First of all, almost 82% of all questions were display questions and only 18% were referential questions. Furthermore referential questions elicited long answers (up to five minutes) compared to display questions which could often be answered in only a few seconds. This does not apply to all referential questions, so she reaches the conclusion that "most, not all, referential questions create more interaction in the classroom than display questions do". In a communicative instructional setting the results would be completely different. In such a setting there would be many more referential questions and less display questions, which has been confirmed by (Lightbown and Spada, 1993).

1.3 Dialogue based teaching systems

There are many existing dialogue-based teaching systems. The SCHOLAR system is often considered to be the first intelligent tutoring system (Carbonell, 1970). The system was intended for learning South American geography in the

form of a dialogue. The system did not try to create a coherent dialogue; its main goal was to assist the student while he was learning the geography.

Some other systems include CIRCSIM-Tutor, AutoTutor and LISTEN which will be described in this section, but first the different kinds of initiative taking that systems can support are explained.

1.3.1 Levels of initiative taking

In a dialogue-based teaching system there is a conversation between a user and the system. Guinn distinguishes between four different levels of initiative taking (Guinn, 1996). First of all there are simple question answering systems in which one agent is in control and the other agent is just answering the questions. The agent who is in control can be either the user in an information system or the system itself, for example in a booking system used in a theatre. There are also systems which allow some kind of mixed-initiative but in which the user can take the initiative only at limited places. Finally there are systems which allow a complete mixed-initiative, so systems in which the system and the user can both take the initiative when they like to do so.

1.3.2 CIRCSIM-Tutor

The CIRCSIM-Tutor system (Freedman, 1997) teaches medical students on the baroreceptor reflex using natural language. It is a mixed-initiative dialogue system which means that the system and the user can both ask questions. Before a lesson starts the student tries to solve a problem by filling in a table with predictions of changes in a number of physiological variables. During the lesson the system has a dialogue with the student to correct his mistakes. In this dialogue the system makes sure that every initiative it takes is coherent with the rest of the conversation.

If the student answers a question, the answer is classified into one of four different types: correct answer, wrong answer, physiological near-miss (a step toward the correct answer) and linguistic near-miss (linguistically close but not exact answer). Depending on the correctness of the answer the system chooses its next initiative.

If the student makes an assertion there are two possibilities: the student adds new information or the student changes the topic. In order to avoid a sudden topic change by the user the system makes sure that every turn ends with an explicit request. Finally Freedman notes that completely unrestricted student initiatives can be too complex to handle. Therefore she came up with a way to reduce the unwanted student initiatives which consists of asking short-answer questions instead of open-ended questions.

1.3.3 AutoTutor

AutoTutor[1] (Graesser et al., 2001) is an animated web-based intelligent tutoring system which can help students studying by having a conversation in natural language about a particular subject. At the moment AutoTutor can assist students who take an introductory course into computer science or students who have to learn Newtonian physics. AutoTutor can however be adapted to support different subjects.

The system works by having a conversation with the student and tries to create a coherent dialogue. If the system has to change the topic the system first presents a discourse marker like "Alright, let's go on" and before asking the actual question it presents a context to introduce this question.

If the student answers a question only partly correctly, the system will not tell the correct answer immediately, but will instead initiate a multi-turn conversation to extract more information from the student. Thus, the system tries to get the student to do the talking and tries to find out what the student knows himself, instead of just *telling* the information. The system encourages the student to talk more by using sentences like "What else?". If the student doesn't know what to say the system will try to help him by using one of the following dialogue moves: hints, prompts and assertions. Hints are questions used to lead the student into the right direction. Prompts are sentences with a word left out that the student is supposed to fill in. Finally assertions directly tell the correct information.

AutoTutor gives the student three different kinds of feedback: backchannel feedback, pedagogical feedback and corrective feedback. Backchannel feedback is the feedback that the system gives when the student is entering new input, e.g. nodding. Pedagogical feedback consists of different intonations and facial expressions based on the correctness of the student's input (e.g. saying "okay" at a moderate nod rate when the answer is only partly correct or saying "okay" with a fast head nod when the answer is completely right). Finally corrective feedback is feedback that has to repair bugs and misconceptions that students make.

The system mainly asks questions itself, but it also supports some kind of mixed-initiative. In order to support this the system classifies the student's input into different categories: assertions, wh-questions, yes-no questions, metacognitive comment (e.g. "I don't understand"), metacommunicative act (e.g. "Could you repeat that?") and short response. The system makes sure that the dialogue move taken by the system corresponds to the student's previous input.

Finally AutoTutor has curriculum scripts for 36 different topics. Associated with each topic are a set of expectations, a set of hints and a set of possible misconceptions and their corrections. The descriptions are English descriptions and can easily be created by a lesson planner. The ultimate goal of the system is to make it very easy to add new subjects and extend the system's domain knowledge.

[1] see also `http://www.autotutor.org`

1.3.4 LISTEN

The LISTEN[2] (Literacy Innovation that Speech Technology ENables) project (Mostow et al., 2004) is a system to measure a student's reading comprehension and vocabulary. The system consists of an automated Reading Tutor that displays stories on a screen and listens to children read aloud. The system analyses a text and generates questions about the text automatically. The system generates two different kinds of questions, one of which is used to estimate reading comprehension and the other is used to estimate the student's vocabulary.

The questions used to measure reading comprehension are fill-in questions: one word in a sentence is left out and the student has to decide which word should be filled in. It is often too hard to guess a word just from its context, so the questions are multiple-choice. This means that the system not only has to decide which words to leave out, but the system also has to generate the alternative answers (called distractors) itself.

The system distinguishes four different word categories:

- sight words: the most frequent words
- easy words: words which appear often, but not as often as the sight words
- hard words: words which don't appear often, so all words except the sight words and the easy words
- defined words: words explicitly annotated for the particular story

While generating the questions it is made sure that the target word and the distractors are in the same category.

The question's difficulty depends on a number of different aspects. First of all the difficulty depends on the category in which the target word and distractors appear. It also depends on the text difficulty, because harder texts contain more complex sentences and harder words. The third aspect is part of speech, because questions with a number of answers which have the *same* part of speech are more difficult than questions with answers with different parts of speech.

The reading comprehension is determined by looking at the number of correct answers and by deciding if the question was an easy or a difficult question. An example question is "Why bother about ... ?" – food / winter / dying / passed. In this example the meanings of the different words are easy, but only one word can be ruled out based on its part of speech. Three possibilities remain and the only way to find the correct answer is by looking at the context.

The LISTEN system also generates vocabulary questions to test students' comprehension of individual words in a story. These questions are also multiple-choice and consist of a description of a particular word and four options. An example question is "Which word means *heavy flows of water?*" – eider-down / bedstead / torrents / scarcely. The distractors are chosen from the same story and the comprehension is determined in the same way as the reading comprehension.

[2]see also `http://www-2.cs.cmu.edu/~listen/`

1.4 Project goals

In the previous section a number of existing computer-aided teaching systems have been described. CIRCSIM-Tutor and AutoTutor are dialogue systems which use full natural language processing to communicate with the student. The systems focus on completely different subjects (the baroreceptor reflex and computer science or Newtonian physics respectively), but both systems try to create a coherent dialogue. The LISTEN system on the other hand does not have a dialogue with the student like the other systems, but is used for language learning.

In this project I want to combine these things; the goal is to use a *dialogue* system that can assist people learning a second *language*. In this context, language is not only the medium of instruction, but also the topic being taught. There are some specific benefits that arise from this situation. The student's utterances have to be parsed to find out the meaning as part of the normal operation of the dialogue system. As a side-effect of parsing, the system gathers information about which syntactic rules and words the student knows, which make a useful contribution to modelling the student's knowledge of the topic being taught.

Within the context of a language-learning dialogue, my focus will be on developing a module that generates useful teaching initiatives. The initiatives have to fulfill two functions. The first is linguistic; the initiatives have to be on topic. Therefore an additional project goal is to find the possible topics that a new initiative can be about. The second is goal-based; the initiatives have to further the system's educational agenda and help the student learn the language.

Chapter 2

Design of the initiative generation algorithm

In this chapter the design of the initiative module is described. In the first section the overview of the algorithm is given. In section 2.2 is described how the system has been modified in order to deal with new tasks like generating assertions and genuine questions. Finally section 2.3 describes the different steps of the algorithm in more detail.

2.1 Overview of the algorithm

When the student is having a normal conversation with the system the initiative module is completely ignored. The student's input is parsed in the normal way and the system also responds in the normal way. When the user decides to pass the initiative to the system and presses the Enter key without other input, the initiative module is invoked.

The overview of the algorithm looks like the following:

1. Find the possible topics
2. Generate a list with possible initiatives
3. Choose the best alternative
4. Add necessary discourse markers
5. Process the initiative and the user's response

2.2 Authoring characters

In order to generate assertions and genuine questions we need to create a 'character' for the system to adopt during its dialogue. A character knows various

things (which it can tell the user), and is curious about various things (about which it can ask the user questions). The characters have an initial private knowledge base which contains information that the character can use to create assertions. This information includes personal details like their names and their ages, as well as simple facts like "The grass is green". A knowledge base looks exactly the same as a common ground, but is private to each character.

Furthermore the characters contain a set of rules to create genuine questions. Those rules are stored in a way which looks mostly like this:

- if the topic is a dog, ask for its name
- if the topic is a dog, ask for its color
- if the topic is a person, ask for his name
- is the topic is a person, ask where he's from
- etc.

Those rules can be applied to entities the user starts talking about. For example the first one of the rules can be used when the user enters the sentence "I have a dog".

Finally each character also has a list of targeted syntactic rules. The system is meant to assist people who want to learn a second language and therefore the characters can be seen as lessons. The character corresponding to the first lesson will only have a very limited knowledge base, e.g. he only knows how to say his name and how to ask how someone else is. The next character will have more knowledge and will have somewhat more complicated questions; the next character will have even more knowledge and so on. The lessons follow the structure of the course book "Te whanake 1: Te kākano" (see (Moorfield, 1988)) which means that new syntactic constructions are learnt in every lesson. In order to measure the student's comprehension the system has to test if the student knows all the targeted syntactic constructions. In order to achieve this a list is kept which contains the set of syntactic rules targeted in a particular lesson together with values representing the extent to which they have been assimilated by the student. Every time the system or the user uses one of the rules the corresponding value will be updated. A more detailed description of the syntactic rules and updating of the values is given in section 3.5.

The lessons are mainly based on the book, but it is still important that someone who knows about second language learning authors the lessons, for example a second language teacher. Most second language teachers have limited knowledge about using computers and writing computer programs, so it is important that the authors can build the characters using natural language. In order to do this Te Kaitito has two different modes: the student mode and the authoring mode. In student mode the student just follows a lesson by having a dialogue with the system. In authoring mode the author can create the different characters.

In authoring mode the author uses the same dialogue system as the student does in student mode, but the way the input is processed is somewhat different and there are some additional functions. The remainder of this section describes how the different elements of the characters are created.

The knowledge base is created by storing the author's sentences in the exact same way as a student's input is stored in the common ground. When the student starts a lesson the character's common ground is copied to the character's knowledge base and the common ground is reset to empty. In this way the character has an initial private knowledge base from which to create assertions and the common ground is empty which means that nothing has been said yet in the current dialogue.

If the author wants to add rules to generate genuine questions he can simply enter a sentence like: "rule: what is a dog's name?". The first word 'rule' is used to tell the system that the question following the colon is not a question which has to be answered, but that it is a question that needs to be stored in the list of question generation rules. The system recognizes that the question is about a dog and stores the question with the corresponding topic in the list of rules to generate genuine questions.

Finally the list of targeted syntactic rules is created automatically. Every time the author enters a sentence the sentence is parsed. After parsing the sentence the system checks which syntactic rules were used. When the author is authoring the first lesson the system simply stores all rules in the list of syntactic rules, because at that time the student has to learn all of those rules. When the author is authoring a later lesson the system should not only determine which rules are used, but it should also check which of those rules are new. Rules which are used in earlier lessons don't have to be added again, assuming the student will follow the lessons in the correct order and will only move on to the next lesson when he has finished the last one successfully. Right now the system simply stores all rules in the list of target rules, because we are only implementing the very first lesson, but comparing the rules with earlier lessons would be a useful addition.

Because the list of syntactic rules is created automatically the author doesn't have to worry about the names of the different rules, but he does have to make sure that the sentences he enters contain the correct syntactic structures. Learning a second language in a classroom setting occurs in a step-by-step way; in the first lesson only a few rules are used, in the next lesson some new rules are added etc. The author thus has to make sure that the lessons are built in a way that satisfies this condition.

2.3 Steps of the algorithm

In this section each of the different steps of the algorithm from section 2.1 is described in more detail.

2.3.1 Finding the topics

When two people are having a conversation there always is a topic. This topic can change over time, but at any given time there will always be a main topic. In order to make sure that the constructed dialogue is coherent the system's initiative should be on topic, at least if possible.

In the past many detailed algorithms for finding the topic have been designed. One of the best known algorithms is the Centering algorithm by Grosz (Grosz et al., 1995). Another example is the complicated algorithm by Nakata which uses several other methods like k-nearest neigbor and topics clustering (Nakata et al., 2002). The main problem of these algorithms is that they are too complex for a three-month project like this. Another problem is that most of the existing algorithms return only one topic. For this project returning only one topic is not sufficient, because it is obviously better when the initiative has more than one topic in common with the list of possible topics than when the initiative only has the best topic in common. Therefore a new algorithm for finding a list of possible topics has to be designed.

This algorithm is based on the Centering algorithm by Grosz (see (Grosz et al., 1995) and (Jurafsky and Martin, 2000)). In the Centering algorithm every sentence has a backward looking center and several forward looking centers. The backward looking center represents the entity currently being focused on after the sentence is interpreted. The forward looking centers form an ordered list containing the entities mentioned in the sentence, in other words the entities which can serve as the backward looking center of the following sentence. The forward looking centers are ordered based on their grammatical role in the sentence.

The Centering algorithm was originally designed for normal texts instead of dialogues. The main difference between dialogues and other texts is that the sentences in dialogues are normally much shorter. Furthermore every word usually appears only once in the sentence so it is hard to apply a statistical method (Nakata et al., 2002).

The algorithm that is used in this project orders the words in the sentence on grammatical position in the following order: event - subject - object - indirect object. Event represents the focus of the whole sentence; this can be an action verb like 'chase' in the sentence "The dog chased the cat", but it can also be an adjective like 'afraid' in the sentence "The cat was afraid". The algorithm returns an ordered list of possible topics in a sentence, with the first element being the best topic. Such a list is created for the last two sentences and afterwards the two lists are combined as will be described in Section 3.1.2.

2.3.2 Generating possible initiatives

There are three different kinds of initiatives which will be generated: genuine questions, assertions and teaching questions.

Genuine questions are equal to the referential questions used by (Lightbown and Spada, 1993), so these are questions whose answers are not yet known to

the system. The questions are questions about one main topic like a dog, a house or a person. The questions are generated using the question generation rules stored for each character. Before adding the question to the list of possible initiatives it has to be checked that the answer is not yet in the common ground.

Assertions are just facts uttered by the system. They are created by combining some relations stored in the character's private knowledge base. Again it has to be checked that the fact being asserted is not yet in the common ground.

Teaching questions, or display questions (Lightbown and Spada, 1993), are questions whose answers are known to the system. They are used to check if the student understands everything that has been said so far. The questions can be generated by constructing a proposition from the relations stored in the common ground and turning that proposition into a question.

2.3.3 Choosing the best alternative

After the possible initiatives have been generated the system has to decide which initiative is best. This decision is based on the following criteria:

1. Focus: is the initiative 'on topic'?

   It is important that the constructed dialogue is coherent, so every initiative taken by the system should be on one of the current topics, at least if possible. The focus criterion consists of a number of different characteristics to measure how much the initiative on topic is. These characteristics will be described in section 3.3.1.

2. Order of acquisition: in which order are the question generation rules entered?

   Assuming that the question generation rules are ordered in the most logical way, initiatives based on rules that were entered first are probably better than questions based on rules entered later.

3. Semantics: does answering the question require reasoning?

   Some teaching questions are more difficult than other questions. For example open-ended questions are often more difficult than multiple-choice questions and questions which involve related entities are also more difficult. In some contexts easy questions might be preferred; in others harder questions. This criterion makes sure that the different teaching questions can be evaluated in relation to their difficulty.

4. Strategy: does the initiative make a good initiative possible later?

   Assertions about topics the system knows much about are useful initiatives, because they ensure that there will be other useful initiatives later in the conversation. The strategy criterion makes sure that those kinds of assertions are preferred to other assertions.

5. Balance: are the different kinds of initiatives balanced?

   It is best if the system generates the different kinds of initiatives in a balanced way. This means that the system should not generate too many initiatives of the same kind in a row. The criterion makes sure that initiatives of the kind which hasn't been used for a long time are more likely to be chosen than initiatives of the other kinds.

6. Syntax: does the initiative contain (or elicit) a targeted syntactic construction?

   Because the system is used to assist people learning a second language some syntactic rules are preferred to other rules, depending on the lesson the student is following. The syntax criterion makes sure that initiatives which contain or elicit particular syntactic constructions are preferred to initiatives which don't contain or elicit such constructions.

The first five of these criteria contribute to the goal at utterance level. They make sure that the chosen initiative is appropriate in the context and that the initiative scores high in at least one of the other criteria. The last criterion is used to pursue the global goal; to teach the student different syntactic rules in such a way that the student can use the rules himself.

The possible initiatives get scores for some of the criteria and afterwards a 'total score' is calculated by applying a function to combine the different scores. Finally the initiative with the highest total score will be chosen.

2.3.4 Adding necessary discourse markers

Using discourse markers can be very helpful in a dialogue-based teaching environment. This is studied by Kim for the CIRCSIM-Tutor project (Kim et al., 2000). However, the main goal of the initiative module is not to add these discourse markers, so we will only add the discourse marker 'also'.

The system will add the discourse marker in two different cases. The first one applies when there is only one word different. This word can be of any word type (a verb, a noun etc) under the condition that the different words are of the same type. The following examples will therefore include the word 'also':

- Verb: The dog barked → The dog also ran
- Noun: The dog ran → The cat also ran
- Adjective: The dog is afraid → The dog is also sad

The second situation in which the word 'also' is added applies when the words are exactly the same, but when two arguments are swapped. These arguments can be any two of the arguments as the following examples show:

- John introduced Pete to Bill → John also introduced Bill to Pete
- John introduced Pete to Bill → Pete also introduced John to Bill
- John introduced Pete to Bill → Bill also introduced Pete to Bill

These three examples all require the word 'also' while the sentence "Pete introduced Bill to John" doesn't need the word 'also'.

In order to find out if one of these situations occurs, so to find out if the word 'also' should be added to the chosen initiative, an algorithm is used which will be described in more detail in section 3.4.1.

Apart from adding the discourse marker 'also' every teaching question is preceded by a sentence like "Let me make sure that you remember:". This is done to make sure that the user knows whether the question is a teaching question or a genuine question. For example if the student says "The dog jumped" and the system generates the question "Did the dog also bark?", it could be unclear if the question is a teaching question or a genuine question. To make sure that the student knows which kind of question is asked the clarification sentence is used. This is somewhat similar to the way Autotutor presents a context to introduce a question.

2.3.5 Processing initiative and user response

The system can generate three different kinds of initiatives. Each of these kinds is processed in a different way which is described in this section.

Genuine questions are used to get new information from the user. Therefore the user's next input will be the answer to the question. Currently, this answer is simply interpreted as a new assertion and the answer is stored in the common ground as if it were a new assertion.

Assertions are just facts from the knowledge base, so they have to be copied to the common ground after the assertion is made. Assertions don't necessarily evoke a reaction by the user, so the user's next input is seen as a normal input and the dialogue continues normally.

Finally teaching questions are based on the common ground, so they do not change the common ground. The question does however require an answer, so the next input by the user is seen as the answer to the teaching question and has to be checked. If the answer is correct the system responds with a 'correct' message and the dialogue is continued in the normal way. If the answer is incorrect the system will repeat the question to give the student a second chance.

Chapter 3

Implementation

In this chapter the implementation of the initiative module is described. Sections 3.1 to section 3.5 describe the different steps of the algorithm given in section 2.1. The generation of the different initiatives can be seen as building a large AI-style state-space tree. Comparing the different possibilities in order to choose the best one can then be seen as searching the tree. If the character played by the system has a large knowledge base and many question generation rules, this can result in a very large tree. Therefore pruning is necessary which is described as part of choosing the best initiative. In the first description of generating the possible initiatives pruning will be completely ignored for clarity.

Section 3.6 is about how to decide if the student has learnt the rules in the lesson well enough, so about how to decide when to stop the lesson. In the algorithm many parameters are used. In the following sections these parameters are only given a name, but in section 3.7 some notes are given on assigning values to these parameters.

During the implementation I will use the ERG (A and Flickinger, 2000), rather than the MEG, so the system will function as a tool for teaching English, rather than Māori. However, the implementation should be portable to the MEG without many changes.

3.1 Finding the topics

As mentioned in section 1.1.3 one single entity is represented by a unique referent. Every entity also has a predicate which shows the kind of entity. For example dog($x9$) is a different entity than dog($x12$), but they have the same predicate 'dog' because they both refer to dogs.

To compare different entities it is best if the referents are the same, because then the system is talking about the exact same entity. On the other hand it is not enough only to look at the referents, because it is still better to have the predicate in common than having nothing in common. For example if a student is talking about a dog and the system doesn't know anything about that dog

in particular, but does know something about another dog then the initiative containing the new dog might be a good initiative. Therefore the list of possible topics will contain referents as well as predicates. The referents will be stored in the list before the predicates to make sure that the list is completely ordered - the first element is most likely the best topic and the last element is least likely the best topic.

The algorithm used in this project looks at the previous two sentences, but if necessary the algorithm can easily be extended to include more sentences.

3.1.1 Finding the topics in one sentence

Appendix A shows some examples of sentences represented as propositions in the system. The second example is "The dog chased the cat" and the third example is "The cat was afraid". As you can see in the figure the most important word in the first one is 'chased' and the most important word in the second one is 'afraid'. These words are the most important words, because their first argument is an e-var, referring to the main event of the proposition, and they are stored in the nucleus. The relations have arguments as explained in section 1.1.3. The x-vars refer to the entities and each x-var refers to one of the presups of the proposition. The first of the x-arguments is always the subject, so this can be used to order the list of possible topics by grammatical position (as in the Centering algorithm by Grosz).

The algorithm to find an ordered list of topics in one sentence looks like the following:

```
1. take the predicate(s) in the nucleus
2. add the arguments
       while adding the arguments:
             add the corresponding predicates
```

The results of this algorithm for the two example sentences are (chase $x9$ dog $x12$ cat) and (afraid $x12$ cat).

The algorithm only takes the main predicates from the sentence; in other words the algorithm takes only one predicate per presup. If the sentence was "The green dog barked", the word 'green' is skipped because otherwise all initiatives on green entities would seem good initiatives. An example of this is the sentence "The pen contains green ink" which is obviously not a good initiative. Every relation includes the part of speech in its name; e.g. the name of the relation which represents a dog is 'dog_n_rel' and the name of the relation which represents green is 'green_j_rel'. This name can be used to make sure that nouns are always added and that an adjective is only added when it is the main event of the proposition.

3.1.2 Combining the topics lists of two sentences

As mentioned in section 2.3.1 the list of possible topics is based on the last two sentences, so an algorithm has been designed to combine two lists of topics. The result of this algorithm is the union of the two lists, with all topics which appear

in both lists moved to the front. In this way the final list is completely ordered by relevance - the first element is most likely the best topic and the last element is least likely the best topic.

The algorithm for combining the two lists looks like this:

```
1. Find the topics of the previous sentence
2. Find the topics of the sentence before the previous sentence
3. Assign the result of step 2 to possible-topics
4. Loop through the elements in the topics-previous-sentence
   list and do the following:
        if the element is a member of possible-topics:
           add the element to a temporary variable (tmp1)
           and remove the element from possible-topics
        else:
           add the element to a second temporary variable (tmp2)
5. Append the tmp1, tmp2 and possible-topics and store the
   result in possible-topics
```

Before executing step 5 of the algorithm $tmp1$ contains the topics which appear in both lists in the same order as they appeared in the previous sentence. $tmp2$ contains the topics which appear in the previous sentence, but don't appear in the sentence before. Possible-topics contains the topics which appear in the second last sentence but not in the last sentence. By appending those lists the desired result is obtained.

3.1.3 Example of execution of the algorithm

In this example the same sentences are used as in the examples before: "The dog chased the cat" followed by "The cat was afraid". The topics lists of the sentences are (chase $x9$ dog $x12$ cat) and (afraid $x12$ cat). Execution of the algorithm for combining those lists results in the following:

```
1. (afraid x12 cat)
2. (chase x9 dog x12 cat)
3. possible-topics = (chase x9 dog x12 cat)
4. tmp1         tmp2          possible-topics
   nil          (afraid)      (chase x9 dog x12 cat)
   (x12)        (afraid)      (chase x9 dog cat)
   (x12 cat)    (afraid)      (chase x9 dog)
5. possible-topics = (x12 cat afraid chase x9 dog)
```

The result of the algorithm ($x12$ cat afraid chase $x9$ dog) means that the referent $x12$ is probably the best topic and that the predicate 'dog' is least likely the best topic.

3.2 Generating initiatives

In the following subsections the generation of the different kinds of initiatives is described. In these sections it is assumed that all initiatives are generated. In fact only a number of initiatives is generated and in section 3.3.4 is described how is decided which initiatives are generated.

3.2.1 Generating genuine questions

As described in section 2.2 every character has a set of question generation rules. These rules are represented by a structure which contains a topic and an MRS. The topic is the predicate which the question is about and the MRS is the actual question. An algorithm is used to find out about which predicate the question is and this predicate is stored as the topic of the question. If a genuine question has to be generated the referring expression of the predicate is replaced by the referring expression of the new entity. An example of this is the question generation rule "What is a dog's name?" with the topic 'dog'. If there is a green dog in the context, the question "What is the green dog's name?" is generated.

There is also a special predicate "person" which can be replaced by any kind of person. In the current system it is not possible to find out that a man is a particular kind of person, but once this is added to the system, it can be very useful.

The algorithm for generating all possible genuine questions loops through the referents in the list of possible topics, creates a referring expression for each of the referents and applies all possible question generation rules to each referent. Only referents in the list of possible topics are taken to make sure that the genuine question is on topic. Before adding the question to the list of possible initiatives it has to be checked that the answer to the question is not in the common ground yet. The current Te Kaitito system cannot answer the question "What is the dog's color?" when the sentence "The dog is green" or the sentence "The dog's color is green" is stored in the common ground. Currently a list is kept for each genuine question containing the referents which the question has already been applied to. If the system is not able to answer the question this list can be used to find out if the question has already been asked. This way all genuine questions are asked only once for each referent.

Furthermore only questions about referents entered by the user are generated. This is done to avoid questions like "What is the dog's name" if the dog was entered by the system, because the user would probably not know the answer.

3.2.2 Generating assertions

Assertions are generated based on the character's knowledge base. At the start of this project the characters only had a common ground. In order to distinguish between facts known only to the system and facts known to both the system and the student, a private knowledge base had to be added to the different characters. Appendix A shows an example common ground containing the two sentences "The dog chased the cat" and "The cat was afraid". A knowledge base containing those two sentences would look exactly the same.

To generate an assertion from an unordered and possibly large set of relations we first need to find a set of relations which correspond to an utterable proposition. To do this, one can take a relation whose first argument is an *e*-var

(representing the main event of the proposition) and create a nucleus containing this relation. Some propositions have two main events with the same handle, so it has to be checked if there is another relation which has the same top handle. If that is the case both relations are turned into a nucleus, otherwise only one relation is turned into a nucleus.

When we have selected a proposition to create, we have to decide how to refer to the entities it contains. The referring expressions originally used by the author when populating the knowledge base may no longer be suitable, as referring expressions depend heavily on the context in which they are uttered. Te Kaitito already had a referring expression generation module. However, this had to be modified for the situation where an utterance contained entities which the user had not encountered before, because in that case the referring expression had to be created based on the relations stored in the knowledge base rather than the common ground. Consequently, it is made sure that an assertion about a new referent uses the determiner 'a' and an assertion about an already known referent uses the determiner 'the'. This can be done by checking if the referent appears only in the knowledge base or both in the knowledge base and in the common ground. When the referring expression contains the determiner 'a' a new referent is introduced. In that case the generated referring expression is added to the nucleus of the proposition to make sure that the new relations are stored in the common ground. On the other hand, if the referring expression contains the determiner 'the' the assertion presupposes the entity is already in the context and therefore the referring expression will be stored as a presup of the proposition.

Furthermore, it is assumed that the objects which the system knows about do not overlap with those which the user knows about. Consequently, we ensure that the system never makes an assertion about a referent entered by the user. The system can however make different assertions about the same entity, but only if that entity was entered by the system itself.

One final thing to note is that the algorithm finds all possible adjectives associated with a referent. If the system wants to say something about a dog and it knows that the dog is green it will add this information. This is done to decrease the chance that there are two different referents with the same predicate which are not distinguishable, for example two different dogs without further information. If there are two indistinguishable entities the system might create confusing teaching questions, so adding the adjectives minimizes the chance that this occurs.

3.2.3 Generating teaching questions

Teaching questions are created using the relations in the common ground. First the relations are turned into all possible propositions in the exact same way as the assertions are generated. After generating these propositions, the system generates all possible questions about each proposition.

There are two different kinds of questions: wh-questions and yes/no-questions. Wh-questions are questions which contain one of the words which, what, who

or where. Officially questions which contain the word 'how' or the word 'why' are also wh-questions, but the current system doesn't support sentences which use these words, so we will only look at questions which contain one of the first four words. Yes/no-questions are questions whose answer is simply 'yes' or 'no'. They can be seen as questions which ask whether the propositional part of the question is either true or false. Wh-questions do therefore have a parameter (e.g. "which dog" or "what"), but yes/no-questions don't.

Because the propositions are based on the common ground, all arguments of the main event are already in the context, so the referring expressions for the arguments appear in the presups of the proposition. In order to generate all wh-questions the algorithm loops through these presups and turns them into a parameter one at a time. In this way the proposition "The dog chased the cat" will be turned into the questions "What chased the cat?" and "The dog chased what?". If there is more than one entity with a particular predicate which can be distinguished a "which-X" question is generated instead of a "what" question. For example if there is only one dog the question "What chased the cat?" is created, but if there is more than one dog the question "Which dog chased the cat?" is generated.

Yes-questions are created by simply turning the proposition into a question without any parameters. An example is "Did the dog chase the cat?".

No-questions are harder to generate, because they are not literally stored in the common ground. The current system doesn't have a way to find out if an entity is an animate object or not. For example the sentence "The garden who was happy barked" is simply parsed and stored in the common ground. Instead every referent has a gender feature which is set for dogs and cats, but is not for objects like gardens. Therefore the algorithm uses this to determine if the referent is an animate object or not. The algorithm first checks if the subject of the sentence is an 'animate' object and if that's the case the referring expression corresponding to the subject is replaced by the referring expressions corresponding to all other 'animate' objects in the list of possible topics. The referents are only replaced by referents which are on topic, because otherwise too many no-questions are generated. Furthermore the referents for the student and the system are never replaced by any other referents, because this can result in rather strange initiatives. Finally, referents are never replaced by any referents which appear somewhere else in the initiative, because this also creates strange initiatives. An example of this is the no-question "Did the cat chase the cat?" after entering the sentence "The dog chased the cat".

Finally a list is kept which contains the teaching questions which have already been asked. This list contains (e-var, parameter)-pairs; the e-var is the event variable representing the proposition's main event and the parameter is the number of the parameter if the question is a wh-question, and the string "yes" or "no" if the question is a yes/no-question. Before the generated question is added to the list of possible initiatives it is checked if the question has not been asked yet. This is done to prevent the system from generating the same question repeatedly. This way a proposition is turned into all possible questions, but all questions are asked only once. For example, the proposition "The dog

chased the cat" will be turned into the following questions: "What chased the cat?", "The dog chased what?", "Did the dog chase the cat?" and "Did the man chase the cat?" if there is a 'man' in the list of possible topics. For each of these questions only one sentence in natural language will be generated.

3.2.4 An example of the generated initiatives

In the previous subsections has been described how the different kinds of initiatives are generated. As already mentioned the generation of the possible initiatives is like building an AI-style state-space tree. Figure 3.1 shows an example of the generated initiatives using only a simple character. The character has question generation rules to create the following genuine questions:

- What is a dog's name'?
- What is a dog's color?
- What is a cat's name?

Furthermore the character's knowledge base contains relations to create the following assertions:

- I have a green dog
- The green dog is happy

The lesson has already been started and the user has entered the following two sentences:

- A dog chased a cat
- The cat was afraid

When the initiative module is called at this moment the search tree in figure 3.1 is generated. This situation is one of the simplest situations which can occur and the search tree is already rather large. Imagine how large the tree will be when the character has much more knowledge and the dialogue has been going on for some time! In section 3.3.5 is described how the efficiency of the algorithm can be improved.

3.3 Choosing the best alternative

In order to choose the best initiative all possible initiatives will get scores for some of the criteria in section 2.3.3. Some of the scores are assigned during the generation step of the algorithm, but for clarity all scores are described in this section. After assigning the scores they have to be scaled and combined which will be described at the end of this section. In this section many parameters are introduced, but for the moment we will only use their names. To make clear which parameters are used their names start and end with an asterisk and in section 3.7 the real values are given.

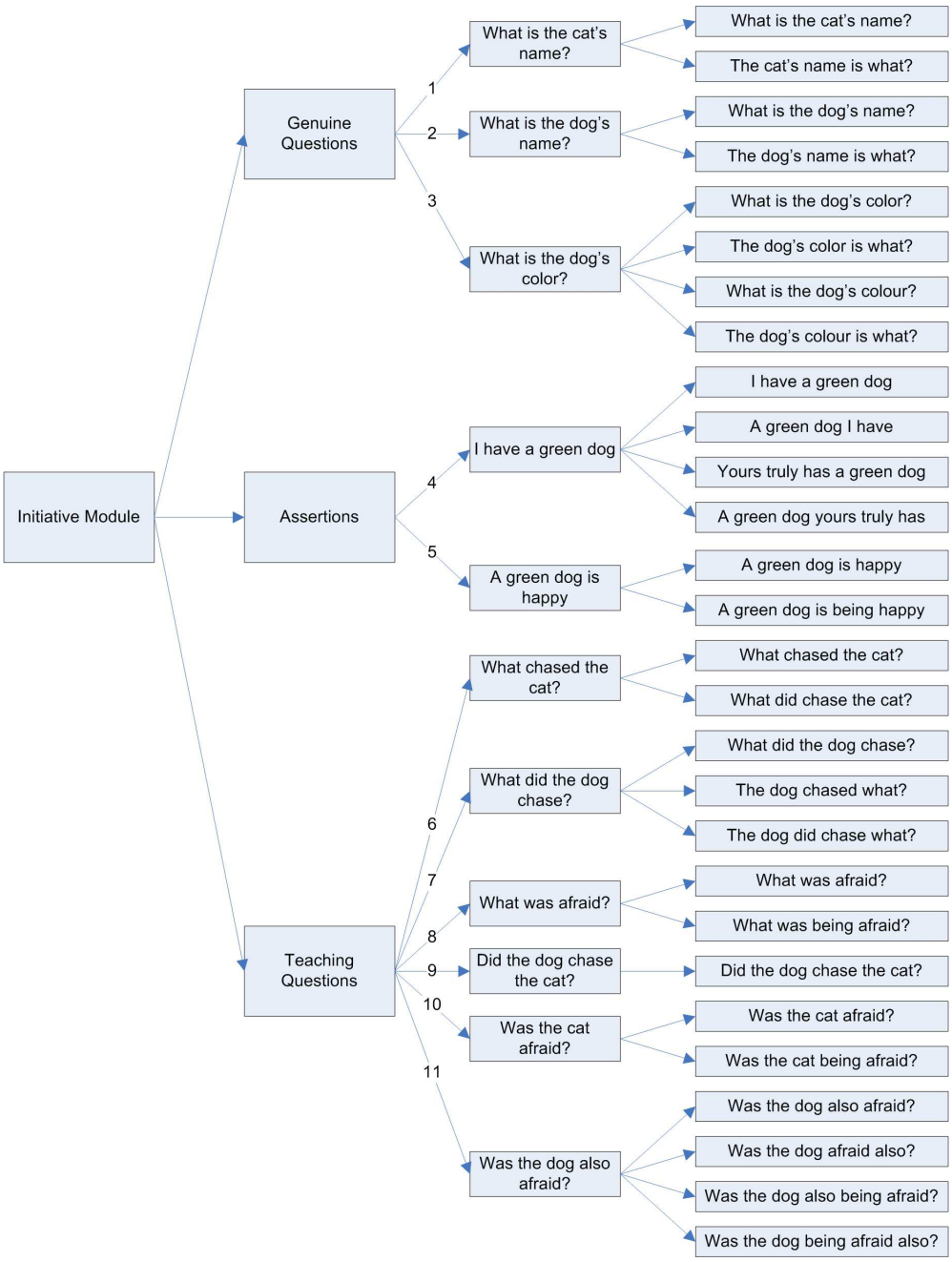


Figure 3.1: Example of a generated tree containing the possible initiatives

3.3.1 Scoring the initiatives based on the criteria

In this section is described how the scores for each criterion are determined. Between brackets is shown which kinds of initiatives the corresponding score is applied to.

Focus (assertions)

The focus score is calculated based on a number of different things. First of all the similarity with the list of possible topics is calculated. In order to do this the topics of the initiative are determined; this is also an ordered list like the topics list of a single sentence.

The similarity between the two lists is calculated by first ranking the possible topics list in the following way:

- The first element will get a score of *focus-start-score*
- The second element will get a score of *focus-start-score* - 1
- ...
- The *focus-start-score*-th element and all later elements will get a score of 1

To find the similarity score the ranked list of possible topics is looped and the values of the elements that appear in the list of initiative topics are added. As mentioned earlier the list of initiative topics contains referents as well as predicates. However, if the referent appears both in the initiative and in the possible topics list, the corresponding predicate automatically appears in both lists as well. If the initiative would get a score for both, the score would be too high. Therefore it is made sure that the predicate is skipped if the corresponding referent is already in the list of possible topics. In that way the initiative will only get a score for the referent if both the referent and the predicate are in the list of possible topics.

Some additional scores are assigned in the following cases:

- *focus-subject-score* points are added if the initiative's subject is in the list of possible topics.

  This is done to make sure that initiatives which contain one of the topics as subject are preferred to initiatives which contain the same topic but as a direct or indirect object.

- *focus-speaker-score* points are added if one of the speakers is in the initiative and the speaker does not appear in the list of possible topics.

  This is done because the speakers should always be in the list of possible topics, because they are always in focus. This score is only added when the speaker does not appear in the list of possible topics, because otherwise points will be assigned twice.

- *focus-also-score* points are added if the word 'also' has to be added.

  This is done because in this case the initiative is almost the same as the last sentence. The algorithm for 'also' only looks at the last utterance and not the previous *two* sentences, so an initiative which contains the word 'also' is even better.

  In section 3.4.1 is described how is decided whether the word 'also' has to be added.

Order of acquisition (genuine questions)

This criterion assumes that all question generation rules are entered in the most natural order. In this case the different questions can be ranked as was done with the list of possible topics:

- The question that was entered first gets a score of *order-score*
- The question that was entered second gets a score of *order-score* - 1
- ...
- The question that was entered *order-score*-th and all later questions get a score of 1

Genuine questions don't get a focus score, because some elements of the focus score are not applicable to genuine questions. However, the focus criterion is important for genuine questions, so the focus criterion is implemented as part of the order score. In the following cases some additional points are assigned:

- *order-new-entity-score* points are added if the entity is a new entity.

  This is done to prefer questions about new entities to questions about entities which have already been on topic for a while.

- *order-subject-score* are added if the question is about the subject of the previous sentence.

  This is somewhat similar to *focus-subject-score*, because this way questions about the subject are preferred to questions about the object. For example, if the user enters the sentence "The dog chased the cat" questions about the dog are preferred to questions about the cat.

- *order-no-speaker-score* points are added if the question is not about the student.

  This contradicts with the *focus-speaker-score*, so this is one of the reasons that the normal focus score is not applicable to genuine questions. This criterion makes sure that questions about explicitly mentioned topics are preferred to questions about the student, even if the student is in the list of topics. This is done because questions about the student are always good initiatives, even if the student is not in the list of possible topics. Questions about other topics can only be asked when they are on topic and therefore it is better to prefer these questions.

Strategy (assertions)

The strategy criterion assigns scores based on how good the initiative is with respect to possible initiatives later in the dialogue. How good the initiative is, is based on two different criteria.

The first of these criteria is the number of conditions in the knowledge base which contain one or more of the referents. If this number is high this means that the system can probably make another assertion about the same referent later.

In order to assign a score based on this criterion first the referents of the initiative are determined and are ranked in the following way:

- The first referent gets a score of *strategy-nr-refs-score*
- The second referent gets a score of *strategy-nr-refs-score* - 1
- ...
- The *strategy-nr-refs-score*-th referent and all later referents get a score of 1

Then for each of the referents the number of conditions in the knowledge base is determined which contain this referent. Finally the weighted sum of these conditions is taken.

The second criterion is the number of possible genuine questions about the used predicates. At first sight this criterion seems useless, because it means that assertions about which genuine questions can be asked are preferred to assertions about which no genuine questions can be asked. However, while generating the genuine questions it is checked that the referent is introduced by the student and not by the system, but extra scores are assigned for the following reason. If the system introduces a new predicate the chance is rather big that the student tells something about a similar entity and therefore the genuine questions can be applied to that entity. For example if the system generates the assertion "I have a dog" and the student also has a dog, he will probably tell so. The system can respond with questions about that new dog afterwards.

Calculation of this part of the strategy score is done in the same way as the score for the genuine questions. First the predicates in the initiative are determined and ranked in the same way as the referents. Then for each predicate the number of possible genuine questions is determined and finally the weighted sum is taken.

Semantics (teaching questions)

The semantics criterion assigns scores based on the difficulty of the question and on the balance between the different teaching questions. First of all *semantics-wh-score* points are assigned to wh-questions, *semantics-yes-score* points are assigned to yes-questions and *semantics-no-score* points are assigned to no-questions. These scores represent the preference between the different kinds of

teaching questions. If a high value is chosen for the *semantics-wh-score* it means that wh-questions will be more likely to be chosen. This is a good idea, first of all because wh-questions are more difficult than yes/no questions, and secondly, because wh-questions stimulate producing of the language because they require a full sentence answer. Furthermore it is useful to assign a high value to *semantics-no-score* because only no-questions which are on topic are generated, so it is good to prefer no-questions if there are any.

Furthermore the complexity of the used referring expression is calculated, in other words the number of necessary adjectives. If the referring expression contains adjectives it means that there is more than one entity with the same predicate and therefore it will be harder to answer the question correctly. The number of necessary adjectives is determined and this result is multiplied by *semantics-complexity-score*.

Finally it is checked if the referring expression used in the initiative is different from the referring expression used when entering the sentence. If the referring expressions are different *semantics-different-score* points are assigned because again it is harder to answer the question correctly. It is also possible to help prompt these kind of questions. The current referring expression generation algorithm takes a list containing adjectives that cannot be used in the referring expression. This way it can be checked if there exists a referring expression that uses no adjectives originally used. If there is such a referring expression that one can be used instead of the original one and a more difficult question has been generated. This is not done in the current algorithm, but it can be included quite easily.

Two additional scores are assigned to make sure that the teaching questions are asked in a somewhat balanced way. The first score is *semantics-new-question-score* which is assigned to questions about propositions which have never been turned into a question before.

Finally *semantics-balanced-score* points are assigned to teaching questions of a particular kind if the previous two teaching questions were of a different kind. This means that yes/no-questions get *semantics-balanced-score* points if the previous two questions were wh-questions and the other way around.

Syntax (all)

The syntax score is calculated by determining which target rules are used in the initiative and taking the sum of the corresponding values. In order to determine which rules are used the actual sentences in natural language have to be generated. Each sentence structure is then traversed and all rules used in the sentence are determined. Next the algorithm loops through all target rules and takes the sum of the values of the rules that are used in the initiative. The values of the target rules will be updated after every initiative which will be described in section 3.5.

In order to calculate the syntax score an MRS is turned into all possible sentences in natural language. However, some of these sentences sound rather strange, e.g. the sentence “From Holland yours truly also is” instead of the

sentence “I am also from Holland”. In section 5.3.3 is described that there is some work ongoing to prefer certain sentences to others. For the moment sentences which contain rules which are not used when authoring the lesson will get a really low syntax score. This way initiatives which contain the construction “yours truly” are not likely to be chosen. Once the code for preferring certain sentences to others is finished the updating of the syntax scores can be done in the normal way again.

In section 2.3.3 also the balance criterion is described. This criterion is not implemented as a score like the other criteria, but is part of the initiative selection algorithm which is described in section 3.3.4.

While generating the possible initiatives initiative-item-structures are used. They contain the initiative in update form and scores for some of the aforementioned criteria. When the syntax score is calculated those initiative-items are transformed into initiative-sentence-items which contain the initiative in update form, the same scores as the initiative-items plus a syntax score, and the actual sentence in natural language. This means that different initiative-sentence-items can correspond to the same initiative-item.

3.3.2 Scaling all scores

In the previous step of the algorithm all scores have been assigned, but before they can be combined they have to be scaled. There are different possibilities to do this, but the easiest way is to scale them linearly. The algorithm simply finds the maximum value of each of the different criteria and then divides all scores related to that criterion by this value and multiplies by 100. This way all scores are scaled between 0 and 100.

3.3.3 Combining the scaled scores

After all scores have been scaled, the scores have to be combined which can be done in many different ways. One way is to take the weighted sum of the separate scores. It is an easy way to implement, but it requires much consideration about assigning useful values to the weight parameters, especially in this project because there are already many parameters for assigning single scores.

Another way is to build a simple elimination algorithm by looking at the different scores independently. In that case you have to decide boundary values for each of the criteria and if an initiative has one of its scores below that boundary value the initiative is ruled out. This is also an easy way to implement, but again it might be hard to come up with good boundary values and there is a chance that either all initiatives are ruled out which means that there would be no good option, or that the algorithm returns several initiatives which have to be compared afterwards using a different algorithm.

One final way to combine the different scores is to make a detailed if-then-else elimination algorithm which does look at all scores, but in an incremental way. The algorithm could for example look at the focus score first and decide

to throw away all initiatives with a focus score below some boundary value. Then the algorithm would look at another criterion (e.g. the balance score) and decide which initiatives can be eliminated based on both the focus score and the balance score. This is actually some kind of combination of the other two ways. One of the problems with this algorithm is that some kinds of initiatives always have a score of zero for a particular criterion. Furthermore this algorithm still needs a way to combine the different scores just as the first algorithm, so it is probably more complicated than necessary.

The main problem with the previous ways to combine the different scores, is that it is hard to compare the different kinds of initiatives. For example, using the scores in section 3.3 it is possible to compare the different genuine questions and decide which genuine question is best, but it is rather difficult to compare a genuine question to an assertion. Therefore it was preferable to find a better way to combine the different scores. Instead of generating all possible initiatives first and then try to combine the separate scores, the initiatives are evaluated while they are generated. An advantage of this is that less initiatives have to be generated which makes the algorithm perform more efficiently. Furthermore the selection algorithm makes sure that different kinds of initiatives never have to be compared, only initiatives of the same kind. This means that only two or three different scores have to be combined (instead of six) and therefore only a few weight parameters have to be used at the same time which makes it much easier to try different values. In the following section the initiative selection algorithm is described.

3.3.4 Selecting the kind of initiative

In order to decide which initiative is best the initiatives are generated in an incremental way. This means that first a number of initiatives is generated and then it is checked if there are initiatives which are good enough. If there are such initiatives the remaining initiatives don't have to be generated anymore. If there are no initiatives which are good enough the next number of initiatives is generated.

The algorithm first generates the assertions and classifies these into the following three categories:

- Assertions which contain 'also'

  This category includes the assertions which contain the word 'also'. These assertions are the best ones because they only look at the very last sentence instead of the previous two sentences.

- Assertions about a topic in the previous sentence

  This category includes the assertions about a topic explicitly mentioned in the very last sentence. These assertions are definitely good ones, but not as good as the the assertions which contain the word 'also'.

- Remaining assertions

  This category includes all remaining assertions, so it includes assertions about the system, assertions about a new topic and assertions about an old topic not used in the previous sentence.

The 'also' assertions are the best initiatives and therefore these assertions will be chosen if there are any. In that case no genuine questions or teaching questions have to be generated. If there are no assertions which include the word 'also' the genuine questions are generated next. These are classified into the following two categories:

- Genuine questions about a topic in the previous sentence

  This category includes the genuine questions about a topic explicitly mentioned in the very last sentence. These are the best genuine questions that are possible.

- Remaining genuine questions

  This category includes the remaining genuine questions, so it includes the genuine questions about the topics mentioned in the second last sentence and the genuine questions about the user.

The genuine questions about topics mentioned some time ago, but not mentioned in one of the previous two sentences are not generated. If they would be generated it could be unclear to which referent they referred. The only way to generate those questions is to get the referent back into focus, which happens automatically if the system generates a teaching question about that referent.

If there are genuine questions in the first category, those questions are very good and will therefore be chosen. In that case no teaching questions have to be generated because it is absolutely sure that the genuine questions are good initiatives to take. If the user starts talking about a dog for example and the character played by the system knows questions about dogs it is definitely the best time to ask them. It is in fact even necessary because it would be rather strange if the system asked them at a later time.

If there are no genuine questions about an explicitly mentioned topic, the algorithm checks if there are assertions about a topic mentioned in the last sentence. If there are any, these assertions are chosen and still no teaching questions have to be generated.

If at this moment no assertions or genuine questions have been found yet three possibilities remain: choosing one of the remaining assertions, choosing one of the remaining genuine questions or creating the teaching questions. This decision is based on the balance criterion. Three variables are kept which represent the number of system initiatives between the last initiative of a particular kind and the current initiative. For example if the last initiative was a teaching question the variable representing the balance criterion for teaching questions has a value of 1; if the last assertion was seven initiatives ago the variable representing this has the value 7.

The kind of initiative which was used longest ago (the variable with the highest value) is chosen if there are any initiatives of that kind and if the last three initiatives were teaching questions. This last condition is used to make sure that the system does not generate all genuine questions and assertions first and end with all teaching questions.

The algorithm in this section actually orders the different kinds of initiatives in the following way:

- assertions which contain 'also'
- genuine questions about an explicitly mentioned topic
- assertions about an explicitly mentioned topic
- remaining initiatives, combined with balance criterion

The algorithm loops through these categories in this order and once it finds initiatives in a category the algorithm returns these. Furthermore it is made sure that during the beginning of the dialogue genuine questions about the student are preferred to assertions about the system. This is firstly done to get more information from the student, but it is also more polite.

The result of this algorithm is that the beginning of the dialogue consists mainly of genuine questions and assertions and the end of the dialogue consists mainly of teaching questions. The condition that at least three teaching questions have to be asked before the system creates a genuine question about the student or an assertion about the system makes sure that the initiatives are mixed somewhat better, but still most teaching questions are asked at the end of the dialogue. However, this is not a disadvantage, because this way the common ground is populated during the beginning of the dialogue and the end of the dialogue is used to check if the student really understands everything.

The algorithm looks like the following:

```
1. generate assertions and classify them:
   a. assertions which contain 'also'
   b. assertions about a topic in the previous sentence
   c. remaining assertions

2. if there are assertions in category a:
      choose them and skip the other steps of the algorithm

3. generate genuine questions and classify them:
   a. questions about a topic in the previous sentence
   b. remaining questions

4. if there are genuine questions in category a:
      choose them and skip the other steps of the algorithm

5. if there are assertions in category b:
      choose them and skip the rest of the algorithm

6. choose the kind of initiative based on the balance criterion:
   if the last three initiatives were teaching questions:
      if there are any assertions and assertions have
         the highest balance score:
```

```
      choose them
   else if there are any genuine questions and genuine
      questions have the highest balance score:
      choose them
   else generate the teaching questions
else generate the teaching questions
```

3.3.5 Improving the efficiency

Figure 3.1 showed an example of a generated tree for only a simple character and a dialogue which had just started. In this section is described how this tree can be pruned like an ordinary AI search tree. As you can see in the figure the search tree consists of three different levels and there is a way to prune each of these levels.

Pruning the different kinds of initiatives (level 1)

The algorithm described in the previous section that selects the initiatives to be generated can be compared to pruning the different kinds of initiatives. If an algorithm like 'weighted sum' would have been used all initiatives had to be generated, so using an algorithm that chooses the kind of initiative first is definitely a good way to prune the search tree. The pruning is different from the kinds of pruning described in the following two subsections, because this kind of pruning is obligatory; it is part of the algorithm for choosing the best alternative. The other ways of pruning are optional; they only make the algorithm perform more efficiently.

Pruning the number of teaching questions (level 2)

If there are no genuine questions or assertions about one of the main topics all possible teaching questions will be generated. The number of teaching questions is usually much higher than the number of genuine questions or the number of assertions. Therefore it would be useful if the number of teaching questions could be reduced.

This can be done by eliminating the teaching questions that have a semantics score of zero. Wh-questions and no-questions always have a semantics score and questions which contain difficult referring expressions also have a semantics score, so only the simplest yes-questions are ruled out. The questions with a semantics score of zero, but which include the word 'also' are included however, because these can be interesting yes/no-questions. An example of this is the following: if the user enters the sentence "The dog jumped", a good teaching question would be "Did the dog also bark?", assuming there is an entity which barked, otherwise the question cannot be generated.

Furthermore the questions which have the same list of topics as the last initiative are eliminated. This is firstly done to prevent the system from asking different questions which were created using the same proposition in a row. These questions often have the same answer, so it is better if another question is asked in between. An additional advantage is that the last sentence entered

by the user is not turned into a teaching question immediately. This is a useful addition because turning the last sentence into a teaching question would generate questions which are too easy to answer.

Finally as already described in section 3.2.3 only the no-questions which are on topic are generated which rules out a large number of no-questions. Furthermore using the list with already asked teaching questions only teaching questions which haven't been asked yet are generated which can also be compared to pruning.

Pruning the number of initiatives to apply the syntax score (level 3)

The final way to prune the search tree is to apply the syntax score only to the initiatives which score high in the other criteria. In order to achieve this the initiatives in the second level are ordered by their total score and only the top four are passed to the next level. Finally the one with the highest total score (including the syntax score) is chosen as the best initiative. If all four initiatives have a syntax score of zero the initiatives are not useful anymore because the rules used in these initiatives have already been learnt. In that case the next group of four initiatives from the second level is passed to the final level. This continues until an initiative is found whose syntax score is higher than zero or until all initiatives have been examined to see if their syntax score is higher than zero.

Results of pruning the search tree

In this section the results of pruning the search tree shown in figure 3.1 is demonstrated.

The first kind of pruning (pruning the different kinds of initiatives) is always applied, because it is part of the algorithm that chooses the best initiative. Therefore first the assertions are generated which results in creating the initiatives 4 and 5. Both assertions do not contain the word 'also', so the genuine questions are generated next. This results in creating the three genuine questions named 1, 2 and 3. Next it is checked if one of these genuine questions is about one of the predicates used in the last sentence ("The cat was afraid"). It turns out that the first question is about one of the main topics and therefore this question is passed to the next level to assign a syntax score. The other questions and the assertions are thrown away and the teaching questions are not even generated. This results in generating five initiatives instead of eleven and generating only two sentences instead of generating 28 sentences if an algorithm like weighted sum was used which is obviously a big progress.

In the following is assumed that there are no genuine questions and no assertions about one of the main topics which means that the teaching questions have to be generated.

If pruning the teaching questions is turned off six different teaching questions are generated. If pruning the teaching questions is switched on however only the wh-questions, the no-questions and the questions which contain 'also' are

generated. Furthermore the question "What was afraid" is thrown away because that question has the same topics list as the previous sentence and is therefore assumed to be too easy. This results in generating only three different teaching questions (the questions named 6, 7 and 11).

Finally depending on which kinds of pruning are applied a number of initiatives is passed to the next level to calculate the syntax scores. Because pruning the different kinds of initiatives is always applied only the genuine question "What is the cat's name?" is passed to the final level. The syntax score is calculated for both generated sentences and the one with the highest score is chosen. This kind of pruning does not affect the result in this example, but we will assume that the algorithm chose to create a teaching question to demonstrate the results of this kind of pruning. If pruning the teaching questions is turned off there are six possible teaching questions. The teaching questions are ordered to their semantics score and the top four are passed to the final level (initiatives 6, 7, 8 and 11). For these initiatives the syntax score is determined and a new total score is calculated. Next the initiatives are ordered to their total score and the one with the highest total score is chosen. If all four of the initiatives have a syntax score of zero the next four initiatives are passed to the next level. In this case there are only two remaining teaching questions (initiatives 9 and 10) and their syntax scores are calculated. The one with the highest total score is chosen and if both teaching questions have a syntax score of zero there are no useful initiatives anymore.

The example in this section shows that applying different kinds of pruning can result in much more efficient generation of initiatives.

3.4 Adding necessary discourse markers

There are many different discourse markers, some of which are obligatory and some of which are only desirable. Adding discourse markers can be an extensive task if you want to do it correctly. Because this was not the main goal of the project it is decided only to add the most necessary discourse markers.

3.4.1 Adding the word 'also'

As explained in section 2.3.4 the word 'also' is sometimes obligatory, so to find out if this is the case the chosen initiative has to be checked once more. From the examples given in that section it appears that there are exactly two cases in which the word is needed. In order to find out if one of those cases applies the algorithm first checks if the main events of the propositions (the first elements of the topics lists) have the same arity. This can be done by checking if the length of the topics list of the previous sentence and the length of the topics list of the initiative are the same. Furthermore it has to be checked if the events are of the same kind, e.g. both verbs or both adjectives. The topics lists (ran $x9$ dog) and (green $x9$ dog) for example are almost the same, but it would be

inappropriate to add the word also to the corresponding sentences "The dog ran" and "The dog is green".

If the list of initiative topics satisfies both conditions the algorithm counts the number of differences between the topics of the previous sentence and the topics of the initiative. If this difference is exactly one, the sentences are almost the same and the word 'also' is needed. If the number of differences is two *and* the difference is a swap of arguments the word 'also' is needed too. Examples of these cases were given in section 2.3.4.

The word 'also' can appear at different places in the sentence, so the string "also" cannot simply be added to the beginning or the end of the sentence. Instead the predicate 'also' has to be added to the nucleus of the MRS of the initiative. The algorithm can then rely on the sentence generator which makes sure that the word will be put at the correct place in the sentence.

3.4.2 Adding a remark before a teaching question

To make sure that the student always knows whether a question is a teaching question or a genuine question, every teaching question is preceded by one of the following strings:

- Let me make sure you remember:
- Let me check if you remember:
- Let's check if you remember:
- Let's see if you remember:

This sentence is not added to the MRS, but is simply concatenated to the beginning of the sentence of the teaching question.

3.5 Processing initiative and user response

The system can generate three different kinds of initiatives. Depending on the kind of initiative the dialogue continues differently. The easiest case is when the system generates an assertion. In that case the assertion is simply stored in the common ground and the dialogue continues normally. We assume for the moment that the user will always understand the assertion and will not ask any clarification questions.

If the system generates a genuine question or a teaching question the user's next input is the answer to the question. Therefore it first needs to be checked whether the last initiative was a question before the user's input can be processed. An answer to a question can be a whole sentence or a simple 'yes' or 'no' input.

Therefore the following four cases can occur:

1. The question is a yes/no genuine question

   If the answer is 'yes' the propositional part of the genuine question is stored in the common ground and the system responds with 'ok'.

   If the answer is 'no' the genuine question doesn't have to be stored, so the system only generates an 'ok' message. Eventually it would be useful to store the negation of the question proposition, but Te Kaitito doesn't handle negation at present.

2. The question is a wh genuine question

   In this case the user's answer is a whole sentence and the sentence is parsed in the normal way and stored in the common ground. The system responds with 'ok'.

3. The question is a yes/no teaching question

   First the system finds the correct answer and compares this answer to the user's input. If the answer is correct the system responds with 'correct'; if the answer is incorrect the system responds with 'incorrect'. Because the teaching question was based on the common ground nothing has to be added to the common ground.

4. The question is a wh teaching question

   In this case the user's answer is a whole sentence. The system finds all possible answers and compares those to the user's answer. If the answer is one of the possible answers the system responds with 'correct' and the dialogue continues normally. If the answer is incorrect the system repeats the question to give the student a second chance and if necessary even a third chance. If the student still doesn't know the correct answer the system will respond with an 'incorrect' message containing the correct answer.

Processing the chosen initiative also involves updating the values of the target syntactic rules. If the initiative is an assertion we cannot know for sure if the student understands the assertion correctly, so the values are not changed.

If the last initiative was a teaching question the updating of the values depends on the correctness of the answer. If the answer is correct the values of all used rules are reduced. If on the other hand the answer is incorrect, it is clear that the student doesn't know the rules yet, so the values of the used rules have to be increased. In the current system reducing the values consists of decreasing the value by one point. Another possible way is to divide the values by some number which means that the scores are updated exponentially. That way the initiatives which contain syntactic rules that have never been used are preferred more strongly to the other initiatives. Increasing the values is achieved by adding one point to the current value, but this can also be done

exponentially. Alternatively, a higher value than one can be chosen to make sure that initiatives containing those rules are preferred even more.

If the last initiative is a genuine question it is not possible to find out if the answer is correct, so it is simply assumed that the answer is correct. The values of the rules used in the question are reduced and if the question is a wh-question also the values of the rules used in the answer are reduced.

The updating of the values of the different target rules represents the actual *language* teaching. The student's difficulties are diagnosed and the system will try to teach the student the difficult rules. Presumably, a real teacher will try to let the student practice these rules in a different context. Currently, the system only tries to find initiatives which contain some of these rules, but this is an area which can be very much elaborated in future work which is described in section 5.4.3.

3.6 'Lesson over' test

At the beginning of the lesson the dialogue contains a set of target rules which has been constructed by a lesson author. The values of those rules are initialized at *target-start-value* and during the dialogue the values of those rules are updated as explained in the previous section. As long as the student keeps taking initiatives there is no need to stop the lesson, but if the student gives the initiative to the system, the system can decide that the lesson is over. There are three different situations in which the system decides to end the lesson.

The first situation occurs when the values of all target rules are zero. If a rule has been used in the correct way for enough times the value will finally reach zero. If all values are zero it means that all rules have been learnt properly and the lesson is over. Then the student can move on to the next lesson to learn new rules.

Another situation in which the lesson is stopped occurs when the system cannot generate any useful initiatives anymore. This happens when all initiatives have a syntax score of zero, but when there are some rules which have a value higher than zero. In that case not all rules have been learnt properly, but the system can't help the student anymore.

The final situation in which the system decides to stop the lesson occurs when the value of one of the rules is higher than a particular boundary value. Every time a rule is used incorrectly the value of that rule is increased. The system will try to choose more initiatives which use this rule, but this does not necessarily mean that the student will understand the rule this time. If the student fails to understand a particular rule for many times in a row the value will increase very quickly. In this case the system decides to stop the lesson and tells the student that it is better to go back to the previous lesson or to find help from his teacher.

3.7 Values of the parameters

In this chapter many parameters have been used without giving the actual chosen values. This has been done because the parameters are easy to adjust, so if the reader likes he can try his own values. In this section we will however give a table with the values we used. In appendix B a description of each of the parameters and a motivation for the choice of its value are given.

Parameter	Value
focus-start-score	5
focus-subject-score	7
focus-speaker-score	7
focus-also-score	10
order-start-score	3
order-new-entity-score	1
order-subject-score	1
order-no-speaker-score	1
strategy-nr-preds-score	2
strategy-nr-refs-score	2
semantics-wh-score	1
semantics-yes-score	1
semantics-no-score	3
semantics-different-score	1
semantics-adjectives-score	1
semantics-new-question-score	1
semantics-balanced-score	2
syntax-start-score	7

Chapter 4

Results

In this chapter the results of the initiative module are presented using some example dialogues. In section 4.1 two examples are given, one showing a dialogue in which the student takes no initiatives and one in which the student does take initiatives himself sometimes. In section 4.2 another example is given to show that the system works well with different characters.

4.1 Example character

In this section two examples are given using the same character. The example described in section 4.1.1 shows a dialogue in which the student never takes any initiatives and the example in section 4.1.2 shows the results if the student takes some initiatives himself.

The character used in this section has question generation rules to create the following genuine questions (entered in the same order):

- How are you?
- What is your name?
- Where are you from?
- What is a dog's name?

Furthermore the character can make the following assertions:

- I am John
- I am fine
- I am from Holland
- I have a green dog
- The green dog chased a cat
- The green dog was happy

4.1.1 Student who takes no initiatives

The following example shows the dialogue in which the user never takes any initiatives. The example in the next section is somewhat better, but this example is used to show how the algorithm that chooses the kind of initiative works. The utterances which are underlined are utterances taken by the system. In the text is referred to these initiatives using the numbers shown between brackets.

```
>
how are you (1)
> i am fine
okay
>
i am also fine (2)
>
what is your name (3)
> my name is ann
okay
>
Let me check if you remember:
who is fine (4)
> i am fine
correct
>
from where are you (5)
> i am from holland
okay
>
holland i am from also (6)
>
a green dog i have (7)
>
a cat the dog chased (8)
>
the dog was happy (9)
>
Let's see if you remember:
what did the dog chase (10)
> the dog chased the cat
correct
>
Let me check if you remember:
was the cat being happy (11)
> no
correct
>
Let me make sure you remember:
who does have the dog (12)
> you have the dog
correct
>
i am john (13)
>
Let's see if you remember:
did john chase the cat (14)
> no
correct
>
Let's check if you remember:
what chased the cat (15)
> the dog chased the cat
correct
>
Let's see if you remember:
was the dog being happy (16)
> yes
correct
>
Let's check if you remember:
am i being from holland (17)
> yes
correct
>
Let's see if you remember:
from what are you (18)
> i am from holland
correct
>
Let me check if you remember:
i do have what (19)
> you have the dog
correct
>
Let me make sure you remember:
are you being fine (20)
> yes
correct
>
Let me make sure you remember:
from what am i (21)
> you are from holland
correct
>
There are no useful initiatives anymore
> bye
bye
```

The first step of the algorithm generates the possible assertions. Nothing has been said yet which means that the list of possible topics is empty, so there are no assertions which contain the word 'also'. The next step is to generate the genuine questions. For the same reason there are no genuine questions about a topic explicitly mentioned in the previous sentence, but there are three genuine questions about the student. Because it is still the beginning of the dialogue these genuine questions are selected. Their order score and syntax score are determined and the one with the highest score is chosen.

The system generates the following output before each initiative if the printing parameters are set to t:

```
------------------------
BEFORE-SCALING-AND-SORTING
"ORD    MRS"
"  3    you are how"
"  2    your name is what"
"  1    where are you from"
------------------------
AFTER-SCALING-AND-SORTING
"ORD    MRS"
"100    you are how"
" 67    your name is what"
" 33    where are you from"
------------------------
INITIATIVE-SENTENCES
"ORD    SYN    TOT    SENTENCE"
"100     78    178    how are you"
" 67    100    167    what is your name"
" 33     78    111    from where are you"
"100      3    103    you are how"
" 33     56     89    where are you from"
" 67      3     70    your name is what"
" 33      3     36    you are from where"
" 33      3     36    you are being from where"
------------------------
how are you
```

The first table shows the possible initiatives and their unscaled order scores. The sentences printed in the table are actually MRSs, but for clarity one of the sentences corresponding to the MRSs is printed. The second table shows the same initiatives, but then with scaled scores and ordered by their scores. The final table shows the sentences that can be generated for each of the MRSs together with their order scores, their syntax scores and their total scores. The initiative sentences are ordered by their total score and finally the first one is chosen; in this case the genuine question "How are you?".

Next the user responds with "I am fine" and the algorithm for finding the best initiative is started again. In this case there is an assertion which includes the word 'also' and therefore this assertion is chosen without generating the genuine questions or the teaching questions.

The next initiative is a genuine question about the user, because there are no assertions which contain the word 'also' and it is still the beginning of the dialogue, so questions about the student are assumed to be good initiatives. The system asks for the student's name and the student answers that her name is Ann.

In the next situation (initiative 4) there are no assertions which contain 'also' and no genuine questions or assertions about one of the main topics. At this point the balance criterion is used to decide that the next initiative should be a teaching question. Therefore the teaching questions are generated which results in the following output:

```
------------------------
BEFORE-SCALING-AND-SORTING
"SEM    MRS"
"  2    am i fine"
```

```
"  2    who is being fine"
"  2    are you fine"
"  2    who is being fine"
------------------------
AFTER-SCALING-AND-SORTING
"SEM    MRS"
"100    am i fine"
"100    who is being fine"
"100    are you fine"
"100    who is being fine"
------------------------
INITIATIVE-SENTENCES
"SEM    SYN    TOT    SENTENCE"
"100    100    200    who is fine"
"100    100    200    who is fine"
"100     80    180    are you being fine"
"100     73    173    are you fine"
"100     73    173    am i being fine"
"100     67    167    am i fine"
"100      7    107    who is being fine"
"100      7    107    who is being fine"
------------------------
Let me check if you remember:
who is fine
```

In this case all teaching questions have the same semantics score, so the choice completely depends on the syntax score. The question "Who is fine" is chosen and the user responds with "I am fine". If the student would have answered "You are fine" the answer would also be correct, because both the student and the system are fine.

At this moment the student is in the list of possible topics and therefore there is a genuine question about one of the main topics and this question is chosen. The system asks from where the student is and the student tells that she's from Holland. Initiative 6 is an assertion which contains the word 'also' and can therefore be chosen immediately.

At that moment the system is in the list of possible topics and there are two possible assertions. The result looks like this:

```
------------------------
BEFORE-SCALING-AND-SORTING
"STR    FOC    MRS"
" 12      9    john i am"
" 17      9    a green dog i have"
------------------------
AFTER-SCALING-AND-SORTING
"STR    FOC    MRS"
"100    100    a green dog i have"
" 71    100    john i am"
------------------------
INITIATIVE-SENTENCES
"STR    FOC    SYN    TOT    SENTENCE"
"100    100    100    300    a green dog i have"
"100    100     76    276    i have a green dog"
"100    100      5    205    yours truly has a green dog"
"100    100      5    205    a green dog yours truly has"
" 71    100     33    204    i am john"
" 71    100      5    176    yours truly is john"
" 71    100      5    176    john yours truly is"
" 71    100      5    176    john i am"
------------------------
a green dog i have
```

Assertions also have a focus score, so the scores in the last table are from left to right the strategy score, the focus score, the syntax score and the total score. The MRS for "I have a green dog" has a higher strategy score, because there are more relations with 'dog' in the knowledge base than relations with 'John'. Furthermore there is a genuine question about a dog, so an additional two points are assigned to the strategy score of that assertion. The focus scores for both assertions are the same, so the result depends on the combination of the strategy score and the syntax score which is shown in the third table of the result.

Next the green dog is on topic and the assertions "The dog chased a cat" and "The dog was happy" are generated, because they are about a topic mentioned in the previous sentence.

At this time all genuine questions have been asked and there is only one assertion left. This assertion is about the system and will only be made once three teaching questions have been generated or the system is in the list of possible topics. In this case both conditions are satisfied at the same time: after the next three initiatives three teaching questions have been asked and the system is on topic. Therefore the next initiative (initiative 13) is the assertion about the system and it tells his name is John.

Right now all assertions have been made and all genuine questions have been asked, so the only thing the system can do is asking teaching questions. The result of this is shown in the dialogue at the beginning of this section, but further explanation is omitted. In the next section an example is given in which the user does take initiatives once in a while and in that example more attention is paid to the teaching questions.

4.1.2 Student who takes initiatives

The following example shows the dialogue in which the user also takes initiatives. The same character is used as in the previous example, but as can be seen the results are different because the user's inputs influence the execution of the algorithm. The beginning of the dialogue is similar to the one in the previous section and will be described only roughly.

```
>
how are you (1)
> i am fine
okay
> how are you
i am being fine
i am fine
>
what is your name (2)
> my name is ann
okay
> who are you
john i am
i am john
john yours truly is
yours truly is john
>
a green dog i have (3)
> i have a red dog
okay
>
what is the red dog 's name (4)
> the red dog's name is mary
okay
>
Let's see if you remember:
who does have the red dog (5)
> i have the red dog
correct
>
from where are you (6)
> i am from holland
okay
>
holland i am from also (7)
>
Let me make sure you remember:
are you also from holland (8)
> yes
correct
>
Let's check if you remember: (9)
is yours truly also being from holland
> yes
correct
>
Let's see if you remember:
who does have the green dog (10)
> you have the green dog
correct
>
a cat the green dog chased (11)
>
the green dog was being happy (12)
> the red dog was sad
okay
>
Let me check if you remember: (13)
was the red dog also being happy
> no
correct
>
Let's see if you remember:
what chased the cat (14)
> the green dog chased the cat
correct
>
Let me make sure you remember:
was the cat being sad (15)
> no
correct
>
Let me check if you remember:
i do have what (16)
> you have the green dog
correct
>
Let me make sure you remember:
what did the green dog chase (17)
> the green dog chased the cat
correct
>
Let me check if you remember:
from what are you (18)
> i am from holland
correct
>
The lesson is over
> bye
bye
```

The first initiative is "How are you?" for the same reasons as in the previous example. The student answers and asks how the character played by the system is. The system answers and generates the next genuine question. Next the student asks for the system's name and the system answers. At this point the system is in the list of possible topics and the assertions "I am from Holland" and "I have a green dog" are the possible assertions. Based on the strategy criterion (as in the previous example) the algorithm chooses the assertion "I have a green dog".

At this point the student takes an initiative and tells that she has a red dog. Then the red dog is in the list of possible topics and the system can ask a genuine question about an explicitly mentioned topic ("What is the red dog's name?"). The originally entered question was "What is a dog's name?" and you can see that the referring expression used for "a dog" is correctly replaced by the referring expression "the red dog".

The student answers the question and at that moment there are no assertions which contain the word 'also' and no genuine questions or assertions about one of the main topics. The balance criterion makes sure that the next initiative will be a teaching question. The possible teaching questions are generated which results in the following:

```
------------------------
BEFORE-SCALING-AND-SORTING
"SEM    MRS"
"  3    do you have the red dog"
"  3    who does have the red dog"
"  3    you have what"
"  3    do i have the green dog"
"  3    who does have the green dog"
"  3    i have what"
"  2    am i fine"
"  2    who is being fine"
"  2    are you fine"
"  2    who is being fine"
------------------------
AFTER-SCALING-AND-SORTING
"SEM    MRS"
"100    do you have the red dog"
"100    who does have the red dog"
"100    you have what"
"100    do i have the green dog"
"100    who does have the green dog"
"100    i have what"
" 67    am i fine"
" 67    who is being fine"
" 67    are you fine"
" 67    who is being fine"
------------------------
INITIATIVE-SENTENCES
"SEM    SYN    TOT    SENTENCE"
"100    100    200    who does have the red dog"
"100     56    156    do i have the green dog"
"100     56    156    do you have the red dog"
"100     33    133    what do you have"
"100     11    111    you do have what"
"100     11    111    you have what"
"100     11    111    who has the red dog"
------------------------
Let's see if you remember:
who does have the red dog
```

At this point no no-questions can be generated, because the referents for 'I' and 'you' are never replaced by other referents. Furthermore no teaching questions have been asked yet, so the semantics scores are very similar; they all have one point because the question has not been asked yet and one point if it is a yes-question or a wh-question (in this case all of the generated questions). Finally the questions which contain one of the dogs get one extra point based on the used referring expressions, because they are somewhat more difficult than the other questions. The scores are scaled and ordered and only the top four are passed to the final level to calculate the syntax scores. It turns out that the question "Who does have the red dog" has a much higher syntax score than the other questions and therefore this question is chosen.

At that moment the student is back in focus and there is still one genuine question left about the student. The system selects this question ("Where are

you from") and the student answers. Next, there is an assertion which contains the word 'also' and this assertion is chosen.

The next two teaching questions may look somewhat easy, but they are chosen because they have a high syntax score. The questions score lower in the semantics score than the questions which contain one of the dogs, but because their syntax scores are much higher they are chosen. It should be noted that initiative 9 looks rather strange at first sight, but there are some ways to correct this which will be described in section 5.3.3. For the moment it is best to read the sentence as if it were "Am I also from Holland?".

After the next three teaching questions the green dog is in the list of possible topics again and the system can make the last two assertions about the green dog: "The green dog chased a cat" and "The green dog was happy".

Next, the user takes an initiative and tells that the red dog is sad. At that point there are three 'animate' objects in the context (the green dog, the red dog and the cat), so from this point on also no-questions can be asked. The next initiative is a teaching question and the following output is generated:

```
------------------------
BEFORE-SCALING-AND-SORTING
"SEM    MRS"
"  5    was the green dog being sad also"
"  5    was the red dog being happy also"
"  3    was the green dog happy"
"  3    what was being happy"
"  5    did the red dog chase the cat"
"  3    did the green dog chase the cat"
"  3    what did chase the cat"
"  3    what did the green dog chase"
"  1    who is being from holland"
"  1    what is yours truly from"
"  1    who is being from holland"
"  1    what are you from"
"  2    do you have the red dog"
"  2    you have what"
"  2    do i have the green dog"
"  2    i have what"
"  2    am i fine"
"  2    who is being fine"
"  2    are you fine"
"  2    who is being fine"
------------------------
AFTER-SCALING-AND-SORTING
"SEM    MRS"
"100    was the green dog being sad also"
"100    was the red dog being happy also"
"100    did the red dog chase the cat"
" 60    was the green dog happy"
" 60    what was being happy"
" 60    did the green dog chase the cat"
" 60    what did chase the cat"
" 60    what did the green dog chase"
" 40    do you have the red dog"
" 40    you have what"
" 40    do i have the green dog"
" 40    i have what"
" 40    am i fine"
" 40    who is being fine"
" 40    are you fine"
" 40    who is being fine"
" 20    who is being from holland"
" 20    what is yours truly from"
```

```
" 20    who is being from holland"
" 20    what are you from"
-------------------------
INITIATIVE-SENTENCES
"SEM    SYN    TOT    SENTENCE"
"100    100    200    was the red dog also being happy"
"100    100    200    was the red dog also being happy"
"100    100    200    was the red dog happy also"
"100    100    200    was the red dog being happy also"
"100    100    200    was the green dog also being sad"
"100    100    200    was the green dog also being sad"
"100    100    200    was the green dog sad also"
"100    100    200    was the green dog being sad also"
" 60    100    160    was the green dog being happy"
"100     50    150    did the red dog chase the cat"
"100      0    100    was the red dog also happy"
"100      0    100    was the green dog also sad"
" 60      0     60    was the green dog happy"
-------------------------
Let's see if you remember:
was the red dog also being happy
```

No-questions have a higher semantics score to prefer them to other questions. In this way the no-question with the highest syntax score is often chosen. The next initiative is also a teaching question and because the previous two teaching questions were yes/no-questions, the wh-questions get one extra point. The result is "What chased the cat?". After this question another no-question can be generated ("Was the cat sad?") and that no-question is chosen. The remainder of the dialogue consists of a number of other teaching questions to make sure that all syntactic constructions are learnt.

4.2 Example teacher character

In this section an example dialogue in given using a different character. In this example not every initiative taken by the system will be explained as in the previous examples, but the most important choices are clarified.

The character in this example has the following genuine questions, entered in the same order:

- Which language do you speak?
- Do you speak Spanish?
- Do you speak English?

Furthermore the character can make the following assertions:

- I am an English teacher
- I speak English
- I speak Spanish
- John is my student

- John likes me
- Mary is my mother
- Mary speaks German

The current referring expression algorithm in the Te Kaitito system does not support substructures which means that assertions with the word 'my' cannot be made. Instead the assertions "John is a student" and "Mary is a mother" will be generated. Once the substructures are added the dialogue will be better which will be described in section 5.2.2.

An example dialogue with this character is the following:

```
>
which language do you speak
> i speak dutch
okay
>
do you speak spanish
> no
ok
>
do you speak english
> yes
okay
>
i also speak english
>
i also speak spanish
>
i am an english teacher
>
john likes me
>
john is a student
>
Let's see if you remember:
who does like me
> john likes you
correct
>
Let me make sure you remember:
what do i speak
> you speak english
correct
>
Let's check if you remember:
do you also speak english
> yes
correct
>
mary speaks german
>
mary is a mother
>
Let me make sure you remember:
what does mary speak
> mary speaks german
correct
>
Let me make sure you remember:
does mary like me
> no
correct
>
Let's see if you remember:
do you speak dutch
> yes
correct
>
Let's check if you remember:
what do i speak
> you speak spanish
correct
>
Let me check if you remember:
do i also speak english
> yes
correct
>
Let me make sure you remember:
do i also speak spanish
> yes
correct
>
Let's check if you remember:
john does like whom
> john likes you
correct
>
Let me make sure you remember:
you speak what
> i speak dutch
correct
>
There are no useful initiative anymore
> bye
bye
```

The first thing that looks strange in this dialogue is that the system first generates the initiative "John likes me" and then the initiative "John is a student". The initiatives are generated in this order because the system is in the list of possible topics and the system appears in the initiative "John likes me", which means that the initiative "John likes me" is about one of the main topics

and the initiative "John is a student" is not on topic at all. Once the algorithm for generating referring expressions has been extended that it will also support substructures, the initiative "John is my student" will also be on topic. Therefore the chance is rather big that this initiative will be preferred to the other one, but this cannot be checked until the referring expression algorithm has been modified.

For the same reason the initiative "Mary is a mother" will be changed to "Mary is my mother" which sounds much better. The order of the initiatives "Mary speaks German" and "Mary is my mother" may change once the correct referring expressions are used, but this will not affect the rest of the dialogue.

As can be seen in the examples in this chapter the system can generate coherent dialogues; all initiatives are appropriate in the context. The beginning of the dialogue is used to populate the common ground and the end of the dialogue consists mainly of teaching questions to check if the student understands everything that has been said and knows all syntactic constructions targeted in the lesson. Furthermore, the teaching questions are a good mixture of wh-questions and yes/no-questions and the questions loop through all different propositions.

Chapter 5

Summary and possible extensions

As the examples in the previous chapter show, it is possible to create some good dialogues, but there are also situations in which the system creates rather strange initiatives. Some of these situations are due to the Te Kaitito system and the grammar currently used, but others have different reasons. There are however things to make the system perform better which will be described in this chapter.

First a summary of the whole project is given in section 5.1. Then the possible extensions are described which can be subdivided into three different kinds of extentions. First of all section 5.2 describes a number of small changes in or extensions to the Te Kaitito system and the grammar which will make the initiative module work better. Furthermore there are some possible extensions to the initiative module itself which will be described in section 5.3. Finally some suggestions for completely new but somehow related projects are given in section 5.4.

5.1 Summary

In the past many computer-aided teaching systems have been developed. Some of these systems are dialogue-based, which means that the student has a dialogue with the system using natural language. These systems can focus on a wide range of subjects, for example computer science or physics. There are also teaching systems which focus on language teaching in particular, but most of these systems are not dialogue-based. In this report a module has been described which combines these two things: it generates teaching initiatives in a computer-aided language learning dialogue.

In order to generate an initiative different characters are used. Each character has a number of structures with which it can create genuine questions, assertions and teaching questions. After some possible initiatives have been

generated the system decides which one is best based on a number of criteria. Using these criteria the system pursues two goals: it tries to create a coherent dialogue and it tries to teach the student a number of syntactic constructions.

The examples in the previous chapter firstly show that the system creates reasonably coherent dialogues. Furthermore it is made sure that all initiatives taken by the system contribute to the goal of teaching the student a number of syntactic constructions. This means that *every* initiative contains certain rules to check if the student knows the corresponding syntactic constructions. At the end of the example dialogues the student has learnt all rules properly and the lesson is ended.

5.2 Changes to Te Kaitito and the grammar

In this section some small changes to the system or to the grammar will be described. These changes don't require changes in the code for the initiative module, but will just make the initiative module work better.

5.2.1 Person, animal, place and animate features

First of all it would be useful if the system would know which predicates are people. Right now the person who authors the new lessons has to enter questions for each kind of person (e.g. "What is a man's name?", "What is a girl's name?" etc.) which takes a long time and which is probably against the author's intuition. It would be much better if the author could just enter the question "What is a person's name?" and that the system could find out that a man is a particular kind of person.

It could also be useful if a similar feature would exist for animals. Even though it is not necessary, the system might perform better if it would support this. There is a number of questions which applies to all animals (e.g. "What is the animal's color?" "What is the animal's size?" etc).

A third addition which is similar to the last two additions is that it would be useful if the system could tell whether a predicate is a place or not. The current system turns the sentence "John is born in Amsterdam" into the question "What is John born in?" instead of the better question "Where is John born?". If the system could distinguish between these cases the generated initiatives would sound more naturally to the user.

A final addition is adding an animate feature. As described in section 3.2.3 it is not possible to find out whether an entity is an animate object or not. If there are person and animal features this feature is not necessary anymore, but otherwise it could be used to create interesting no-questions.

5.2.2 Extending referring expression generation code

A somewhat more complicated change would be to extend the code for referring expressions. Right now the code only includes the necessary adjectives, but it

does not support substructures. The following dialogue could happen using the current system:

```
S: My mother is happy
S: Your mother is sad
S: Who is happy
Tk: The mother is happy
```

The system cannot distinguish both mothers just by adding some adjectives, because there are no adjectives associated with the mothers. Instead the words 'my' and 'your' are stored as substructures for the word 'mother'. Because the code for creating referring expressions simply ignores substructures the system is not able to answer this question properly. A consequence of this is that some sentences are turned into questions which look rather strange at first sight. An example is the sentence "The dog's color is green" which is turned into the questions "What is the color?" and "What is green?". These questions are generated because 'color' is the subject of the sentence and 'dog' should actually be in its substructure but is ignored. This problem is rather difficult to solve without changing the code for referring expressions and has therefore been ignored.

5.3 Additions to the Initiative Module

In this section some other additions to the current system and the grammar will be described which will make the system function better. These additions do however also require changes in the initiative module and are therefore described in a separate section.

5.3.1 Adding cause to the grammar

First of all the currently used grammar doesn't support reasons, explanations and other complicated structures. If the grammar however would support those structures the initiative module could also generate questions like "Why?" and "How?". At first sight that doesn't seem necessary, but the following example shows why the system will create better questions when they are supported: If the student enters the sentence "The man is angry", the current system generates a genuine question like "What is the man's name?". It would be much better if the system would come up with a question like "Why is the man angry?", because the main event of the sentence is not 'man' but 'angry'. Extending the initiative module in order to support genuine questions like the ones mentioned before is very easy. It just requires one extra rule which makes it possible to enter a question like "If the main event is an action verb, ask why". Adding this extension however requires changing the grammar in such a way that it supports more complicated sentences which will be difficult but also very useful.

5.3.2 Multi-speaker dialogues

At the moment there is some work ongoing to make the Te Kaitito system a multi-speaker dialogue system. In this way the student could talk to different characters at the same time. If the Te Kaitito system would support multi-speaker dialogues the initiative module could be run for each of the characters. Each character returns his or her best initiative and afterwards these initiatives have to be compared. An additional criterion might be needed to make sure that the different speakers speak in a balanced way. Adding the multi-speaker function to the initiative module in this way is rather easy, but there are other ways to incorporate the multi-speaker function into the initiative module which will be described in section 5.4.4.

5.3.3 Criterion based on 'naturalness' of the sentence

Finally it would be very useful if the system would support one more criterion. To generate the actual sentence in natural language every MRS is turned into all possible sentences. At the moment the code is being written to decide which of those sentences is the most natural one. For example the MRS for the sentence "John is from Holland" can be turned into the following four sentences:

- John is from Holland
- John is being from Holland
- From Holland john is
- Holland John is from

The sentence "I am born in Amsterdam" can even be turned into 22 different sentences, including the sentences containing the form "Yours truly" instead of "I". Obviously some sentences are used more frequently than others and therefore it would be a good idea to assign higher scores to sentences which are more natural than sentences which contain rare structures. When the code for ranking the sentences will be finished the criterion can be added and a score can be assigned at the same time as the syntax score is assigned.

5.4 Suggestions for new projects

During the design phase of a project new ideas for all different kinds of projects pop up. In this last section some suggestions for related projects are given.

5.4.1 Extending the discourse markers algorithm

The initiative module described in this report includes an algorithm to decide whether it is necessary to add the word 'also'. There are however many more discourse markers, some of which are also necessary and some of which are

only desirable. Examples are discourse markers like 'but', 'however' and 'by the way'. Much research has been done into discourse markers which can be useful for a project like this. In the current system discourse markers like 'but' and 'however' cannot be supported, because the grammar doesn't support those complex sentences. If the grammar however would support this kind of sentences adding discourse markers can be an interesting follow-up project.

5.4.2 Handling student's grammatical errors

Another possible project is to make a module to handle student's grammatical errors. If the system generates a question and the student makes an error trying to answer the question, the answer cannot be parsed and is therefore incorrect. It would be an improvement if another module would pop up and focus on the error made. This can be done by starting a subdialogue to correct the grammatical error and afterwards the system can return to the normal dialogue to check if the (grammatically correct) sentence was the correct answer to the question.

People who are just starting to learn a second language always make many mistakes. Research has shown that it is better not to correct all errors the very first time, because students can feel offended which can reduce the learning process. An additional project of handling the student's grammatical errors could focus on deciding when to correct a student's mistake.

5.4.3 Modelling the student's grammatical knowledge

The CALL-system is used to assist people learning a second language. The initiative module detects the student's difficulties and focusses on these difficulties. It does so by simply preferring the initiatives which contain syntactic rules that the student does not know yet. However, if the student doesn't know a particular rule and the system uses the rule again, this does not necessarily mean that the student will understand the rule this time. Therefore it would be useful if the system could use the rules in different contexts. If the student understands the rule in the new context, the system could check if the student really learnt the rule by checking if the student understands the rule in the old context.

Furthermore the student's grammatical knowledge is measured using the syntactic constructions used in the utterances. There may be additional ways to measure this knowledge which can be combined to get a better model of the student's knowledge.

5.4.4 Multi-speaker dialogues using agent technology

In the previous section some notes are given on how the initiative module can be used in a multi-speaker environment. There are however different ways to support this. The first one was described in section 5.3.2, but a more interesting one is to include agent technology. Using this technology the different speakers

can communicate with each other and cooperate to construct a better dialogue. The different characters can for example talk to each other and combine their private knowledge bases to extend the strategy criterion. Doing this they can decide if one of the characters can make an assertion which makes an assertion of the other character a good initiative.

Bibliography

A and Flickinger, D. (2000), On building a more efficient grammar by exploiting types, *in* 'Natural Language Engineering', pp. 15–28.

Bayard, I., Knott, A. and Moorfield, J. (2002), Syntax and semantics for sentence processing in english and māori, *in* 'Proceedings of the 2nd Australasian Natural Language Processing Workshop', Canberra Australia, pp. 33–40.

Carbonell, J. R. (1970), *AI in CAI: An Artificial Intelligence Approach to Computer-assisted Instruction.*

Copestake, A. and Flickinger, D. (2002), An open-source grammar development environment and broad-coverage english grammar using hpsg, *in* 'Proceedings of LREC 2000', Athens, Greece.

Copestake, A., Flickinger, D. and Sag, A. (1999), Minimal recursion semantics: An introduction, CSLI, Stanford University.

Freedman, R. (1997), Degrees of mixed-initiative interaction in an intelligent tutoring system, Department of CSAM, Chicago.

Graesser, A., VanLehn, K., Rose, C., Jordan, P. and Harter, D. (2001), Intelligent tutoring systems with conversational dialogue, University of Memphis and University of Pittsburgh.

Grosz, B. J., Joshi, A. K. and Weinstein, S. (1995), 'Centering: A framework for modeling the local coherence of discourse', *Computational Linguistics* **21**(2), 203–225.

Guinn, C. I. (1996), Mechanisms for mixed-initiative human-computer collaborative discourse, Department of Computer Science, Duke University, Durham.

Jurafsky, D. and Martin, J. H. (2000), *Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*, Prentice Hall. University of Colorado.

Kamp, H. and Reyle, U. (1993), *From Discourse to Logic*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Kim, J. H., Glass, M., Freedman, R. and Evens, M. E. (2000), Learning the use of discourse markers in tutorial dialogue for an intelligent tutoring system, Department of Computer Science, Illinois Institute of Technology, Chicago.

Knott, A., Bayard, I., de Jager, S. and Wright, N. (2002), An architecture for bilingual and bidirectional nlp, *in* 'Proceedings of the 2nd Australasian Natural Language Processing Workshop (ANLP 2002)'.

Knott, A., Moorfield, J., Meaney, T. and Ng, L. (2003), A human-computer dialogue system for mori language learning, *in* 'Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA)', Hawaï.

Lightbown, P. M. and Spada, N. (1993), *How Languages are Learned*, Oxford University Press, chapter 5 Observing second language teaching, pp. 91–101.

Moorfield, J. (1988), *Te whanake 1: Te kākano*, Addison Wesley Longman New Zealand, Auckland.

Mostow, J., Beck, J., Bey, J., Cuneo, A., Sison, J., Tobin, B. and Valeri, J. (2004), Using automated questions to assess reading comprehension, vocabulary and effects of tutorial interventions, Project LISTEN, Carnegie Mellon University, Pittsburgh.

Nakata, T., Ikeda, T., Ando, S. and Okumura, A. (2002), Topic detection based on dialogue history, Multimedia Research Laboratories, NEC Corporation, Japan.

Shomoossi, N. (2004), The effect of teacher's questioning behavior on efl classroom interaction: A classroom research study, Tehran University, Iran.

Vlugter, P., Knott, A. and Weatherall, V. (2004), A human-machine dialogue system for CALL, *in* 'Proceedings of InSTIL/ICALL 2004: NLP and speech technologies in Advanced Language Learning Systems', Venice, pp. 215–218.

Appendix A

Propositions and common ground

The following is an example proposition of the sentence “A dog chased a cat”:

```
#S(PROPOSITION
    :NUCLEUS h5:e4:{ h11:"_chase_v_rel"(e4,x7,x12)
                     h13:_a_q_rel(x12,h15,h14)
                     h6:_a_q_rel(x7,h9,h8)
                     h16:"_cat_n_rel"(x12)
                     h10:"_dog_n_rel"(x7) }
    :PRESUPS NIL))
```

This is an example proposition of the sentence “The dog chased the cat”. You can see that the presuppositions are stored in the presups instead of the top-nucleus.

```
#S(PROPOSITION
    :NUCLEUS h5:e4:{ h11:"_chase_v_rel"(e4,x9,x12) }
    :PRESUPS (#S(PRESUP
                 :NUCLEUS h13:x12:{ h13:_the_q_rel(x12,h15,h14)
                                    h16:"_cat_n_rel"(x12) }
                 :PRESUPS NIL)
              #S(PRESUP
                 :NUCLEUS h6:x9:{ h6:_the_q_rel(x9,h8,h7)
                                  h10:"_dog_n_rel"(x9) }
                 :PRESUPS NIL))))
```

This is an example proposition of the sentence “The cat was afraid”:

```
#S(PROPOSITION
    :NUCLEUS h19:e18:{ h25:"_afraid_j_rel"(e18,x23) }
    :PRESUPS (#S(PRESUP
                 :NUCLEUS h20:x23:{ h20:_the_q_rel(x23,h22,h21)
                                    h24:"_cat_n_rel"(x23) }
                 :PRESUPS NIL))))
```

The following is an example of the question "What chased the cat?". You can see that one of the presups in the original proposition is turned into a parameter and the rest of the proposition is the same.

```
#S(QUESTION
    :PARAMS (#S(PARAM
                :NUCLEUS h8:x7:{ h8:which_q_rel(x7,h9,h10) h6:thing_rel(x7) }
                :PRESUPS NIL))
    :PROP #S(PROPOSITION
                :NUCLEUS h11:e4:{ h12:"_chase_v_rel"(e4,x7,x13) }
                :PRESUPS (#S(PRESUP
                             :NUCLEUS h14:x13:{ h14:_the_q_rel(x13,h16,h15)
                                                h17:"_cat_n_rel"(x13) }
                             :PRESUPS NIL)))))
```

The common ground containing the two propositions "The dog chased the cat" and "The cat was afraid" is stored in the following way:

```
#S(DRS
    :REFERENTS (x9 x12 x1 x0)
    :CONDITIONS (h2:named_rel(x0,te kaitito)
                 h11:_chase_v_rel(e4,x9,x12)
                 h16:_cat_n_rel(x12)
                 h10:_dog_n_rel(x9)
                 h25:_afraid_j_rel(e18,x12)))
```

The common ground can also be represented as a split box in the way DRSs are normally represented:

$x9$ $x12$ $x1$ $x0$
name($x0$, "te kaitito")
chase($e4$, $x9$, $x12$)
cat($x12$)
dog($x9$)
afraid($e18$, $x12$)

Appendix B

Parameters

In this appendix all parameters used in the algorithms are described and their values are given. The parameters can be subdivided into three different kinds of parameters which will be described in the following subsections.

B.1 Parameters to calculate the separate scores

The first parameters are the parameters necessary to calculate the separate scores which include the following:

focus-start-score

The elements in the list of possible topics are ranked to calculate the similarity between the list of possible topics and the topics of a possible initiative. The first element has a score of *focus-start-score* and the following elements have a score of one lower.

If there are assertions which include the word 'also' they are chosen for sure, so this score is mostly applied to assertions which belong to one of the other assertion categories. A value of 5 has been chosen, because a higher value would result in less influence of the other elements of the focus score. A lower value than five on the other hand will result in less difference between the first and the last element in the list and that is also not the intention.

focus-subject-score

This score is added to the focus score if the subject of the initiative is in the list of possible initiative, so if the subject is on focus. A value of 7 has been chosen because it is much better if the subject appears in the list, but it is for example less important than adding the word 'also'. Furthermore part of the score has already been assigned when calculating the similarity score.

focus-speaker-score

This score is added when one of the speakers is mentioned in the initiative. This is done because the speakers are always on focus and should therefore always be in the list with possible topics. A value of 7 has been chosen for the same reason as the focus-subject-score.

focus-also-score

This score is assigned when the word 'also' can be added to the initiative which means that the initiative and the previous utterance are very similar. Such an initiative will already have a high similarity score, but because the similarity score looks at the previous two sentences and the 'also' score looks only at the very last sentence this score is really important. Therefore a value of 10 has been chosen.

order-start-score

This score represents the value that is assigned to the question generation rule on a particular topic that has been entered first. Presumably, there won't be many questions about the same topic and if there are many questions they are probably not ordered in a very strict way. The criterion is just meant to prefer the very first question to the later questions. Therefore only a score of 3 has been chosen.

order-new-entity-score

This score is used to prefer questions about new entities. The order scores are all quite low, so a value of only 1 has been chosen.

order-subject-score

This score makes sure that questions about the subject of the previous sentence are more likely to be chosen than other questions. For the same reason as the previous score a value of 1 has been chosen.

order-no-speaker-score

This score applies to genuine questions which are about entities that are not the student and also a value of 1 has been chosen. The last three scores represent the focus score for genuine questions, so by choosing a value of 1 for all of them the focus score simply consists of counting the number of conditions that the question satisfies.

strategy-nr-preds-score

This parameter represents the number of predicates that is looked at when calculating the number of genuine questions which are about the topics in the initiative. Only n-preds are looked at and in most cases there are only one or two predicates in a sentence, so a value of 2 has been chosen. There are some verbs that have three arguments (e.g. 'give') or even four (e.g. 'exchange'), so if the entities used in sentences like these also have to be used the value of the parameter should be increased. A value of 2

has been chosen because in this way the subject is strongly preferred to the object.

strategy-nr-refs-score

This parameter represents the number of referents that is looked at when calculating the number of facts in the knowledge base which contain one of the referents. The value for this parameter is 2 for the exact same reasons as the value of the *strategy-nr-preds-score*.

semantics-wh-score

This score is assigned if the teaching question is a wh-question instead of a yes/no question. A value of 1 has been chosen to make sure that these questions and the wh-questions are mixed in a balanced way.

semantics-yes-score

This score is assigned if the teaching question is a yes-question. A value of 1 has been chosen to make sure that the wh-questions and the yes/no-questions are mixed in a balanced way.

semantics-no-score

This score is assigned if the teaching question is a no-question. Only no-questions which are on topic are generated, so often there are no no-questions at all. A value of 3 has been chosen to make sure that a no-question has a big chance to be chosen if there is one.

semantics-different-score

This score is assigned if the teaching question uses a referring expression which is different from the one used when introducing the corresponding entity. A question will be harder if the referring expressions are different and therefore a score of 2 has been chosen.

semantics-adjectives-score

Another aspect of the semantics score is the complexity of the used referring expression which is expressed by the number of adjectives necessary in the referring expression. First the number of adjectives is determined and that number is multiplied by the *semantics-adjectives-score*. A value of 1 has been chosen which means that it has the same effect when the referring expression has two adjectives as when the referring expressions are different.

semantics-new-question-score

This score is added if the proposition corresponding to the question has never been turned into a question before. A value of 1 has been assigned to make sure that these questions are preferred only a little bit.

semantics-balanced-score

This score is assigned to questions of a particular kind if the previous two teaching questions were of the other kind. This means that wh-questions get this score if the previous two questions were yes/no-questions and the other way around. A value of 2 has been chosen, because this way the preferred kind is chosen quite often.

syntax-start-score

This score represents the initial values of the target syntactic rules. Everytime a rule is used correctly the values are decreased by one. A value of 4 is chosen because this means that a rule has to be used correctly for four times before its value reaches zero. If the value of this parameter will be higher the lessons will take somewhat longer.

B.2 Weight parameters

After a number of initiatives have been selected and their syntax scores have been calculated the different scores have to be combined to get a total score. Genuine questions have an order score and a syntax score, assertions have a strategy score, a focus score and a syntax score and teaching questions have a semantics score and a syntax score. This means that always only two or three scores have to be combined and therefore the total score is determined by taking the weighted sum of the separate scores. It is implemented using the parameters *order-weight*, *strategy-weight*, *focus-weight*, *semantics-weight* and *syntax-weight*. Currently, all parameters have a value of 1 which means that all criteria are just as important. Some other values have been tried, but the results were similar or worse, but because the weights have been implemented as parameters the values can be changed.

B.3 Remaining parameters

The final group of parameters consists of parameters which can be changed to improve the efficiency and parameters which are used for user friendliness. Some of these parameters have boolean values instead of numerical values.

pruning-teaching

If this parameter is set to t the teaching questions will be pruned. This parameter was set to *nil* during the testing phase, but in the final system the parameter is set to t.

pruning-syntax

This parameter shows whether pruning the number of initiatives for which the syntax score is calculated has to be applied or not. Like the other pruning parameter this parameter was set to *nil* while testing and will be set to t during actual use.

nr-syntax

If *pruning-syntax* is set to t only the initiatives with the highest total scores are passed to the next level to calculate the syntax score. This parameter represents the number of initiatives which will be passed to this next level at a time. Right now the value 4 is used because in most cases this will result in an acceptable response time. If the value is much higher the response time will not differ much from the result without applying this kind of pruning. If the value is lower the chance is rather large that an initiative with a really high syntax score is not chosen because it scores somewhat lower in the other criteria.

syntax-fail-score

If one of the values of the target syntactic rules is higher than *syntax-fail-score* the system decides to stop the lesson, because the student has made too many mistakes. The value of 8 has been chosen, because this means that the student has made at least four mistakes without using the rule only once.

minimum-nr-tqs

This parameter represents the number of teaching questions that is asked before a genuine question about the student or an assertion about the system will be chosen. Right now the value 3 is used, because this way some teaching questions are asked first and therefore the kinds of initiatives in the final result will be more balanced.

print-topics

If this parameter is set to t the list with possible topics is printed. This is not done in the final system, but can be useful when testing.

print-initiatives

If this parameter is set to t a list with possible initiatives and a table with the corresponding scores is printed. This is not used in the final system, but is very useful when deciding which values are best.

print-initiative-sentences

This parameter is the same as *print-initiatives* except that it prints the possible initiative sentences. The parameter is also set to *nil* in the final system.

B.4 Parameter values table

In the following table the different parameters and the corresponding values used in the final system are given.

Parameter	Value
focus-start-score	5
focus-subject-score	7
focus-speaker-score	7
focus-also-score	10
order-start-score	3
order-new-entity-score	1
order-subject-score	1
order-no-speaker-score	1
strategy-nr-preds-score	2
strategy-nr-refs-score	2
semantics-wh-score	1
semantics-yes-score	1
semantics-no-score	3
semantics-different-score	1
semantics-adjectives-score	1
semantics-new-question-score	1
semantics-balanced-score	2
syntax-start-score	7
order-weight	1
semantics-weight	1
strategy-weight	1
focus-weight	1
syntax-weight	1
pruning-teaching	t
pruning-syntax	t
nr-syntax	4
syntax-fail-score	8
minimum-nr-tqs	3
print-topics	nil
print-initiatives	nil
print-initiative-sentences	nil