# Model Checking for Contextual Reasoning in NLG

Kristina Striegnitz
Computational Linguistics
University of the Saarland, Germany
kris@coli.uni-sb.de

September 3, 2001

**Abstract**

Presupposition triggers, such as e.g. *the*, *too*, *another*, impose constraints on the context they are used in. A violation of these constraints results in an infelicitous utterance. A natural language generation system therefore has to reason on the context to check that they are satisfied. We argue that this kind of contextual reasoning is essentially a model checking task and demonstrate this for a variety of presupposition triggers. To account for the influence of some background knowledge, we propose to embed queries to a description logic knowledge base in a first order model checking algorithm.

## 1 Introduction

The production of natural language takes place against a certain background context containing information about the previous discourse, the current situation, and the knowledge of speaker and hearer. This requires a natural language generation (NLG) system to reason on this context in order to produce natural sounding, pragmatically acceptable text (see e.g. Hovy 1988; Danlos 1987; Appelt 1985; Stone 2000 for work on pragmatics effects in NLG). In particular, there are certain lexical items that impose restrictions on the context they may be used in felicitously. So called presupposition triggers, such as *the*, *another* or *too*, for instance, exhibit this behavior (Levinson 1983). The following examples illustrated this.

(1) a. There is a rabbit in my vegetable garden. It's eating all of my carrots. **Another rabbit** is digging holes into my flowerbed.

b. ?? There is a rabbit in my vegetable garden. It's eating all of my carrots. **Another mole** is digging holes into my flowerbed.

c. There is a rabbit in my vegetable garden. It's eating all of my carrots. **Another animal** is digging holes into my flowerbed.

The expression *another rabbit* in (1a) e.g. requires – or presupposes – the existence of a rabbit in the context which is different from the one that's digging

the holes. Similarly, *another mole* in (1b) requires the existence of a second mole in the context. This is not given in the example, which makes the utterance infelicitous. In (1c), the presupposition of *another animal* is satisfied, since there is a rabbit in the context and we know by world knowledge that rabbits are animals.

In this paper, we will examine the inferences necessary to check the contextual constraints associated with presupposing expressions. We argue that it is essentially a model checking task and show how interactions with world knowledge as exhibited by (1c) can be captured by embedding queries to a description logic knowledge base in a first order model checking algorithm.

**Plan of the paper.** In the following section, we examine the contextual inferences that an NLG system would have to carry out in generating presupposition triggers and show how first order model checking can be used to carry out these inferences as long as all relevant information is specified in the discourse model. However, it is not always natural to code all of the relevant information as a semantic model. In Section 3, we therefore present a model checking algorithm that may query a description logic knowledge base on simple properties of discourse model elements, so that some background knowledge can be taken into account. Section 4 briefly summarizes some features of model checking that make it well suited for contextual reasoning in natural language processing and compares the use of description logic for representing the necessary world knowledge to other possibilities. In Section 5, we describe a generation architecture which interleaves the generation process with queries to the model checker and show how presupposition triggers are treated in this architecture. Finally, in Section 6, we relate our approach to other work on presuppositions and NLG and point out some issues that we plan to investigate into in the future.

## 2 Presuppositions from an NLG Perspective

As we have seen in the introduction, presupposition triggers interact with the context they are uttered in, as they can only be used felicitously, if the context satisfies certain constraints.

In this paper, we will assume the context to look as follows: The speaker, or the generation system, has a model representing the situation which is to be described. The discourse model is a sub-model of this speaker's model and contains the information that has already been communicated in the previous discourse. The system therefore assumes the discourse model to be shared between speaker and hearer. As the discourse evolves this shared part of the speaker's model grows. At the moment, we simply assume that whenever an utterance is made its the semantic content is added to the discourse model[1]. Furthermore, there may be some additional general knowledge about the world,

[1]Note, that in this paper, we are looking at intersentential/interutterance anaphora only, so that taking the discourse model to contain the information conveyed by previous utterances in the discourse is sufficient. To be able to deal with sentence internal anaphora, the discourse model would have to be extended with information conveyed by the currently produced utterance as well, but we haven't worked out the details of this update, yet.

which is also assumed to be shared knowledge. (2) gives an example for such a context. The models are represented by a (finite) list of individuals and a (finite) list of facts describing properties of these individuals. Individuals are taken to be pairwise distinct. For the moment, we will not allow additional world knowledge, so that we can assume that whatever is not explicitly listed of an individual is false and that the list of individuals is exhaustive.

(2) **speaker's model** $(\{a, e_1, b, e_2\}, \{sleep(e_1), rabbit(a), agent(e_1, a), eat(e_2), rabbit(b), agent(e_2, b), carrot(c), theme(e_2, c)\})$

**discourse model** $(\{a, e_1\}, \{sleep(e_1), rabbit(a), agent(e_1, a)\})$

**world knowledge** $\{\forall x[rabbit(x) \rightarrow animal(x)]\}$

The generation task can be summarized as follows: given some communicative goal $G$ and a context $C$ the generator must produce a text $T$ which achieves $G$ in $C$. Following Stone and Doran (1997) we will assume communicative goals consisting of a) an instruction of the form *describe* $X$, where $X$ is an eventuality, and b) a collection of facts (from the speaker's model) that are to be communicated by the target utterance $T$.

Thus, an input to our generation system may look like this (we will leave the speaker's model implicit, whenever it doesn't contain any relevant information that's not part of the discourse model or the set of facts that is to be communicated):

(3) a. **Goal:** describe $e$

**To Communicate:** $\{sleep(e),\ agent(e, a)\}$

**Discourse Model:** $(\{a\},\ \{rabbit(a)\})$

b. **Goal:** describe $e$

**To Communicate:** $\{sleep(e),\ agent(e, a),\ rabbit(a)\}$

**Discourse Model:** $(\{b\},\ \{rabbit(b)\})$

That means, the communicative goal of the system is to describe a sleeping event involving a rabbit. In case of Input (3a), this rabbit is one that has been mentioned before and is therefore an element of the discourse model, while in the situation described by (3b) the sleeping rabbit is new to the discourse, but the discourse model contains another rabbit.

In order to describe event $e$, the system has to describe entity $a$ at some point during generation. There are various ways of doing so. Among them are producing *the rabbit* or *another rabbit*. However, both of these expressions trigger presuppositions which place restrictions on the context the respective expression can be used in. *The rabbit* presupposes that there must be an entity $u$ in the discourse model, such that $rabbit(u)$ & $u = a$, where $a$ is the referent of the NP, and *another rabbit* presupposes that there must be an entity $u$ in the discourse model, such that $rabbit(u)$ & $u \neq a$. We will call this entity $u$ the *anchor* of the presupposition[2].

[2]This follows van der Sandt's (1992) account of presuppositions as anaphora. However,

So, in order to satisfy the presuppositional constraints of *the* and *another* the discourse model must satisfy one of the following formulas: $\exists x[rabbit(x)\ \&\ x = a]$ for *the rabbit* and $\exists x[rabbit(x)\ \&\ x \neq a]$ for *another rabbit*. And this is exactly the type of query a model checker can answer: given a model $M$ and a formula $\phi$, is $\phi$ true in $M$?

Applying this to the input given in (3a), the queries that the generator sends to the model checker are

$$(\{a\},\ \{rabbit(a)\})\ \models_?\ \exists x[rabbit(x)\ \&\ x = a]$$

which is true licensing the production of *the rabbit* and

$$(\{a\},\ \{rabbit(a)\})\ \models_?\ \exists x[rabbit(x)\ \&\ x \neq a]$$

which is false preventing the production of *another rabbit*. For input (3b), on the other hand, *another rabbit* is licensed for referring to entity $a$, while *the rabbit* is not generated.

Some expressions pose further constraints on the context. The definite article, e.g, requires that the description should uniquely identify the referent. That means, in the case of *the rabbit*, there must not be an entity different from the referent in the discourse model that also is a rabbit. *Another*, on the other hand, require that the referent be new to the discourse model. These constraints can also be checked by model checking. In case of situation (3a) the corresponding queries would look like this: $(\{a\},\ \{rabbit(a)\})\ \models_?\ \forall x[rabbit(x) \rightarrow x = a]$ for *the rabbit* and $(\{a\},\ \{rabbit(a)\})\ \models_?\ \neg\exists x[x = a]$ for *another rabbit*.

Other presupposition triggers can be treated in the same way. Below is a table listing triggers and the corresponding formulas that have to be checked for truth in the discourse model. The pronoun *he*, for instance, presupposes the existence of an entity that is male and in focus, where we assume that $infocus(u)$ only holds of a discourse model entity $u$, if $u$ is in focus in the sense of Gundel et al. (1993) and therefore salient enough to be referred to by a pronoun. *too* presupposes the existence of an event $x$ that is of the same type as the event $e$ described by the VP modified by *too* and requires $x$ to be different from $e$. Similarly, *again* presupposes an event $x$ that shares the properties determined by the VP semantics with $e$, but is different from $e$ and takes place later than $e$.

---

at the moment, every presupposition must have an anchor in the discourse model which has all of the properties specified in the presuppositional constraint, i.e., for now, we don't allow accommodation or partial matching, since, without restriction, this would lead to overgeneration.

| trigger | presupposition | contextual constraint |
|---|---|---|
| the $N(a)$ | $\exists x[N(x)\ \&\ x = a]$ | $\forall x[N(x) \rightarrow x = a]$ |
| another $N(a)$[3] | $\exists x[N(x)\ \&\ x \neq a]$ | $\neg\exists x[x = a]$ |
| he($a$) | $\exists x[male(x)\ \&\ infocus(x)\ \&\ x = a]$ | $\forall x[(male(x)\ \&\ infocus(x)) \rightarrow x = a]$ |
| $VP(e)$, too | $\exists x[VP(x)\ \&\ x \neq e]$ | |
| regret that $S(e)$ | $\exists x[S(x)\ \&\ x = e]$ | |
| $VP(e)$ again | $\exists x[VP(x)\ \&\ time(x) \leq time(e)]$ | |

## 3 Adding Some Background Knowledge

In Section 2, we have looked at cases where the presupposed material could be found directly in the discourse model. Now, we will consider cases where some world knowledge has to be taken into account to establish that the presupposition is satisfied in the context.

The presupposition triggered by *another animal* in Example (4a) for instance is that there is an entity in the context which is an animal and different from the one that *another animal* refers to. In the example, this presupposition is satisfied, by the rabbit introduced in the first sentence, since we know by world knowledge that rabbits are animals. Similarly, the presupposition triggered by *too* in the second sentence of (4b), because (by world knowledge) *watching Casablanca* is subsumed by *watching a movie.*

(4) a. There is a rabbit in the garden. It is eating the vegetables. *Another animal* is digging holes into the flowerbed.

b. John is going to watch 'Casablanca', tonight. Peter *is going to watch a movie, too.*

But how does our model checking approach to contextual reasoning deal with cases like this? It doesn't because it cannot take into account background information which is not part of the discourse model, i.e. which is not mentioned explicitly in the previous discourse.

Assume the following situation:

(5) a. **G:** describe $e_1$

b. **ToC:** $\{pull\_out(e_1),\ agent(e_1, a_1),\ animal(a_1),\ theme(e_1, a_2)\ldots\}$

c. **DM:** $(\{e_2, a_3, a_4\ldots\},\ \{eat(e_2),\ agent(e_2, a_3),\ rabbit(a_3)\ldots\})$

That is, the goal is to describe an event which involves an animal, entity $a_1$. When deciding which expression to use to refer to $a_1$, the generator sends the

[3] As Bonnie Webber has pointed out to me, this is a rather strict definition of the presupposition associated with *another* and will have to be refined, when we know more about the way certain parts of the $\overline{\text{N}}$ add or don't add to the requirements imposed on the anchor. While premodifiers, for instance, usually are included in the presupposition, postmodifiers not necessarily have to be.

queries associated with their presuppositions to the model checker. Contrary to what we saw above, the model checker will determine that *another animal* is not licensed, since there is no entity in the discourse model which is different from $a_1$ and known to be an animal. In particular, the information that $a_3$ is an animal is not contained in the discourse model. Additional world knowledge which let's the system infer that anything known to be a rabbit is also an animal and that Casablanca is a movie has to be taken into account. Given such a representation of world knowledge, the generation system has to check that there is an entity in the discourse model of which we know by the information in the discourse model together with world knowledge that it is an animal or a movie watching event. So, the discourse model entities have to be inspected and then the system has to check whether a specific discourse model entity has certain properties, given the discourse model and the world knowledge. That means that we are weakening our closed world assumption: every property neither given explicitly in the model *nor entailed by world knowledge* is taken to be false.

We propose to use description logics to represent the world knowledge. Description logics are designed for knowledge representation. They are not as expressive as first order logic but are usually restricted to a decidable subset of first order logic. Modern description logic provers like RACER (Haarslev and Möller 1999) or FaCT (Horrocks 1999) are fast and can efficiently handle even large sets of data. For the toy examples presented in this paper, this is not an issue, but it in any real application the amount of background knowledge needed will increase dramatically.

We will now give a very brief introduction to description logics and then present an algorithm that checks whether a formula is satisfied by a given discourse model when taking into account world knowledge for determining whether an entity has a certain property. To this end, the algorithm queries a description logic prover.

### 3.1 A Crash Course in Description Logic

We will now briefly explain the basic concepts of DL that are important for our application. For more information on DL see e.g. Donini et al. (1996) or Baader et al. (2001).

A description logic knowledge base consists of two parts: the T-Box and the A-Box. The A-Box contains information about individuals and how they are related, such as *rabbit(a)* or *part_of(a,b)* for instance. The discourse model can therefore be represented as part of an A-Box. The T-Box contains *concepts*, as e.g. *rabbit* and *animal*, which are related through a subsumption hierarchy, so e.g. $rabbit \sqsubseteq animal$. These concepts may be atomic or they may be complex, i.e. defined through other complex and relations between concepts. *Parent* can for example be defined as someone who is female and has a child ($mother \sqsubseteq female \sqcap \exists(has\_child).top$).

DL theorem provers usually offer a variety of different reasoning services. The one we are interested in is instance checking, i.e. checking whether an entity is an instance of a given concept. Given the T-Box $\{rabbit \sqsubseteq animal\}$ and the A-Box $\{rabbit(a)\}$, for instance, we can pose the query $KB \models_? animal(a)$,

which will be evaluated as true.

### 3.2 Combining first order model checking with DL classification

We have seen above that, in the case of Example (5), normal model checking cannot handle correctly the presuppositions associated with the description *another animal* for entity $a_1$, because it involves checking properties that are not specified in the discourse model itself, but may be given through some additional world knowledge. We will represent the world knowledge as a description logic knowledge base $KB$ where the discourse model is part of the A-Box. The algorithm then proceeds as for normal model checking until an atomic formula $F$, i.e. an expressions of the form $P(u)$ or $P(u, v)$ for some predicate $P$ and discourse model entities $u$ and $v$, is reached, where, instead of checking, whether $F$ is among the facts specified by the discourse model, we send the query $KB \models_? F$ to a DL theorem prover. The following pseudo code sketches this algorithm. *Formula* is the formula that is to be checked, $DMU$ is the universe of the discourse model, and *Context* the knowledge base.

Satisfy (*Formula*, $DMU$, *Context*)
  if *Formula* matches *P(x)* then true if *Context* $\models$ *P(x)*,
    false otherwise
  elseif *Formula* matches *P(x,y)* then true if *Context* $\models$ *P(x,y)*,
    false otherwise
  elseif *Formula* matches $x = y$ then true if $x$ and $y$ are the same entity,
    false otherwise
  elseif *Formula* matches $\neg F$ then false if Satisfies($F$,$DMU$,*Context*),
    true otherwise
  elseif *Formula* matches $F_1$ & $F_2$ then true if Satisfy($F_1$,$DMU$,*Context*)
      and
      Satisfy($F_2$,$DMU$,*Context*),
    false otherwise
  elseif *Formula* matches $F_1 \vee F_2$ then true if Satisfy($F_1$,$DMU$,*Context*)
      or
      Satisfy($F_2$,$DMU$,*Context*),
    false otherwise
  elseif *Formula* matches $F_1 \rightarrow F_2$ then true if Satisfy($\neg F_1$,$DMU$, *Context*)
      or
      Satisfy($F_2$,$DMU$,*Context*),
    false otherwise
  elseif *Formula* matches $\exists x.\ F$ then true if there is an $u \in DMU$, such that
      Satisfy($F_1[u/x]$,$DMU$, *Context*)
    false otherwise
  elseif *Formula* matches $\forall x.\ F$ then true if for all $u \in DMU$,
      Satisfy($F_1[u/x]$,$DMU$, *Context*)
    false otherwise

## 4 Reasoning on the Discourse Model

In work on NLG, presuppositions have mostly been considered only as part of the generation of referring expressions. One of the most influential works is Dale and Reiter (1995), which presents special purpose algorithms for this task. We can mimic their algorithms in our system by appropriately tailoring the way additional information is adjoined to noun phrases in case the uniqueness constraint of the definite article is not satisfied. Our main goal, however, was to generalize from the generation of definite descriptions to any kind of presupposition triggers. Furthermore, we extended their approach by taking into account some background knowledge.

We decided to represent the background knowledge by a description logic knowledge base.

The examples that we saw so far needed only a subsumption hierarchy of atomic concepts. But that won't be enough in general. Description logic lets us also define complex concepts, as is necessary to be able to deal with examples as (6a), and we can talk about relations between entities, as will be necessary for extending the approach to indirect anaphora as in (6b), where it is possible (at least in conversations between Harry Potter fans) to use a definite description to refer to *the snitch*, since a snitch is part of any quidditch match.

(6) a. The mother of my father lives in Frankfurt. **My other grandmother** lives in the black forest.

b. Gryffindor won the last quidditch match, because Harry caught **the snitch**.

Furthermore, we are able to use state of the art description logic theorem provers which are highly optimized and can deal with large sets of data. Description logic is decidable (in contrast to full first order logic) and, in general, description logic provers outperform first order logic theorem provers on many problems. However, if this will also be the case in our application when the world knowledge gets bigger remains to be tested.

In Stone and Doran's (1997) SPUD system, from which our approach is inspired, context is represented by means of modal logic formulas and uses logic programming techniques to perform inferences. His approach is more general and allows an elegant and fine grained representation of the context (via modal operators) and a fuller account of inferences in NLG. It allows, e.g., flexibility in formulating inferences over the knowledge of the hearer, as, for instance, a system for generating instructions would have to do in order to decide what information to include in the instruction and what information to leave implicit (cf. Stone 2000). We take a more restricted approach, concentrating on an appropriate treatment of the inferences that arise in an NLG system due to the contextual requirements of presuppositions. Our interest is to develop a map of the types of inference tasks in NLP associated with different linguistic phenomena and to examine which reasoning tools are best suited to carry them out.

In representing the discourse as a model we follow such work as Baumgartner and Kühn (1999), Ramsay and Seville (2000), and Gardent and Konrad (2000), who use model builders in natural language *interpretation* to construct models satisfying a given discourse, thereby resolving pronouns, definite descriptions and more generally, the underspecified items of natural language. Presuppositions and the like constrain what models are possible. From a natural language *generation* perspective the task is to make sure that, in the given model, the target utterance makes the hearer resolve such items in the intended way. Here, presuppositions constrain what utterances are possible in the given model, i.e. whether a certain utterance can be used in the given model depends on whether the presuppositional constraints are satisfied in that model. Using model checking is a natural choice for this task.

Furthermore, model checking allows a natural formulation of our closed world assumption (any property of an entity that doesn't follow from the knowledge base is taken to be false).

Finally, model checking enables a treatment of higher-order queries which are usually not amenable to the theorem-proving approach (Vardi 1986). In natural language processing higher order queries will arise for the treatment of expressions like *five rabbits are hopping through my garden.*

# 5 Generating Presupposition Triggers

In this section, we will show how the approach to accounting for the influence of presuppositions on NLG presented in the previous sections can be integrated into a generation system. The system that we will describe is based on Stone et al.'s (2001) SPUD system. It integrates microplanning and surface realization and interleaves the generation process with queries to a reasoning tool in order to check pragmatic constraints.

## 5.1 Syntactic, Semantic, and Pragmatic Constraints in the Grammar

Linguistic knowledge is encoded using the LTAG (Lexicalized Tree Adjoining Grammar) formalism. For a precise definition of LTAGs we refer the reader to Joshi and Schabes (1997). Here we briefly describe the basic properties of LTAG.

An LTAG consists of a set of tree fragments called *elementary* trees (the lexicon) and of two operations on trees called *adjunction* and *substitution*, that allow to build larger tree structures by combining two smaller ones.

There are two types of elementary trees in LTAG: *initial* trees (such as e.g. trees $\alpha_1$ and $\alpha_2$ in Figure 1) which are used to encode the basic syntactic frame of syntactic functors and *auxiliary* trees (such as e.g. tree $\beta 1$ in Figure 1) which encode modifiers e.g. adjectives, prepositional phrases (PP) or adverbs. Initial trees may contain so-called substitution nodes (marked with $\downarrow$), whereas auxiliary trees must have a unique foot node (marked with $\star$), i.e. a leaf node labeled with the same category as the root of the tree.

The two operations, substitution and adjunction, are then used to combine trees into bigger trees. Intuitively, substitution inserts an initial tree with root category $X$ at some substitution node $n$ with category $X$ in some other tree. Adjunction, on the other hand, caters for recursion and permits inserting an auxiliary tree with root and foot node category $X$ at a node labeled with category $X$ in some other tree. Substitution and adjunction are illustrated in Figure 1.

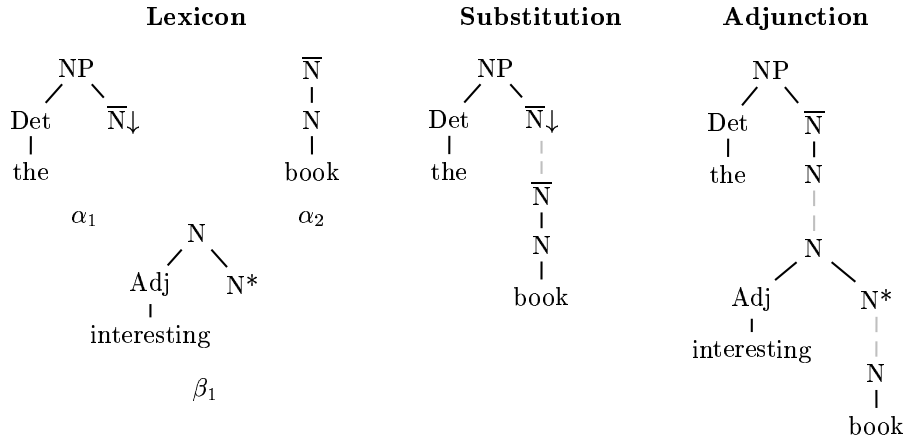


Figure 1: LTAG

In the lexicon, we follow Stone and Doran (1997) and associate semantic and pragmatic information with the elementary trees. This information is interfaced with the syntactic structure via the semantic entities that nodes in the syntax tree are associated with. The idea behind this is that syntactic constituents are associated with those semantic entities that are described by this constituent. The semantic information is coded in a flat semantic representation along the lines of Hobbs (1985).

Figure 2 shows some lexical entries for presupposition triggering expressions, introducing the presuppositions and constraints that we have seen in Section 2. $NP_{sem}$ stands here for a formula representing the semantic content of the NP, and similarly, $VP_{sem}$ stands for the semantic content of the VP. Figure 2 further shows the lexical entry for the indefinite article, which is no presupposition trigger and therefore doesn't have a presuppositional requirement, but carries a contextual constraint saying that the referent must be new to the discourse and the there is no other entity in the discourse model which fits the same description. This constraint prevents the generation of *a* where *the* or *another* should be used instead. In principle, it is too strict, though. For instance, since not in all contexts where *another* is possible, *a* is inappropriate.

## 5.2 Interleaving Generation with Contextual Reasoning

The generation process consists of two main phases. In the first phase, the generation process is driven by the initial communicative goal and the requirements that open substitution nodes may pose on the structure. The generator tries

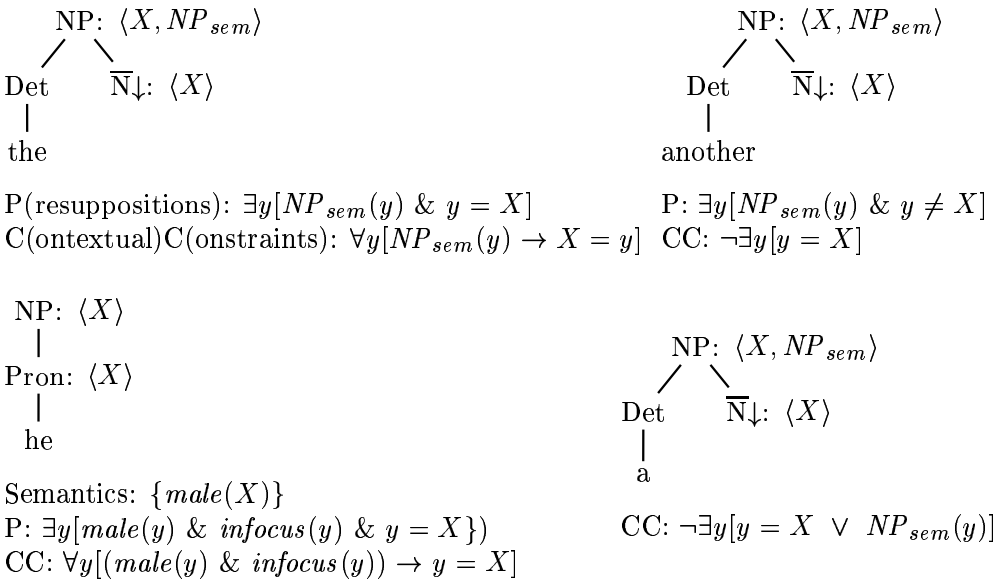


Figure 2: Some lexicon entries for presupposition triggers.

to build a *minimal, syntactically complete* (i.e. all substitution nodes are filled) linguistic structure by recursively substituting into the leftmost open substitution node. No adjunction takes place in this first phase. The resulting structure is then checked as to whether it is *semantically complete* (i.e. the content that was to be communicate is conveyed by the text) and *pragmatically appropriate* (i.e. the presuppositions and additional contextual constraints are satisfied). If that is not the case, additional information can be integrated by adjunction.

Figure 3 shows an example of how the generation algorithm assembles trees. The input to the generator for this example might have been as follows.

(7) **DM:** $(\{a_1,\ e_1\},\ \{rabbit(a_1),\ sleep(e_1, a_1)\})$
**Goal:** describe $e_2$
**To Communicate:** $\{eat(e_2, a_2, a_3),\ rabbit(a_2),\ carrot(a_3)\}$

Roughly, the generation process works in the following way. First, a prediction step selects elementary trees (such as e.g. $\alpha_1$) with the following properties from the lexicon that: (i) the root node is labeled with S and (ii) the semantic representation unifies with one or more of $e_1$'s properties. The result is not syntactically complete, hence, generation proceeds with the next goals determined by the open substitution nodes: a noun phrase describing entity $a_2$ and a noun phrase describing entity $a_3$ have to be found. In both cases, there are several semantically possible realizations, such as e.g. *the rabbit*, *another rabbit*, *the carrot*, *a carrot* etc. However, some of them trigger presuppositions and are only predicted if the contextual requirements of these presuppositions are fulfilled. Whenever a syntactically complete tree has been built, it is checked whether this tree is semantically complete and pragmatically appropriate. *Another rabbit is eating a carrot*, for instance, is semantically complete and in

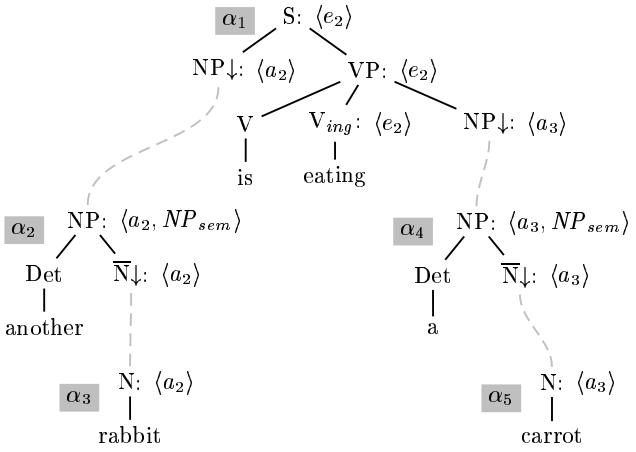


Figure 3: Generating Example (7).

order to check the pragmatic requirements the generator sends the following queries to the model checker.

| | |
|---|---|
| Presupposition of *another* : | $DM \models_? \exists x[rabbit(x)\ \&\ x \neq a_2]$ |
| Contextual Constraint of *another*: | $DM \models_? \neg\exists x[x = a_2]$ |
| Contextual Constraint of *a*: | $DM \models_? \neg\exists x[x = a_2 \ \vee\ carrot(x)]$ |

All of these queries return true, so that this is in fact a possible paraphrase and no adjunction of further information is necessary.

The production of *the rabbit* is ruled out, since the query $DM \models_? \exists x[rabbit(x)\ \&\ x = a_2]$, representing the presuppositional requirement of the definite article, will evaluate to false in the given context. Similarly, all other possibilities of expressing reference to $a_2$ and $a_3$ are ruled out.

Our system doesn't use a mechanism for ranking predicted items according to how likely they are, yet. Stone et al. (2001) propose a strategy of testing for violations of constraints which is similar to what's done in optimality theory. We plan to integrate a similar strategy into our system to better direct the search and to rank paraphrases in case there are several possibilities. In particular, this will also allow us to weaken the contextual constraints on the indefinite article.

## 6 Linguistic Coverage

There is extensive literature on presuppositions (Stalnaker 1974; Karttunen 1974; Heim 1983; van der Sandt 1992; Beaver 1997; Geurts 1999). Mostly, they are looked at from a language interpretation point of view, though. The task is therefore, to detect presuppositions and to find a way of anchoring them in the discourse context. The base case is that some entity in the discourse model satisfies or binds the presupposition. This is the case that we have restricted ourselves to in this paper. It is, however, a well known fact that presupposing

expressions can sometimes be used without an explicit anchor (Clark 1975; Lewis 1979). See, for instance, Examples (8a) and (8b).

(8) a. You are standing in front of a red brick house. **The door** is locked.

b. We **regret** that children cannot accompany their parents to commencement exercises.

In Example (8a), there is no element in the discourse model that satisfies the presupposition of *the door*. However, there is an implicit anchor in the discourse model, i.e. an element, namely the house, from which it follows that there must also be a door. In Gardent and Striegnitz (2001), we have identified reasoning tasks that arise in an NLG system in connection with such *indirect anaphora*. It remains to be seen, how the reasoning tasks we have specified there, fit into the more general treatment of presuppositions presented in this paper.

In Example (8b), on the other hand, there is neither an explicit nor an implicit anchor for the presupposition triggered by *regret* in the discourse model. Presupposition theories handle such cases by allowing to *accommodate* the presupposed material in the discourse model in case the presupposition cannot be bound. Usually, accommodation is not further restricted. However, in order to avoid serious overgeneration, an NLG system has to strictly control the use of accommodation. More research is needed here to determine, under which circumstances accommodation is allowed and natural. But see e.g. Zeevat (2000) for some insights on how different presupposition triggers behave wrt. accommodation.

Similar remarks also hold for examples as (9a) where the presupposition trigger in the second sentence conveys new information about its anchor and examples as (9b) where only part of the semantic content of the NP seems to be presupposed. The constraints we used in this paper are too strict to allow for that. They will have to be weakened, e.g. by excluding parts of the NP semantics from the presupposition in certain cases, or the way they are checked has to be weakened, e.g. by checking whether the fact that an entity has a certain property is *consistent* with the knowledge base instead of checking whether the entity actually has this property. But this has to be done carefully to avoid overgeneration.

(9) a. I saw *a rabbit* in my vegetable garden, this morning. **The fat animal** was eating the carrots.

b. Yesterday, I saw *a rabbit with long white ears* in my vegetable garden. Today, I saw **another rabbit with short black ears** in my flowerbed.

# 7 Conclusion

Presupposition triggers impose constraints on the discourse context they are used in. A natural language generation system has to verify that these constraints are satisfied before licensing the use of such an expression. We have

argued that model checking supports the necessary inferences and allows a natural formulation of the corresponding reasoning tasks.

Since the presupposed material is often more general than what has been said (explicitly) about the anchor, some background knowledge about subsumption relations between concepts has to be taken into account. To capture this interaction with world knowledge, we proposed to embed queries to a description logic theorem prover into a model checking algorithm.

The model checking algorithm presented in Section 3 is implemented with an interface to the description logic prover RACER (Haarslev and Möller 1999) and will be integrated into the InDiGen NLG system (Striegnitz 2001). This system, which interleaves the generation process with contextual reasoning, as described in Section 5, provides a framework for experimenting with different inference tools.

As a next step in this line of research, we will investigate how indirect anchoring and accommodation can be encompassed with the approach presented in this paper. To do so, we will also have to enhance the discourse model by a modeling of salience of entities (cf. Kohlhase and Koller 2000, Krahmer and Theune 1998).

Furthermore, we are interested in further investigating the use of model checking for natural language processing and examining which other linguistic constructions involve contextual requirements that can be captured as model checking tasks.

**Acknowledgements** I thank Claire Gardent for supporting me and for all her help in making the work presented in this paper happen. I also thank Bonnie Webber for reading and commenting on an earlier, *very* drafty draft of this paper and Carlos Areces for his extensive comments on a later draft. Furthermore, I would like to thank four anonymous reviewers of ICoS-3 who made detailed comments and asked many very good questions. This work was supported by the project InDiGen in the priority program Language Production of the DFG (Deutsche Forschungsgemeinschaft).

## References

Appelt, D. E. (1985). *Planning English Sentences.* Cambridge University Press.

Baader, F., D. McGuinness, D. Nardi, and P. F. Patel-Schneider (Eds.) (2001). *The Description Logics Handbook.* Cambridge University Press. To appear.

Baumgartner, P. and M. Kühn (1999). Abductive Coreference by Model Construction. In C. Monz and M. de Rijke (Eds.), *Proceedings of ICOS-1*, pp. 21–37.

Beaver, D. I. (1997). Presupposition. In J. van Benthem and A. ter Meulen (Eds.), *Handbook of Logic and Language*, pp. 939–1008. Elsevier Science B.V.

Clark, H. H. (1975). Bridging. In Schank and Nash-Webber (Eds.), *Theoretical Issues in Natural Language Processing.* MIT Press. Reprinted in: Johnson-Laird and Wason (eds.) *Thinking. Readings in Cognitive Science.* Cambridge University Press. 1977.

Dale, R. and E. Reiter (1995). Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science 18*, 233–263.

Danlos, L. (1987). *The Linguistic Basis of Text Generation.* Studies in Natural Language Processing. Cambridge: Cambridge University Press.

Donini, F. M., M. Lenzerini, D. Nardi, and A. Schaerf (1996). Reasoning in Description Logics. In G. Brewka (Ed.), *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pp. 193–238. CSLI Publications.

Gardent, C. and K. Konrad (2000). Interpreting Definites Using Model Generation. *Journal of Language and Computation 1*(2), 193–208.

Gardent, C. and K. Striegnitz (2001). Generating Indirect Anaphora. In H. Bunt, I. van Sluis, and E. Thijsse (Eds.), *Proceedings of the 4th International Workshop on Computational Semantics*, pp. 138–155. Tilburg University.

Geurts, B. (1999). *Presuppositions and Pronouns*. Oxford: Elsevier.

Gundel, K., N. Hedberg, and R. Zacharski (1993). Cognitive Status and the Form of Referring Expressions in Discourse. *Language 69*(2), 274–307.

Haarslev, V. and R. Möller (1999). RACE System Description. In P. Lambrix, A. Borgida, R. Möller, M. Lenzerini, and P. Patel-Schneider (Eds.), *Proceedings of 1999 International Workshop on Description Logics*, pp. 130–132. RACE is now called RACER and available at: `http://kogs-www.informatik.uni-hamburg.de/~race/`.

Heim, I. (1983). On the Projection Problem for Presuppositions. In M. Barlow, D. Flickinger, and M. Wescoat (Eds.), *Proceedings of WCCFL 2*, pp. 114–125. Stanford University.

Hobbs, J. (1985). Ontological Promiscuity. In *Proceedings of ACL*, pp. 61–69.

Horrocks, I. (1999). FaCT and iFaCT. In P. Lambrix, A. Borgida, R. Möller, M. Lenzerini, and P. Patel-Schneider (Eds.), *Proceedings of 1999 International Workshop on Description Logics*, pp. 133–135. FaCT is available at: `http://www.cs.man.ac.uk/~horrocks/FaCT/`.

Hovy, E. (1988). *Generating Natural Language Under Pragmatic Constraints*. Hillsdale, NJ: Lawrence Erlenbaum Associates.

Joshi, A. and Y. Schabes (1997). Tree-Adjoining Grammars. In G. Rozenberg and A. Salomaa (Eds.), *Handbook of Formal Languages*, Chapter 2, pp. 69–123. Berlin: Springer-Verlag.

Karttunen, L. (1974). Presuppositions and Linguistic Context. *Theoretical Linguistics 1*, 181–194.

Kohlhase, M. and A. Koller (2000). Towards A Tableaux Machine for Language Understanding. In *Proceedings of ICOS-2*, Dagstuhl.

Krahmer, E. and M. Theune (1998). Context Sensitive Generation of Descriptions. In *Proceedings International Conference on Spoken Language Processing (ICSLP'98)*, Volume 4, Sydney, pp. 1151–1154.

Levinson, S. C. (1983). *Pragmatics*. Cambridge Textbooks in Linguistics. Cambridge: Cambridge University Press.

Lewis, D. (1979). Score-keeping in a Language Game. In R. Bauerle, U. Egli, and A. von Stechow (Eds.), *Semantics from different points of view*, pp. 172–187. Berlin: Springer Verlag.

Ramsay, A. and H. Seville (2000). Models and Discourse Models. *Journal of Language and Computation 1*(2), 159–174.

Stalnaker, R. (1974). Pragmatic Presuppositions. In M. Kunitz and D. Unger (Eds.), *Semantics and Philosophy*, pp. 197–213. New York University Press.

Stone, M. (2000). Towards a Computational Account of Knowledge, Action and Inference in Instructions. *Journal of Language and Computation 1*(2), 231–246.

Stone, M. and C. Doran (1997). Sentence Planning as Description Using Tree Adjoining Grammar. In *Proceedings of ACL*, pp. 198–205.

Stone, M., C. Doran, B. Webber, T. Bleam, and M. Palmer (2001). Microplanning with Communicative Intentions: The SPUD System. Avvailable at `http://arXiv.org/abs/cs/0104022`. Submitted to Computational Intelligence.

Striegnitz, K. (2001.). Pragmatic Constraints and Contextual Reasoning in a Natural Language Generation System: a system description. Draft. Available at `http://www.coli.uni-sb.de/cl/projects/indigen/publications.html`.

van der Sandt, R. (1992). Presupposition Projection as Anaphora Resolution. *Journal of Semantics 9*, 333–377.

Vardi, M. Y. (1986). Querying Logical Databases. In *Journal of Computer and System Sciences*, Volume 33, pp. 142–160.

Zeevat, H. (2000). Explaining Presupposition Triggers. Draft. Available at `http://www.hum.uva.nl/computerlinguistiek/henk/henkPUB.html`. (to appear).