Presupposition Projection and Accommodation in Mathematical Texts

Marcos Cramer **Daniel Kühlwein**
University of Bonn
{cramer,kuehlwei}@math.uni-bonn.de

Bernhard Schröder
University of Duisburg-Essen
bernhard.schroeder@uni-due.de

Abstract

This paper discusses presuppositions in mathematical texts and describes how presupposition handling was implemented in the Naproche system for checking natural language mathematical proofs. Mathematical texts have special properties from a pragmatic point of view, since in a mathematical proof every new assertion is expected to logically follow from previously known material, whereas in most other texts one expects new assertions to add logically new information to the context. This pragmatic difference has its influence on how presuppositions can be projected and accommodated in mathematical texts. Nevertheless, the account of presupposition handling developed for the Naproche system turned out to have equivalent projection predictions to an existing account of presupposition projection.

1 Introduction

The special language that is used in mathematical journals and textbooks has some unique linguistic features, on the syntactic, on the semantic and on the pragmatic level: For example, on the syntactic level, it can incorporate complex symbolic material into natural language sentences. On the semantic level, it refers to rigorously defined abstract objects, and is in general less open to ambiguity than most other text types. On the pragmatic level, it reverses the expectation on assertions, which have to be implied by the context rather than adding new information to it.

We call this special language *the semi-formal language of mathematics* (SFLM), and call texts written in this language *mathematical texts*. The content of mathematical texts can be divided into object level content (mathematical statements and their proofs) and meta level content (e.g. historical remarks and motivation behind certain definitions and one's interest in certain theorems). For the rest of the paper, we will focus on object level content of mathematical texts.

Our work on presuppositions in mathematical texts has been conducted in the context of the *Naproche project*. The Naproche project studies SFLM from the perspectives of linguistics, logic and mathematics. A central goal of Naproche is to develop a controlled natural language (CNL) for mathematical texts and implement a system, the *Naproche system*, which can check texts written in this CNL for logical correctness using methods from computational linguistics and automatic theorem proving.

The Naproche system first translates a CNL text into a *Proof Representation Structure* (*PRS*). PRSs are Discourse Representation Structures (DRSs, (Kamp and Reyle, 1993)), which are enriched in such a way as to represent the distinguishing characteristics of SFLM (Cramer et al., 2010). For every statement made in the text, the Naproche System extracts from the PRS a proof obligation that gets sent to an automated theorem prover, in order to check that the claim made by the statement follows from the available information (Cramer et al., 2009).

In this paper we describe how the checking algorithm of Naproche had to be altered when expressions triggering presuppositions were added to the Naproche CNL. Presuppositions also have to be checked for correctness, but behave differently from assertions in mathematical texts because of their special pragmatic properties to be discussed below.

2 Presuppositions

Losely speaking, a *presupposition* of some utterance is an implicit assumption that is taken for granted when making the utterance. In the literature, presuppositions are generally accepted to be triggered by certain lexical items called *presupposition triggers*. Among them are definite noun phrases (in English marked by the definite article "the", possessive pronouns or genitives), factive verbs (like "regret", "realize" and "know"), change of state verbs ("stop" and "begin"), iteratives ("again") and some others.

Presupposition projection is the way in which presuppositions triggered by expressions within the scope of some operator have to be evaluated outside this scope. The precise way in which presuppositions project under various operators has been disputed at great length in the literature (see for example Levinson (1983) and Kadmon (2001) for overviews of this dispute). The DRS-based presupposition projection mechanism that we came up with for dealing with presuppositions in mathematical texts turned out to be equivalent to Heim's (1983) approach to presupposition projection.

Presupposition accommodation is what we do if we find ourselves faced with a presupposition the truth of which we cannot establish in the given context: We add the presupposition to the context, in order to be able to process the sentence that presupposes it.

In mathematical texts, most of the presupposition triggers discussed in the linguistic literature, e.g. factive verbs, change of state verbs and iteratives, are not very common or even completely absent. Definite descriptions, however, do appear in mathematical texts (e.g. "the smallest natural number n such that $n^2 - 1$ is prime"). And there is another kind of presupposition trigger, which does not exist outside mathematical texts: Function symbols. For example, the devision symbol "$/$" presupposes that its second (right hand) argument is non-zero; and in a context where one is working only with real and not with complex numbers, the sqare root symbol "$\sqrt{}$" presupposes that its argument is non-negative. As has been pointed out by Kadmon (2001), the kind of presupposition trigger does, however, not have any significant influence on the projection and accommodation properties of presuppositions. For this reason, we will concentrate on examples with definite descriptions.

Although terminology is not used in a fully uniform fashion among linguists, we will make the following distinctions suitable for our purposes. We analyse noun phrases semantically into a determiner (here: "the") and a restricting property. Definite noun phrases referring to a single object by a restricting property whose extension contains exactly one object we call *definite descriptions*. Definite noun phrases in the singular with restricting properties whose extension contains more than one object get their referential uniqueness usually by anaphoric reference to an object mentioned previously; they are called *anaphoric definite noun phrases*. A mathematical example of an anaphoric definite noun phrase is "the group" used to refer to a group mentioned recently in the text. The example above ("the smallest natural number n such that $n^2 - 1$ is prime") was an example of a definite description.

The presupposition of a singular definite description with the restricting property F is that there is a unique object with property F. This presupposition can be divided into two separate presuppositions: One existential presupposition, claiming that there is at least one F, and one uniqueness presupposition, claiming that there is at most one F.

3 Proof Representation Structures

For the purpose of this paper, we provide a simplified definition of Proof Representation Structures, which is very similar to standard definitions of Discourse Representation Structures. A full-fledged definition of Proof Representation Structures can be found in Cramer et al. (2010).

A Proof Representation Structure is a pair consisting of a list of discourse referents and an ordered list of conditions. Just as in the case of DRSs, PRSs and PRS conditions are defined recursively: Let A, B be PRSs and $d, d_1, \ldots, d_n$ discourse referents. Then

- for any n-ary predicate p (e.g. expressed by adjectives and noun phrases in predicative use and verbs in SFLM), $p(d_1, \ldots, d_n)$ is a condition.
- $\neg A$ is a condition, representing a negation.
- $B \Rightarrow A$ is a condition, representing an assumption (B) and the set of claims made inside the scope of this assumption (A).
- A is a condition.
- *the*(d, A) is a condition, representing a definite description with restricting property F, where

$$\begin{array}{|l|}\hline d_1,\ldots,d_m \\ \hline c_1 \\ \vdots \\ c_n \\ \hline \end{array}$$

Figure 1: A PRS with discourse referents $d_1, ..., d_m$, and conditions $c_1, ..., c_n$.

> d is the discourse referent introduced by this definite description and A is the representation of F.

Apart from the *the*-condition, which was also absent from PRSs as they were defined in Cramer et al. (2010), there are two differences between this definition of PRSs and standard definitions of DRSs: Firstly, the list of PRS conditions is ordered, whereas DRS conditions are normally thought to form an unordered set. Secondly, a bare PRS can be a direct condition of a PRS. Both of these differences are due to the fact that a PRS does not only represent which information is known and which discourse referents are accessible after processing some (possibly partial) discourse, but also represents in which order the pieces of information and discourse referents were added to the discourse context.

Similar to DRSs, we can display PRSs as "boxes" (Figure 1). If $m = 0$, we leave out the top cell.

4 Checking PRSs without presuppositions

In order to explain our treatment of presuppositions, we first need to explain how PRSs without presuppositions are checked for correctness by the Naproche system.

The checking algorithm makes use of *Automated Theorem Provers* (ATPs) for first-order logic (Fitting, 1996).[1] Given a set of *axioms* and a *conjecture*, an ATP tries to find either a proof that the axioms logically imply the conjecture, or build a model for the premisses and the negation of the conjecture, which shows that that they don't imply it. With difficult problems, an ATP might not find any proof or counter-model within the time limit that one fixed in advance. A conjecture together with a set of axioms handed to an ATP is called a *proof obligation*.

[1]The checking algorithm is implemented in such a way that it is easily possible to change the ATP used. Most of our tests of the system are performed with the prover E (Schulz, 2002).

The checking algorithm keeps a list of first-order formulae considered to be true, called *premises*, which gets continuously updated during the checking process. The conditions of a PRS are checked sequentially. Each condition is checked under the currently active premises. According to the kind of condition, the Naproche system creates obligations which have to be discharged by an ATP.

Below we list how the algorithm proceeds depending on the PRS condition encountered. We use Γ to denote the list of premises considered true before encountering the condition in question, and Γ' to denote the list of premises considered true after encountering the condition in question. A proof obligation checking that ϕ follows from Γ will be denoted by $\Gamma \vdash \phi$. For any given PRS A, we denote by $FI(A)$ the *formula image* of A, which is a list of first-order formulae representing the information introduced in A; the definitions of $FI(A)$ and of the checking algorithm are mutually recursive, as specified below.

We first present the algorithm for PRS conditions that do not introduce new discourse referents. Next we extend it to conditions introducing discourse referents. In section 5, we extend it further to conditions that trigger presuppositions. (For simplifying the presentation, we treat formula lists as formula sets, i.e. allow ourselves to use set notation when in reality the algorithm works with ordered lists.)

(1) For a condition of the form $p(d_1, \ldots, d_n)$, check $\Gamma \vdash p(d_1, \ldots, d_n)$ and set $\Gamma' := \Gamma \cup \{p(d_1, \ldots, d_n)\}$.

(2) For a condition of the form $\neg A$, check $\Gamma \vdash \neg \bigwedge FI(A)$ and set $\Gamma' := \Gamma \cup \{\neg \bigwedge FI(A)\}$.

(3) For a condition of the form $B \Rightarrow A$, where no discourse referents are introduced in A, check A with initial premise set $\Gamma \cup FI(B)$, and set $\Gamma' := \Gamma \cup \Delta$, where $\Delta := \{\forall \vec{x}(\bigwedge FI(B) \rightarrow \phi) | \phi \in FI(A)\}$ and $\vec{x}$ is the list of free variables in $FI(B)$.

(4) For a condition of the form A, where no discourse referents are introduced in A, check A with initial premise set Γ, and set $\Gamma' := \Gamma \cup FI(A)$.

For computing $FI(A)$, the algorithm proceeds analoguously to the checking of A, only that no proof obligations are sent to the ATP: The updated premise lists are still computed, and $FI(A)$ is defined to be $\Gamma' - \Gamma$, where Γ is the premise list before processing the first condition in A and Γ' is the premise list after processing the last condition in A. This is implemented by allowing the algorithm to process a PRS A in two different modes: The Check-Mode described above for checking the content of A, and the No-Check-Mode, which refrains from sending proof obligations to the ATP, but still expands the premise list in order to compute $FI(A)$.

For the cases (1) and (2), it is easy to see that what gets added to the list of premises in the Check-Mode is precisely what has been checked to be correct by the ATP. In the cases (3) and (4), it can also be shown that what gets added to the set of premises has implicitly been established to be correct by the ATP.

Special care is required when in conditions of the form $B \Rightarrow A$ or A, new discourse referents are introduced in A. Let us first consider the simpler case of a condition of the form A that introduces new discourse referents; this corresponds to sentences like "There is an integer x such that $2x - 1$ is prime.", i.e. sentences with an existential claim, which make the existentially introduced entity anaphorically referencible by the later discourse. As would be expected, we need to check $\Gamma \vdash \exists x(\mathit{integer}(x) \wedge \mathit{prime}(2x-1))$ in this case. But we cannot just add $\exists x(\mathit{integer}(x) \wedge \mathit{prime}(2x-1))$ to the premise set, since the first order quantifier $\exists$ does not have the dynamic properties of the natural language quantification with "there is": When we later say "$x \neq 1$", the proof obligation of the form $\Gamma \cup \{\exists x(\mathit{integer}(x) \wedge \mathit{prime}(2x-1))\} \vdash x \neq 1$ for this sentence would not make sense, since the free x in the conjecture would not corefer with the existentially bound x in the axiom.

We solve this problem by *Skolemizing* (Brachman and Levesque, 2004) existential formulae before adding them to the premise list: In our example, we add $\mathit{integer}(c) \wedge \mathit{prime}(2c-1)$ (for some new constant symbol c) to the premise list. Later uses of x will then also have to be substituted by c, so the proof obligation for "$x \neq 1$" becomes $\Gamma \cup \{\mathit{integer}(c) \wedge \mathit{prime}(2c-1)\} \vdash c \neq 1$. Given that the discourse referents introduced in A become free variables in $FI(A)$, we can require more generally: For a condition of the form A which introduces discourse referents, check $\Gamma \vdash \exists \vec{x}(\bigwedge FI(A))$ (where $\vec{x}$ is the list of free variables in $FI(A)$), and set $\Gamma' := \Gamma \cup S(FI(A))$. We define $S(FI(A))$ (the Skolemized version of of $FI(A)$) to be the set of formulae that we get when we substitute each free variable used in some formula in $FI(A)$ by a different new constant symbol, ensuring that the same constant symbol is used for the same free variable across different formulae in $FI(A)$.

In the case of conditions of the form $B \Rightarrow A$ with A introducing new discourse referents, we need the more general kind of Skolemization, which involves introducing new function symbols rather than new constant symbols: We proceed in the same way as for the case when A doesn't introduce new discourse referents, only that in the definition of Γ' we replace Δ by its Skolemized form $S(\Delta)$. $S(\Delta)$ consists of Skolemized versions of the formulae in Δ, where the Skolem functions are chosen in such a way that any free variable appearing in more than one formula in Δ gets replaced by the same function across the different formulae in which it appears.

5 Checking PRSs with presuppositions

Most accounts of presupposition make reference to the *context* in which an utterance is uttered, and claim that presuppositions have to be satisfied in the context in which they are made. There are different formalisations of how a context should be conceptualised. For enabling the Naproche checking algorithm described in the previous section to handle presuppositions, it is an obvious approach to use the list of active premises (which include definitions) as the context in which our presuppositions have to be satisfied.

As noted before, assertions in mathematical texts are expected to be logically implied by the available knowledge rather than adding something logically new to it. Because of this pragmatic peculiarity, both presuppositions and assertions in proof texts have to follow logically from the context. For a sentence like "The largest element of M is finite" to be legitimately used in a mathematical text, both the unique existence of a largest element of M and its finiteness must be inferable from the context.[2]

This parallel treatment of presuppositions and assertions, however, does not necessarily hold for presupposition triggers that are subordinated by a logical operation like negation or implication. For example, in the sentence "A does not contain the empty set", the existence and uniqueness presuppo-

[2]The remaining distinctive feature between assertions and presuppositions is that the failure of the latter ones makes the containing sentences meaningless, not only false.

sitions do not get negated, whereas the containment assertion does. This is explained in the following way: In order to make sense of the negated sentence, we first need to *make sense of* what is inside the scope of the negation. In order to make sense of some expression, all presuppositions of that expression have to follow from the current context. The presuppositions triggered by "the empty set" are inside the scope of the negation, so they have to follow from the current context. The containment assertion, however, does not have to follow from the current context, since it is not a presupposition, and since it is negated rather than being directly asserted.

In our implementation, *making sense of something* corresponds to processing its PRS, whether in the Check-Mode or in the No-Check-Mode. So according to the above explanation, presuppositions, unlike assertions, also have to be checked when encountered in the No-Check-Mode.

For example, the PRS of sentence (5) is (6).

(5) A does not contain the empty set.

$$(6)\quad \boxed{\neg\ \boxed{\begin{array}{l} \\ \mathit{the}(x, \boxed{\begin{array}{l}\mathit{empty}(x)\\ \mathit{set}(x)\end{array}}\,) \\ \mathit{contain}(A,x)\end{array}}}$$

When the checking algorithm encounters the negated PRS, it needs to find the formula image of the PRS, for which it will process this PRS in No-Check-Mode. Now the *the*-condition triggers two presuppositions, which have to be checked despite being in No-Check-Mode. So we send the proof obligations (7) and (8) (for a new constant symbol c) to the ATP. Finally, the proof obligation that we want for the assertion of the sentence is (9).

(7) $\Gamma \vdash \exists x(\mathit{empty}(x) \wedge \mathit{set}(x))$

(8) $\Gamma \cup \{\mathit{empty}(c) \wedge \mathit{set}(c)\} \vdash$
$\forall y(\mathit{empty}(y) \wedge \mathit{set}(y) \rightarrow y = c)$

(9) $\Gamma \cup \{\mathit{empty}(c) \wedge \mathit{set}(c), \forall y(\mathit{empty}(y) \wedge \mathit{set}(y) \rightarrow y = c)\} \vdash \neg \mathit{contain}(A, c)$

In order to get this, we need to use the non-presuppositional formula image $\{\mathit{contain}(A, c)\}$ of the negated PRS: The non-presuppositional formula image is defined to be the subset of formulae of the formula image that do not originate from presuppositions. When implementing the checking algorithm for PRSs with presuppositions, we have to use this non-presuppositional formula image wherever we used the formula image in the original checking algorithm. The presupposition premises which get pulled out of the formula image have to be added to the list of premises that were active before starting to calculate the formula image.

This pulling out of presupposition premises is not always as simple as in the above example. Consider for example sentence (10), whose PRS is (11).

(10) There is a finite non-empty set M of natural numbers such that the largest element of M is even.[3]

$$(11)\quad \begin{array}{|l|}\hline M \\ \hline \mathit{finite}(M) \\ \mathit{non\text{-}empty}(M) \\ \mathit{set\_of\_nats}(M) \\ \mathit{the}(x, \boxed{\mathit{largest\_elt}(x, M)}\,) \\ \mathit{even}(x) \\ \hline \end{array}$$

The Skolemized premise from the existential presupposition is $\mathit{largest\_elt}(c, M)$, which contains a free occurence of the variable M, but should be pulled out of the PRS introducing M, i.e. out of the scope of M, in order to be added to the set Γ of premises available before encountering this sentence. Pulling this occurence of M out of the scope of M would make the premise meaningless, so we need a more sophisticated approach to pulling out presupposition premises:

According to the above account, we will check the existential presupposition in question using the proof obligation (12). Given that M does not appear in Γ (as it is a newly introduced discourse referent), this is logically equivalent to having checked (13), whose Skolemized form (14) will be added to Γ (where sk_x is the new function symbol introduced

[3] The definite noun phrase "The largest element of M" can be read like a function depending on M. When, like in our example, such functional definite descriptions are used as functions on a variable that we are quantifying over, the presuppositions of the functional definite description can restrict the domain of the quantifier to entities for which the presupposition is satisfied. Such a restriction of a quantifier is an instance of accommodation (local accomodation in our account), which will be treated in section 7. In this section we are interested in presupposition handling without accommodation, i.e. without restricting the domain of the quantifier in this example. So the presuppositions of "the largest element of M" have to be fulfilled for any finite non-empty set M of natural numbers.

for x when Skolemizing). This extended premise set is used to check the existential claim of the sentence in (15).

(12) $\Gamma \cup \{\mathit{finite}(M), \mathit{non\text{-}empty}(M), \mathit{set\_of\_nats}(M)\} \vdash \exists x\ \mathit{largest\_elt}(x, M)$

(13) $\Gamma \vdash \forall M\ (\mathit{finite}(M) \wedge \mathit{non\text{-}empty}(M) \wedge \mathit{set\_of\_nats}(M) \rightarrow \exists x\ \mathit{largest\_elt}(x, M))$

(14) $\forall M\ (\mathit{finite}(M) \wedge \mathit{non\text{-}empty}(M) \wedge \mathit{set\_of\_nats}(M) \rightarrow \mathit{largest\_elt}(sk_x(M), M))$

(15) $\Gamma \cup \{(14)\} \vdash \exists M\ (\mathit{finite}(M) \wedge \mathit{non\text{-}empty}(M) \wedge \mathit{set\_of\_nats}(M) \wedge \mathit{even}(sk_x(M)))$

6 Comparison to Heim's presupposition projection

Heim (1983) is concerned with the projection problem, i.e. with "predicting the presuppositions of complex sentences in a compositional fashion from the presuppositions of their parts". For us, the projection problem only had indirect importance: The reason for our occupation with presupposition was to be able to check mathematical texts containing presupposition triggers for correctness. This does involve checking that the presuppositions of every trigger are satisfied in the local context of the trigger, but it doesn't necessarily involve explicitly computing presuppositions for complex sentences.

Given the sentence (16), Heim's theory predicts that the existential presupposition of the definite description gives rise to the presupposition (17) for the complex sentence.

(16) If x is positive, then the multiplicative inverse of x is positive.

(17) If x is positive, then x has a multiplicative inverse.

(18) $$\boxed{\begin{array}{c}\begin{array}{|c|}\hline x \\ \hline \mathit{pos}(x) \\ \hline\end{array} \Rightarrow \begin{array}{|l|}\hline \\ \hline \mathit{the}(y, \boxed{\mathit{mult\_inv}(y, x)}) \\ \mathit{pos}(y) \\ \hline\end{array}\end{array}}$$

In our treatment of presupposition, computing the presupposition (17) explicitly is not the central issue; what we do instead is to justify the presupposition with the information locally available when encountering the presupposition. In the example sentence, whose PRS is (18), the formula image of the left PRS of the implication condition ($\{\mathit{pos}(x)\}$) is Skolemized and added to the set Γ of premises available before encountering this sentence, so that the set of premises available when encountering the *the*-condition is $\Gamma \cup \{\mathit{pos}(c)\}$. Hence the proof obligation for justifying the existential presupposition of the definite description is (19). Having checked this is equivalent to having checked (20), i.e. having deduced Heim's projected presupposition from Γ.

(19) $\Gamma \cup \{\mathit{pos}(c)\} \vdash \exists y\ \mathit{mult\_inv}(y, c)$

(20) $\Gamma \vdash \forall x\ (\mathit{pos}(x) \rightarrow \exists y\ \mathit{mult\_inv}(y, x))$

Also for presuppositions subordinated under other kinds of logical operators, our theory is in this way equivalent to Heim's theory. This is the sense in which one can say that our theory is equivalent to Heim's theory.[4] On the other hand, we arrive at these results in a somewhat different way to Heim: Heim defines contexts in a semantical way as sets of possible worlds or, in her account of functional definite descriptions, as a set of pairs of the form $\langle g, w\rangle$, where g is a sequence of individuals and w is a possible world, whereas we define a context syntactically as a list of first-order formulae.

7 Accommodation

One commonly distinguishes between *global* and *local* accommodation. Global accommodation is the process of altering the global context in such a way that the presupposition in question can be justified; local accommodation on the other hand involves only altering some local context, leaving the global context untouched. It is a generally accepted pragmatic principle that *ceteris paribus* global accommodation is preferred over local accommodation.

In the introduction, we mentioned the pragmatic principle in mathematical texts that new assertions do not add new information (in the sense of logically not infereable information) to the context. Here "context" of course doesn't refer to our formal definition of context as a list of formulae. In fact, if

[4]Here we are comparing our theory without accommodation to Heim's theory without accommodation. Heim calls the projected universal presupposition for functional noun phrases (as discussed in the previous section, cf. example (10) with presupposition (13)) "unintuitively strong", and gives an explanation for this using accommodation. But this universal presupposition is what her account without accommodation predicts, just as in our case. Cf. the justification for the strong universal presupposition in footnote 3.

we take a possible world to be a situation in which certain axioms, definitions, and assumptions hold or do not hold, we can make sense of the use of “context” in this assertion by applying Heim’s definition of a context as a set of possible worlds. When mathematicians state axioms, they limit the context, i.e. the set of possible worlds they consider, to the set where the axioms hold. Similarly, when they make local assumptions, they temporarily limit the context. But when making assertions, these assertions are thought be logically implied by what has been assumed and proved so far, so they do not further limit the context.

This pragmatic principle of not adding anything to the context implies that global accommodation is not possible in mathematical texts, since global accommodation implies adding something new to the global context. Local accommodation, on the other hand, is allowed, and does occur in real mathematical texts:

> Suppose that f has n derivatives at x_0 and n is the smallest positive integer such that $f^{(n)}(x_0) \neq 0$.
>
> (Trench, 2003)

This is a local assumption. The projected existential presupposition of the definite description “the smallest positive integer such that $f^{(n)}(x_0) \neq 0$” is that for any function f with some derivatives at some point x_0, there is a smallest positive integer n such that $f^{(n)}(x_0) \neq 0$. Now this is not valid in real analysis, and we cannot just assume that it holds using global accommodation. Instead, we make use of local accommodation, thus adding the accommodated fact that there is a smallest such integer for f to the assumptions that we make about f with this sentence.

The fact that one has to accommodate locally rather than globally does not, however, always fix which context we alter when accommodating. Consider for example sentence (21), used in a context where we have already defined a set A_x of real numbers for every real number x.

(21) For all $x \in \mathbb{R}$, if A_x doesn’t contain $\frac{1}{x}$, then A_x is finite.

The question is whether we need to check the finiteness of A_0 in order to establish the truth of (21), or whether the the finiteness of A_0 is irrelevant. Since the use of $\frac{1}{x}$ presupposes that $x \neq 0$, which doesn’t hold for any arbitrary $x \in \mathbb{R}$, we have to locally accommodate that $x \neq 0$. But we can either accommodate this within the scope of the negation or outside the scope of the negation, but still locally within the conditional. In the first case, we have to establish that A_0 is finite, whereas in the second case we don’t.

Unlike the presupposition handling described in the previous sections, local accommodation has not yet been implemented into Naproche. Before this can be done, we need some mechanism for deciding which of a number of possible local accommodations is preferred in cases like the above.

8 Related Work

Presuppositions in mathematical texts have already been studied before: Zinn (2000) discusses presuppositions and implicatures in mathematical texts. His work on presuppositions focuses on the presuppositions that are justified using information from proof plans. Since Naproche currently doesn’t use proof plans, this kind of presupposition is not yet implemented in the Naproche CNL.

Ganesalingam (2009) describes an innovative way of computing the presuppositions triggered by mathematical function symbols (like $-^{-1}$) and the presuppositions given rise to by selective restrictions (e.g. the presupposition “x is a natural number” of the utterance “x is prime”) from the definitions where the corresponding function symbols or expressions were defined. Once Naproche implements other presupposition triggers than just definite descriptions, an algorithm similar to that presented by Ganesalingam will be implemented for computing presuppositions triggered by symbols or expressions defined in the text.

9 Conclusion

In this paper we discussed the handling of presuppositions in the checking algorithm of the Naproche system, and compared its projection predictions to those of Heim (1983). We noted that our projection predictions are equivalent to those of Heim, despite the fact that we arrive at these predictions in a different way.

Additionally, we considered accommodation in mathematical texts, and noted that global accommodation is blocked by a pragmatic peculiarity of mathematical texts. Future work will involve implementing local accommodation into the Naproche system.

References

Brachman, Ronald J., and Hector J. Levesque. 2004. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers, Massachusetts, US.

Cramer, Marcos, Peter Koepke, Daniel Kühlwein, and Bernhard Schröder. 2009. *The Naproche System. Calculemus 2009 Emerging Trend Paper*.

Cramer, Marcos, Bernhard Fisseni, Peter Koepke, Daniel Kühlwein, Bernhard Schröder, and Jip Veldman. 2010 (in press). *The Naproche Project – Controlled Natural Language Proof Checking of Mathematical Texts*. *CNL 2009 Workshop, LNAI 5972 proceedings*. Springer.

Fitting, Melvin. 1996. *First-order logic and automated theorem proving*. Springer. Springer.

Ganesalingam, Mohan. 2009. *The Language of Mathematics*. Doctoral thesis draft. `http://people.pwf.cam.ac.uk/mg262/GanesalingamMdis.pdf`.

Heim, Irene. 1983. *On the projection problem for presuppositions*. D. Flickinger et al. (eds.). *Proceedings of the Second West Coast Conference on Formal Linguistics, 114-125*.

Kamp, Hans, and Uwe Reyle. 1993. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language*. Kluwer Academic Publisher.

Kadmon, Nirit. 2001. *Formal Pragmatics*. Wiley-Blackwell, Oxford, UK.

Levinson, Stephen C. 1983. *Pragmatics*. Cambridge University Press, Cambridge, UK.

Schulz, Stephan. 2002. E – A Brainiac Theorem Prover. *Journal of AI Communications*, 15(2):111–126.

Trench, William F. 2003. *Introduction to Real Analysis*. Pearson Education.

Zinn, Claus. 2000. *Computing Presuppositions and Implicatures in Mathematical Discourse*. J. Bos and M. Kohlhase (eds.). *Proceedings of the Second Workshop on Inference in Computational Semantics, 121-135*.