Interpreting Plurals in the Naproche CNL

Marcos Cramer, Bernhard Schröder

University of Bonn and University of Duisburg-Essen
`cramer@math.uni-bonn.de`
`bernhard.schroeder@uni-due.de`
`http://www.naproche.net`

13th September 2010

Outline

1 Introduction

2 Proof Representation Structures

3 Collective vs. distributive readings of plurals

4 Scope ambiguity

5 Pairwise interpretations of collective plurals

6 The plural interpretation algorithm

7 Related and future work

8 Conclusion

The Naproche Project

- The **Naproche** project (**Na**tural language **Pro**of **Che**cking) is a joint project of the University of Bonn and the University of Duisburg-Essen.
- We study the **semi-formal language of mathematics** (SFLM) as used in journals and textbooks from the perspectives of linguistics 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 CNL and the Naproche system

- The **Naproche CNL** is a controlled natural language for mathematical texts, i.e. a controlled subset of SFLM.
- The **Naproche system** translates Naproche CNL texts first into **Proof Representation Structures** (**PRS**s), an adapted version of Discourse Representation Structures.
- PRSs are further translated into lists of first-order formulae which are used for checking the logical correctness of a Naproche text using automated theorem provers.

Applications of Naproche

There are two main applications we have in mind for Naproche:

- to make formal mathematics more readable to the average mathematician
- to use it as a tool that supports undergraduate students in writing formally correct proofs and thus get used to (a subset of) SFML

Plurals in Naproche

- A recent addition to the Naproche CNL are plural statements.
- By this we mean not only statements involving nouns in the plural (e.g. "numbers") and verbs conjugated in plural forms (e.g. "are"), but also conjunctive coordinations of noun phrases (e.g. "$x + y$ and $x \cdot y$ are even").
- We discuss the collective-distributive ambiguity as well as a special scope ambiguity conjunctions give rise to, and explain how these ambiguities are resolved in Naproche.
- Plural definite noun phrases (e.g. "the real numbers") are not yet implemented in Naproche and are left out of the discussion in this talk.

Outline

1 Introduction

2 **Proof Representation Structures**

3 Collective vs. distributive readings of plurals

4 Scope ambiguity

5 Pairwise interpretations of collective plurals

6 The plural interpretation algorithm

7 Related and future work

8 Conclusion

Proof Representation Structures

- PRSs are Discourse Representation Structures, which are enriched in such a way as to represent the distinguishing characteristics of the mathematical language.
- For the purpose of this talk, we present a simplified definition of PRSs.
- A PRS is a pair consisting of a list of discourse referents and an ordered list of conditions.

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

PRS conditions

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.
- A mathematical formula 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).
- $static(A)$ is a condition.

Accessibility and static PRS conditions

- Accessibility in PRSs is defined analogously to accessibility in DRSs:
  - Discourse referent introduced in conditions of the form $\neg A$ or $B \Rightarrow A$ are not accessibly outside these conditions.
- We have introduced an additional condition of the form $\mathit{static}(A)$:
  - Discourse referent introduced in a condition of the form $\mathit{static}(A)$ are also not accessibly outside this condition.
  - These static PRS conditions allow us to represent existential claims with a static rather than a dynamic existential quantification.

Outline

1 Introduction

2 Proof Representation Structures

3 Collective vs. distributive readings of plurals

4 Scope ambiguity

5 Pairwise interpretations of collective plurals

6 The plural interpretation algorithm

7 Related and future work

8 Conclusion

Collective vs. distributive readings of plurals

The following sentence is ambiguous:

Example

Three men lifted a piano.

- It can mean either that three men lifted a piano together (in a single lifting act), or that there were three lifting acts, each of which involved a different man lifting a piano.
- The first is called the **collective** reading, the second the **distributive** reading.
- The ambiguity arises because the agent of a lifting event can either be a collection of individuals or a single individual.

Collective and distributive readings in SFLM

In SFLM, both the collective and the distributive reading exist:

Example

12 and 25 are coprime.

read as: $\mathit{coprime}(12, 25)$

2 and 3 are prime numbers.

read as: $\mathit{prime\_number}(2) \wedge \mathit{prime\_number}(3)$

Collective vs. transitive use of “coprime”

- Instead of “12 and 25 are coprime”, one could also say “12 is coprime to 25.”
- So the adjective “coprime” can be used in two grammatically distinct ways, but in both cases refers to the same mathematical binary relation:
  - either it is (predicatively or attributively) attached to a plural NP that gets a collective reading,
  - or it has as a complement a prepositional phrase with “to”.
- When used in the first way, we call “coprime” a **collective adjective**, when used in the second way, a **transitive adjective**.

Grouped arguments

- We say that the two logical arguments of "coprime" are **grouped** into one collective linguistic argument, a plural NP with collective reading.
- In general, mathematical adjectives expressing a symmetric binary relation have these two uses:
  - "parallel"
  - "equivalent"
  - "distinct"
  - "disjoint"
- In the case of "distinct" and "disjoint", the preposition used for the transitive case is "from" rather than "to".
- Other cases of grouped arguments are:
  - "x and y commute" (cf. "x commutes with y")
  - "x connects y and z" (cf. "x connects y to z").
  - "x is between y and z" is an example of an expression with a grouped argument for which there is no corresponding expression without grouped arguments.

Grouped arguments continued

- Since “prime number” expresses a unary relation, it is not possible to group two of its logical arguments into a single linguistic argument.
- This explains why “2 and 3 are prime numbers” can’t have a collective reading of the sort that “12 and 25 are coprime” has.
- We know of no example where a mathematical expression has a linguistic argument that can be either a collectively interpreted plural NP or a singular NP (and can hence also be a distributively interpreted plural NP), and could therefore give rise to an ambiguity like that of the *piano* sentence.

Outline

1. Introduction
2. Proof Representation Structures
3. Collective vs. distributive readings of plurals
4. **Scope ambiguity**
5. Pairwise interpretations of collective plurals
6. The plural interpretation algorithm
7. Related and future work
8. Conclusion

Scope ambiguity

In sentences containing a noun phrase conjunction and a quantifier, there can be a certain kind of scope ambiguity:

Example

A and B contain some prime.

- This can mean either that A contains a prime and B contains a (possibly different) prime, or that there is a prime that is contained in both A and B.
- In the first case we say that the scope of the noun phrase conjunction "A and B" contains the quantifier "some", whereas in the second case we say that the scope of "some" contains the noun phrase conjunction.
- We call the first reading the **wide-conjunction-scope** reading and the second the **narrow-conjunction-scope** reading.

Forced readings

Sometimes certain considerations of reference or variable range force one of the two readings:

Example

x and y are integers such that some odd prime number divides $x + y$.

x and y are prime numbers p such that some odd prime number q divides $p + 1$.

- The first example only has a narrow-conjunction-scope reading, because the existentially introduced entity is linked via a predicate ("divides") to a term ("$x + y$") that refers to the coordinated noun phrases individually.
- The second example only has a wide-conjunction-scope reading, because the variable p must range over the values of both x and y, and q depends on p.

Disambiguation in Naproche

- In general, there is, like in common language use, a strong tendency in SFLM texts to resolve scope ambiguities by giving wider scope to a quantifier that is introduced earlier in a sentence than to a quantifier introduced later in the sentence.
- This is a principle that we have already long ago adopted into Naproche in order to avoid scope ambiguities in the Naproche CNL.
- With the addition of coordinated NPs, we extended this principle to their scopes, with the exception of cases like the first example on the previous slide, where another reading is forced by certain syntactical considerations.
- In the mathematical texts that we have worked with, this extended principle always gave the intended reading.

Outline

1. Introduction
2. Proof Representation Structures
3. Collective vs. distributive readings of plurals
4. Scope ambiguity
5. **Pairwise interpretations of collective plurals**
6. The plural interpretation algorithm
7. Related and future work
8. Conclusion

Pairwise interpretations of collective plurals

Example

7, 12 and 25 are coprime.
All lines in A are parallel.

are interpreted in a pairwise way as follows:

Example

$\mathit{coprime}(7, 12) \wedge \mathit{coprime}(12, 25) \wedge \mathit{coprime}(7, 25)$
$\forall x, y \in A\ (x \neq y \rightarrow \mathit{parallel}(x, y))$

Explicit use of “pairwise”

- Sometimes, especially in connection with the negative collective adjectives “distinct” and “disjoint”, this interpretation is reinforced through the use of the word “pairwise”, in order to ensure that one applies the predicate to all pairs of objects collectively referred to by the plural NP.
- But given that this pairwise interpretation is at any rate the standard interpretation of such sentences even in the absence of the adverb “pairwise”, we decided not to require the use of the word “pairwise” in the Naproche CNL.

Pairwise interpretation in Naproche

- The Naproche CNL allows only this pairwise interpretation for a plural NP that is used as a grouped argument of such a collective adjective.
- Which adjectives classify as *collective adjectives* is coded into the lexicon of the Naproche CNL.
- In most of such cases in SFLM texts, the pairwise interpretation is the intended interpretation.

Example with other interpretation

- Below is a sentence where another reading might naturally be preferred.
- However, it seems to us that such sentences hardly appear in real mathematical texts.

Example

Some numbers in A and B are coprime.

Pairwise reading (Naproche)

$\exists n, m\ (number(n) \wedge n \in A \wedge n \in B \wedge number(m) \wedge m \in A \wedge m \in B \wedge coprime(n, m))$

Naturally preferred reading

$\exists n, m\ (number(n) \wedge n \in A \wedge number(m) \wedge m \in B \wedge coprime(n, m))$

Outline

1 Introduction

2 Proof Representation Structures

3 Collective vs. distributive readings of plurals

4 Scope ambiguity

5 Pairwise interpretations of collective plurals

6 **The plural interpretation algorithm**

7 Related and future work

8 Conclusion

PRS construction

- The PRS construction algorithm for the representation of single sentences is similar to the standard threading algorithm for DRS construction.
- It is implemented in Prolog.
- This algorithm has been adapted in order to cope with plurals, plural ambiguity resolution and pairwise interpretations as explained in the previous sections.
- We illustrate how the algorithm treats plurals by considering an example sentence.

Example sentence

Example

x and y are distinct primes p such that $2p + 1$ is a square number and some odd prime divides $x + y$.

- The algorithm works by first producing a preliminary representation.
- Here the NP conjunction gets a plural discourse referent (p), which is linked to the discourse referents of the conjuncts by a **plural_dref condition**.
- We give the NP conjunction wide scope over all quantifiers introduced later, and all assertions made in the scope of the plural NP are inserted in a special **plural sub-PRS**.

$$\begin{array}{|l|}\hline x, y, p \\ \hline \text{plural\_dref}(p,[x,y]) \\ \text{plural}\left(p, \begin{array}{|l|}\hline a,\ b, c \\ \hline \text{distinct}(p) \\ \text{prime}(p) \\ a = 2p + 1 \\ \text{square}(a) \\ \text{odd}(b) \\ \text{prime}(b) \\ c = x + y \\ \text{divide}(b,c) \\ \hline \end{array}\right) \\ \\ \hline \end{array}$$

First step of the algorithm

Now a number of processes yield the final PRS:
In the plural sub-PRS, we mark every PRS conditions which consists of a predicate that has the plural discourse referent as grouped argument:

$$\begin{array}{|l|}\hline x, y, p \\ \hline \text{plural\_dref}(p,[x,y]) \\ \text{plural}(p, \begin{array}{|l|}\hline a,\ b, c \\ \hline \text{distinct}(p) \\ \text{prime}(p) \\ a = 2p + 1 \\ \text{square}(a) \\ \text{odd}(b) \\ \text{prime}(b) \\ c = x + y \\ \text{divide}(b,c) \\ \hline \end{array}\ ) \\ \\ \hline \end{array} \quad \leadsto \quad \begin{array}{|l|}\hline x, y, p \\ \hline \text{plural\_dref}(p,[x,y]) \\ \text{plural}(p, \begin{array}{|l|}\hline a,\ b, c \\ \hline \text{distinct}(p) \\ \text{prime}(p) \\ a = 2p + 1 \\ \text{square}(a) \\ \text{odd}(b) \\ \text{prime}(b) \\ c = x + y \\ \text{divide}(b,c) \\ \hline \end{array}\ ) \\ \\ \hline \end{array}$$

Second step of the algorithm

In the plural sub-PRS, we recursively mark all PRS conditions that weren't marked in step 1 and contain the plural discourse referent or a marked discourse referent, and all discourse referents contained in a PRS condition marked in this way, until no more conditions and discourse referents can be marked by this process.

$$\begin{array}{|l|}\hline x, y, p \\ \hline \text{plural\_dref}(p,[x,y]) \\ \text{plural}(p, \begin{array}{|l|}\hline a,\ b, c \\ \hline {\color{green}\text{distinct}(p)} \\ \text{prime}(p) \\ a = 2p + 1 \\ \text{square}(a) \\ \text{odd}(b) \\ \text{prime}(b) \\ c = x + y \\ \text{divide}(b,c) \\ \hline \end{array}) \\ \\ \hline \end{array} \quad \rightsquigarrow \quad \begin{array}{|l|}\hline x, y, p \\ \hline \text{plural\_dref}(p,[x,y]) \\ \text{plural}(p, \begin{array}{|l|}\hline {\color{red}a},\ b, c \\ \hline {\color{green}\text{distinct}(p)} \\ {\color{red}\text{prime}(p)} \\ {\color{red}a = 2p + 1} \\ {\color{red}\text{square}(a)} \\ \text{odd}(b) \\ \text{prime}(b) \\ c = x + y \\ \text{divide}(b,c) \\ \hline \end{array}) \\ \\ \hline \end{array}$$

Third step of the algorithm

All discourse referents and PRS conditions in the plural sub-PRS not marked in the second step get pulled out of the plural sub-PRS and inserted into its super-PRS:

$$\begin{array}{|l|}\hline x, y, p \\ \hline \text{plural\_dref}(p,[x,y]) \\ \text{plural}(p, \begin{array}{|l|}\hline a,\ b, c \\ \hline \text{distinct}(p) \\ \text{prime}(p) \\ a = 2p+1 \\ \text{square}(a) \\ \text{odd}(b) \\ \text{prime}(b) \\ c = x+y \\ \text{divide}(b,c) \\ \hline \end{array}) \\ \\ \hline \end{array} \quad \rightsquigarrow \quad \begin{array}{|l|}\hline x, y, p, b, c \\ \hline \text{plural\_dref}(p,[x,y]) \\ \text{distinct}(p) \\ \text{odd}(b) \\ \text{prime}(b) \\ c = x+y \\ \text{divide}(b,c) \\ \text{plural}(p, \begin{array}{|l|}\hline a \\ \hline \text{prime}(p) \\ a = 2p+1 \\ \text{square}(a) \\ \hline \end{array}) \\ \\ \hline \end{array}$$

Fourth step of the algorithm

For every PRS condition $p(d)$ with grouped argument d, and every pair d_1, d_2 of distinct discourse referents linked to d via a plural_dref condition, we create a PRS condition of the form $p(d_1, d_2)$ and remove the original PRS condition $p(d)$.
In our example this amounts to replacing "distinct(p)" by "distinct(x, y)":

$$
\begin{array}{|l|}
\hline
x, y, p, b, c \\
\hline
\text{plural\_dref}(p,[x,y]) \\
\text{distinct}(p) \\
\text{odd}(b) \\
\text{prime}(b) \\
c = x + y \\
\text{divide}(b,c) \\
\text{plural}\left(p, \begin{array}{|l|} \hline a \\ \hline \text{prime}(p) \\ a = 2p + 1 \\ \text{square}(a) \\ \hline \end{array} \right) \\
\\
\hline
\end{array}
\quad \rightsquigarrow \quad
\begin{array}{|l|}
\hline
x, y, p, b, c \\
\hline
\text{plural\_dref}(p,[x,y]) \\
\text{distinct}(x, y) \\
\text{odd}(b) \\
\text{prime}(b) \\
c = x + y \\
\text{divide}(b,c) \\
\text{plural}\left(p, \begin{array}{|l|} \hline a \\ \hline \text{prime}(p) \\ a = 2p + 1 \\ \text{square}(a) \\ \hline \end{array} \right) \\
\\
\hline
\end{array}
$$

Last step of the algorithm

For every discourse referent d linked to the plural discourse referent p, we make a static copy of the plural sub-PRS in which every instance of p is replaced by d, removing the original plural sub-PRS:

$$\begin{array}{|l|}\hline x, y, p, b, c \\ \hline \text{plural\_dref}(p,[x,y]) \\ \text{distinct}(x, y) \\ \text{odd}(b) \\ \text{prime}(b) \\ c = x + y \\ \text{divide}(b,c) \\ \text{plural}(p, \begin{array}{|l|}\hline a \\ \hline \text{prime}(p) \\ a = 2p + 1 \\ \text{square}(a) \\ \hline \end{array}) \\ \\ \hline \end{array} \rightsquigarrow \begin{array}{|l|}\hline x, y, b, c \\ \hline \text{distinct}(x, y) \\ \text{odd}(b) \\ \text{prime}(b) \\ c = x + y \\ \text{divide}(b,c) \\ \text{static}(\begin{array}{|l|}\hline a \\ \hline \text{prime}(x) \\ a = 2x + 1 \\ \text{square}(a) \\ \hline \end{array}) \\ \text{static}(\begin{array}{|l|}\hline a \\ \hline \text{prime}(y) \\ a = 2y + 1 \\ \text{square}(a) \\ \hline \end{array}) \\ \\ \hline \end{array}$$

Final interpretation of example

Example

x and y are distinct primes p such that $2p + 1$ is a square number and some odd prime divides $x + y$.

$$\begin{array}{|l|}\hline x, y, b, c \\ \hline \text{distinct}(x, y) \\ \text{odd}(b) \\ \text{prime}(b) \\ c = x + y \\ \text{divide}(b, c) \\ \text{static}\left(\begin{array}{|l|}\hline a \\ \hline \text{prime}(x) \\ a = 2x + 1 \\ \text{square}(a) \\ \hline \end{array}\right) \\ \\ \text{static}\left(\begin{array}{|l|}\hline a \\ \hline \text{prime}(y) \\ a = 2y + 1 \\ \text{square}(a) \\ \hline \end{array}\right) \\ \\ \hline \end{array}$$

Motivation of the algorithm

This algorithm might seem a bit *ad hoc*, but one can actually motivate it as follows:

- The goal of the algorithm is to eliminate the plural discourse referents in favour of the singular discourse referents they subordinate.
- This has to be done separately for the distributively and collectively interpreted parts.
- The distributive interpretations opens a scopus, in which there may occur dependent variables.
- Now the five steps of the algorithm can be motivated as follows:

Motivation of the algorithm steps

1. Mark the collective uses of the plural referent.
2. Mark the distributive uses of the plural referent and dependent variables.
3. Separate the scopus of distributive uses of the plural referent from the rest.
4. Replace collective variable occurrences.
5. Replace distributive variable occurrences.

Outline

1. Introduction
2. Proof Representation Structures
3. Collective vs. distributive readings of plurals
4. Scope ambiguity
5. Pairwise interpretations of collective plurals
6. The plural interpretation algorithm
7. **Related and future work**
8. Conclusion

Comparison to ACE

- The syntax of Attempto Controlled English (ACE) allows plurals, which are interpreted in ACE in an unambiguous way.
- The disambiguation used by ACE is very distinct from Naproche's:
    - While Naproche gives preference to distributive and wide-conjunction-scope readings, ACE allows only collective and narrow-conjunction-scope readings, unless the word "each" is used.
- This difference is due to the fact that for Naproche we focused on the interpretations common in SFLM, whereas ACE took the English language as a whole into account.
- Our focus on mathematical language also made it important for us to treat "x and y are coprime" and "x is coprime to y" as logically equivalent, which ACE does not do.

Comparison to ForTheL

- ForTheL is the controlled natural language of the System for Automated Deduction (SAD), a project with similar goals to Naproche.
- ForTheL already included the two uses of words like "parallel" and "to commute" and produced the same representation no matter in which way they were used.

Example

"L and M are parallel." $\rightsquigarrow$ "L is parallel to M."

Future work

- At the moment, Naproche does not yet allow anaphoric pronouns like “it” and “they”.
- When Naproche is extended to allow them, some rules specifying how to control the many ways in which an anaphoric antecedent for “they” can be chosen will have to be specified and implemented, again with special attention to existing usage in SFLM.
- It is desirable to systematically evaluate to which extend plural constructions in mathematical texts are correctly interpreted by our plural interpretation algorithm.

Outline

1 Introduction

2 Proof Representation Structures

3 Collective vs. distributive readings of plurals

4 Scope ambiguity

5 Pairwise interpretations of collective plurals

6 The plural interpretation algorithm

7 Related and future work

8 Conclusion

Conclusion

We have implemented a plural intepretation algorithm that can handle a number of constructs related to plurals in a way that seems desirable for a mathematical CNL:

- A distributive reading of plurals is preferred.
- A collective reading is chosen for predicates with grouped arguments.
- The pairwise interpretation of predicates with grouped arguments is chosen when feasible.
- The scope ambiguity that noun phrase conjunctions give rise to is disambiguated in an intelligent way.

Thanks for listening

Try out the web interface to Naproche:
`http://www.naproche.net`