Dreistadt: A language enabled MOO for language learning

Till Christopher Lech[1] and **Koenraad de Smedt**[2]

Abstract. Dreistadt is an educational MOO (Multi User Domain, Object Oriented) for language learning. It presents a virtual world in which learners of German communicate with their fellow learners, teachers and native language users in other locations via the Internet. While the original Dreistadt had an artificial command language for interaction with the system, we have provided it with natural language processing capabilities, in order to allow a more seamless linguistic interaction. For this purpose, an NLP interface for controlled German has been added. The student's natural language commands are translated to system internal instructions by a set of syntactic, semantic and pragmatic analysis tools. The system is capable of handling pronouns and other referring expressions by applying domain knowledge and includes an inferencing component based on predicate logic.

1 BACKGROUND

A *MOO* (Multi User Domain, Object Oriented) is a programmable platform for network based interaction between role players in a virtual world [1]. Like chat programs, MOO-based environments primarily enable the communication between multiple users, but in addition, this communication is set in an artificial world in which the players move about and act.

MOOs are typically used for multi-user computer games, but they are also appreciated as learning environments [4, 3]. There are a number of so-called educational MOOs, examples of which are the Lingua MOO at the Dallas University and the MediaMoo at Georgia Tech[3]. The educational appeal is in the fact that communication takes place in a virtual game reality which strongly appeals to the students' creativity and fantasy. The students participate in a reality which, though virtual, ideally can be transformed into an integral part of the students' inner learning universe [10].

Dreistadt is such an environment for learners of German, and presents a virtual world resembling a German village. It is implemented in enCore, a Web-based MOO client specifically developed for eductional purposes [3]. A screenshot of Dreistadt is shown in Figure 1, where the following four windows are displayed:

1. Icons and menus for object manipulation.
2. Messages from the system and from other users.
3. A graphical and textual representation of the virtual world, its objects and its status.
4. An area for writing commands and messages to other users.

[1] CognIT, Norway, email: till.lech@cognit.no
[2] University of Bergen, email: desmedt@uib.no
[3] A more comprehensive list of educational MOOs can be found at http://moolist.yeehaw.net/edu.html

The Dreistadt world appears to the student as a German village with buildings and rooms in which the students enter, move about, interact with other participants, manipulate objects, and exit. The main purpose of Dreistadt as a learning platform is to stimulate the communication in German between participants. In this respect, it works somewhat like a chat room, with the important difference that a chat room is unstructured, whereas the interaction in Dreistadt is set in a context defined by the virtual world and guided by assignments to the students.

The possible movements and actions in this virtual world, carried out by the student's *avatar* (role figure), may include, e.g., reading road signs, going places, exchanging virtual objects with other users, reading letters, etc. These actions imply the need to communicate not only with other users, but also with the system itself. In the original version of Dreistadt, this is achieved by means of a command language for navigation and manipulation. An excerpt of an interaction log in the original version of Dreistadt is shown in Example 1, where commands to the system are preceded with an asterisk, while calls to other participants are preceded by an asterisk and a double quote.

```
(1) #$#mcp version: 2.1 to: 2.1
  Till kommt um die toten Augen zu besuchen.
  * "Hallo Till!
  Du rufst: "Hallo Till!"
  Till meint: "Hallo! Ich habe gleiche eine
   Literaturvorlesung und..."
  Till ruft: "...muss noch den Projektor für
   den Vrotrag vorbereiten!"
  Till gibt Dir den Projektor 'uibpro'.
  * schau Projektor
  Ich sehe kein "Projektor" hier.
  * schau uibpro

  Du betrachtest den Projektor 'uibpro' ...
   <http://cmc.hf.uib.no:7001/3640/>.
  * gib uibpro an Till
  Du gibst Till den Projektor 'uibpro'.
  Till sagt: "hier hast du noch einen Brief"
  Till gibt Dir das Dokument 'Brief'.
  * "Danke!
  Du rufst: "Danke!"
  * lies Brief

  Du betrachtest das Dokument 'Brief' ...
   <http://cmc.hf.uib.no:7001/907/>.
  ----------------------------------------
  Vorlesungsplan für die nächste Woche:
```

```
Mittwoch und Donnerstag, 18-20,
anschliessend Zeit für Gruppenübungen!
---------------------------------------
(Du bist fertig mit dem Lesen.)

* gehe Universität
Du betrachtest die Universität ...
```

Communication among students or between student and teachers is obviously supposed to be in natural German, allowing the teachers to monitor the development of the students' use of German. Examples of such interaction are the student's utterance *Hallo Till* in Example 1, and Till's reply *Hallo! Ich habe gleiche eine Literaturvorlesung und ... muss noch den Projector für den Vrotrag vorbereiten!* (Hello! I have right now a Literature lecture and ... still have to prepare the projector for the persentation[misspelled]).

In contrast to the natural language communication between users, any commands to the system had to be given in an artificial command language in the original Dreistadt. Although this command language was based on German vocabulary, its syntax was severely 'crippled' compared to German syntax, as can be seen from Example 1. Indeed, commands like *schau Projector* (look projector), *lies Brief* (read letter) and *gehe Universität* (go university) are equally unidiomatic as their English counterparts.

The original command language syntax was basically limited to that in definition 2, where parentheses denote optionality and *name* is the name of someone or something in the virtual world.

(2) imperative (name (adverb | particle | preposition name))

The complete absence of function words from this syntax, among other things, practically prevents the construction of any grammatical German sentences at all. Moreover, the command language has other severe restrictions in the semantic and pragmatic domain. While object names can be used after verbs, object classes (including hypernyms) cannot be used. In the above Example, *schau Projector* (look projector) gives a negative result, but when the projector *uibpro* is mentioned by name, the result is positive. It is also impossible to refer to objects in other ways than their names, e.g. by means of pronouns or other referring expressions.

Furthermore, commands are basically imperatives. The command language does not allow for questions, even though the possibility of question answering would be very useful to the participants, and even though relevant object properties, e.g. concerning the location of objects, are in fact technically accessible for this purpose.

Since the choice between the command language and natural German is clearly dependent on the addressee, the code switching is not too difficult for users. Still, it is unfortunate from a pedagogical point of view, because the student needs to learn two languages at once, thereby complicating the learning process. Also, there is always a risk of command language syntax slipping into the student's conversational German.

2 OVERVIEW OF THE NLP INTERFACE

In order to overcome the limitations outlined in the previous section, we built a language enabled version of Dreistadt where the artificial command language was substituted by an appropriate subset of German. In the new system, students can use correct and idiomatic natural language in their interaction with the system as well as with other human participants. The assumption is that this will increase the language learning effect.

This change required new linguistic capabilities on the part of the system, not only for analyzing German sentences syntactically, but also for their semantic and pragmatic representation, and finally their translation to system internal instructions that access or manipulate properties of objects in the Dreistadt domain.

Some of the challenges of our endeavour are represented by Examples 3–5 which were included in the controlled German for the language enabled version. Among the syntactic challenges are the analysis of questions (Example 3) and imperatives (Example 4), with a variety of complementation patterns and word orders. A semantic challenge is represented by distinguishing between concrete objects and their classes, as in Examples 3 and 4, where *Testarchiv* is the name of a specific object and *Archiv* refers to any object in the generic class of 'file cabinets'. One of the pragmatic challenges is the handling of anaphoric pronouns as in Example 5.

(3) Liegt das Buch im Testarchiv? (Does the book lie in the test archive?)
(4) Lege das Dokument in das Archiv. (Put the book in the archive.)
(5) Nimm es aus dem Testarchiv. (Take it out of the test archive.)

We have found other reports of research on language enabled MOOs in the literature [9, 14, 8], which in our view do not achieve the same semantic depth as the work in the present paper, as will be discussed below. Our strategy consists of processing input sentences by means of the following steps, which in the remainder of the paper will be described in more detail.

1. Preprocessing: transformation of context objects into lexical items and database facts.
2. Syntactic processing: chart parsing with a unification based grammar.
3. Semantic and pragmatic processing: transformation of feature structures into terms and application of inference rules.
4. Postprocessing: transformation from semantic structures into MOO-instructions by means of an ATN (Augmented Transition Network).

The interface and the MOO run on different machines, connected by a data stream in a client-server setup. While the MOO runs on a Windows NT server, all NLP programs run in Allegro Common Lisp on a Sun Solaris workstation. The main NLP tools (chart parser and ATN) were slightly adapted versions of Lisp implementations described in the literature [2].

2.1 Preprocessing

In the preprocessing stage, the student's utterance is explicitly tied to its context, i.e. the objects in the vicinity of the role figure representing the student. This context includes the role figure's location (i.e. virtual room) in *Dreistadt*, as well as all MOO-objects that are located in the room. Thus a typical context includes a room, and possibly other players, or things they may have created or found. All context objects are represented by their unique identifier, which consists of a hash symbol followed by an integer (e.g. #1842 for the object named *Testarchiv*).

Together with these *context objects*, the utterance is put in a data stream to the NLP modules. On the NLP side, the first step consists of the extraction of lexical units from information in the context objects, as illustrated in Example 6 for the object *Testarchiv*. The lexicon is incrementally extended as objects are created in the virtual world. A description of lexical features is given in section 3.3.

```
(6) (WORD (TESTARCHIV)
    (CAT) = N
    (GENDER) = NEUTRUM
    (PRE PREDICATE) = NAME
    (PRE ARG1) = TESTARCHIV
    (PRE ARG0) = (REFERENT))
```

Furthermore, selected facts extracted from the context objects are extracted and added to a database which is later used for inference. In Example 7, object #1842's class (*$Dokumentenarchiv*, a generic document archive), its name (*Testarchiv*), its location (a room with object identifier #874) and its gender (*neuter*) are transformed into database facts.

```
(7) ((CLASS #1842 $DOKUMENTENARCHIV))
    ((NAME #1842 TESTARCHIV))
    ((LOCATION #1842 #874))
    ((GENDER #1842 NEUTRUM)))
```

2.2 Syntactic processing

Input sentences are analyzed with a unification based chart parser implemented in Lisp [2]. The parser was driven by our PATR-style [11] unification grammar [13] and lexicon for German. Further details on the grammar are given in section 3.3. The lexicon consists of two parts:

- A *fixed* part that includes function words as well as open class words with a constant denotation such as the nouns for the generic object classes in the MOO.
- A *dynamic* part, which includes all the nouns in a given context, based on information from the current Dreistadt context objects.

Since the dynamic lexicon is based on the given context of the user command, its entries must be automatically generated on the fly. As mentioned in section 2.1, this is achieved by retrieving the object's name and gender from the MOO database, as this information must be provided upon any object's creation. Furthermore, noun-specific semantic and pragmatic information is added to the lexicon entry, as illustrated in 6.

The parser output consists of two feature structures, one representing the semantics and one representing a set of presuppositions concerning the sentence's meaning. Both of these structures are converted into terms. The sentence in Example 8 gives rise to a semantic structure represented as a term (Example 9) and presuppositions (Example 10). It should be noted that these structures, at this stage, may contain a number of logical variables (starting with an underscore).

(8) Lege das Buch in das Testarchiv. (Put the book in the test archive.)

```
(9) (PUT_IN _505 _4018)
(10) (AND (FIT_IN _505 _4018)
          (AND (NAME _505 BUCH)
               (NAME _4018 TESTARCHIV)))
```

2.3 Semantic and pragmatic processing

The semantic structure in Example 9 specifies only that something (represented by the logical variable _505) has to be put into something else (represented by _4018). At the same time, the presuppositions specify a number of conditions that need to be true for whatever objects that may instantiate the logical variables. In the presuppositions shown in Example 10, these objects must be named *Buch* (book) and *Testarchiv* (test archive) respectively, and we also assume that the first one must fit into the other one, as specified by the *FIT_IN* predicate, in order to be put into it.

These conditions are checked against a database of rules and facts extracted from the context objects in the preprocessing step. Suppose that there are two objects that match the respective logical variables, as represented by the variable bindings in Example 11, then these objects may replace the variables, resulting in an interpretation (Example 12), which is a fully instantiated semantic structure (as compared to Example 9).

```
(11) ((_505 #2820) (_4018 #1842)))
(12) (PUT_IN #2820 #1842)
```

2.4 Postprocessing

The final processing step consists of converting the instantiated semantic structure into instructions internal to the MOO system. This is achieved by an Augmented Transition Network (ATN, [12]) implemented in Lisp [2]. The choice of the ATN formalism was motivated by the need for flexibility, while linguistic complexity at this level is limited. For instance, the interpretation in Example 12 was translated into the instruction in Example 13. The instruction is sent back on the client-server connection to the MOO, where it is executed.

```
(13) (@move #2820 to #1842)
```

3 INFERENCING

An important part of any NLP interface consists of a semantic and pragmatic component which lend the interface robustness with respect to targeting the user's intentions. Our approach to this level of processing is strongly based on inferencing with predicate logic. This section first discusses how inferencing plays a role in question answering, then how presuppositions are used in anaphor resolution, and finally explains how presuppositions are computed with the help of our grammar.

3.1 Question answering

In contrast to earlier approaches[9, 14, 8], our semantic structures are based on predicate calculus with logical connectives and quantified expressions, as depicted in Example 15, which is our semantic representation of the natural language question in Example 14.

(14) Passen alle Gegenstände, die ins Testarchiv passen, auch in die Ideenkiste? (Do all objects, which fit in the test archive, also fit in the idea box?)

```
(15) (all _x
       (fit_in _x Testarchiv)
       (fit_in _x Ideenkiste))
```

Questions are resolved by means of inferencing on a database containing MOO-specific inference rules and facts concerning the objects currently present in the virtual world. Some rules are given in Example 16, which specifies that anything of class *$thing* fits into anything of class *$container*, and in Example 17, which specifies that anything of class *$note* fits into anything of class *$dokumentenarchiv* (document archive). To the extent that all notes are also things, the question in Example 14 can be answered as true.

```
(16) ((fit_in _x _y)
       (class _x $thing) (class _y $container))
```

```
(17) ((fit_in _x _y)
     (class _x $note)
     (class _y $dokumentenarchiv))
```

We use logical inference based on standard backward chaining; in other words, inference is goal driven. In order to check the truth of a term, we attempt to prove it by means of any rules. When a term matches the conclusion in a rule (which in our notation is the first element of the rule), we attempt to prove the premises (which in our notation are the remaining elements of the rule). Only if all premises are true, the conclusion is considered true. Facts are rules without any premises.

3.2 Anaphora resolution

The inferencing engine is also extensively used for anaphora resolution. Whereas some earlier work [9, 14] attempts to achieve resolution of anaphoric pronouns in a MOO by means of pattern matching heuristics, taking into account e.g. gender agreement, we employ a logical inference mechanism on all referring expressions, not only pronouns, but also e.g. nominalized adjectives as in Example 18.

(18) Nimm *den roten* aus der Sammelkiste. (Take *the red one* out of the collecting box.)

```
(19) (TAKE _8532)
(20) (AND (TAKEABLE _8532)
       (AND (ROT _8532)
         (AND (LOCATION _8532 _10012)
           (NAME _10012 SAMMELKISTE))))
```

The analysis of the sentence in Example 18 produces a semantic structure (Example 19) and presuppositions (Example 20). While the semantic structure does not specify more than that something needs to be taken, the presuppositions specify that the object to be taken needs to be 'takeable' as well as red, and that its location is something with the name *Sammelkiste* (collecting box). When these conditions are satisfied, the logical variable in Example 19 can be instantiated.

In the case of pronouns, which are marked for gender in German, a gender match is included as one of the presuppositions, as for Example 21 which obtains a semantic structure (Example 22) and a set of presuppositions (Example 23). Apart from this gender matching, anaphora resolution in our NLP interface can be called knowledge based, in the spirit of related work based on domain knowledge and ontologies [5, 6].

(21) Nimm es aus dem Testarchiv. (Take it out of the test archive.)

```
(22) (TAKE _326)
(23) (AND (TAKEABLE _326)
       (AND (GENDER _326 NEUTRUM)
         (AND (LOCATION _326 _1394)
           (NAME _1394 TESTARCHIV))))
```

3.3 Presupposition-oriented grammar

In order to support the extensive inferencing possibilities, the grammar and lexicon need to generate the appropriate presuppositions. Since our unification grammar is a lexicalized grammar, this is crucially achieved by including appropriate features in the lexicon. By way of example, consider the lexical entry for *nimm* (take) in Example 24.

```
(24) (Word (nimm)
       (cat) = V
       (sem predicate) = take
       (sem arg1) = (arg1)
       (pre predicate) = takeable
       (pre arg0) = (arg1))
```

Besides specifying the predicate and first argument in the semantic structure, the lexical entry in Example 24 also specifies part of the presuppositions in Example 23, namely that the first argument of the verb *take* should also be the first argument (i.e. arg0) of a predicate *takeable*, in other words, whatever needs to be taken should be 'takeable'.

Another case in point is the lexical entry for *Buch* (book) in Example 25, which, besides providing morphosyntactic features, specifies that a presupposition be constructed with the predicate *name*, and with the referent (designating the MOO item, e.g. #543) and *Buch* as arguments, in other words, the referent should be a book. An illustration of the use of this presupposition is given in Example 10.

```
(25) (Word (Buch)
       (cat) = N
       (gender) = NEUTRUM
       (pre predicate) = name
       (pre arg1) = Buch
       (pre arg0) = (referent))
```

The assembly of the complete semantic structures and presuppositions is executed at the levels of the phrase and the clause. In Example 27, we find the rule for the simple imperative sentence in Example 26. Besides a description of the phrase constituents and their morphosyntactic features, this rule contains the semantic representation of the imperative phrase `(IMP sem)`, which in this case is identical with `(V sem)`, i.e. the verb's semantics (as shown in Example 24).

(26) Nimm das Buch! (Take the book!)

```
(27) (Rule (IMP -> V NP)
       (IMP cat) = IMP
       (V cat) = V
       (NP cat) = NP
       (NP case) = akkusativ
       (V arg1) = (NP referent)
       (IMP sem) = (V sem)
       (IMP pre connective) = and
       (IMP pre prop1) = (V pre)
       (IMP pre prop2) = (NP pre))
```

Furthermore, the rule in Example 27 provides features that provide for the assembly of the corresponding presupposition `(IMP pre)`, which is composed of `(V pre)`, i.e. the verb's presupposition information, and `(NP pre)`, i.e. the NP's presupposition information, combined with the logical connective `AND`. This ultimately leads to the presupposition feature structure as shown in Example 28 which is transformed into a term (Example 29).

```
(28) ((CONNECTIVE AND)
      (PROP1 ((PREDICATE TAKEABLE)
              (ARG0 _2348)))
      (PROP2 ((PREDICATE NAME)
              (ARG0 _2348)
              (ARG1 BUCH))))
(29) (AND (TAKEABLE _2348) (NAME _2348 BUCH)
```

4 DISCUSSION AND CONCLUSION

In conclusion, we have built a natural language enabled version of the MOO-based Dreistadt virtual world. The interface language consists of controlled German, which is analyzed at syntactic, semantic and pragmatic levels. The subset of German which the system is capable of handling can be used to give instructions in the Dreistadt world and ask questions about objects. Although the system's natural language capabilities cover only a fraction of natural German, we claim that our effort is worthwhile for a number of reasons.

We have demonstrated the feasibility of enhancing a MOO with natural language. We could identify a subset of German that naturally fits the needs of interaction in Dreistadt, because the need for navigation and manipulation in the virtual world is limited to such an extent that the interface language can be equally limited. The restricted context of the MOO allows us to explore the benefits of using natural language without facing the burden that unrestricted dialogue would otherwise represent.

Although the domain can thus be controlled, we have shown that challenges for natural language interaction in a MOO are far from trivial. For one thing, the frequent creation or discovery of new objects in the virtual world implies that the lexicon must be highly extensible, as also clearly recognized by others in this research area [9]. We have solved this by generating new lexical entries on the fly as needed during our preprocessing stage for each new sentence analysis.

Another challenge we have tackled consists of the user's references to the multitude of objects which are typically present in the virtual world. This requires a solid treatment of quantifiers, anaphors, etc. In these respects, we found that educational MOOs are a highly interesting application of language-enabled educational systems. We claim that we have gone further than any other known research in our semantic and pragmatic analysis of MOO user input, which is crucial for an adequate interpretation of user intentions.

We have built our NLP interface on a platform largely based on textbook components [2] which we have adapted and supplemented with our own grammar, lexicon, inference rules, and pre- and post-processing. The results seem to indicate that by using tools based on well proven approaches and provided with appropriate linguistic and domain information, it is possible to build good mechanisms for interpreting, and ultimately achieving, the user's intentions in this kind of context.

Not only does our new interface offer the benefit of naturalness, it also extends the functional capabilities of interaction. It is now possible to ask questions about objects in the virtual world, whereas the artificial command language did not allow for any queries. Also, it is now possible to refer to classes of objects rather than just to individual named objects.

We believe that in with respect to referring expressions, our inference based approach is more sophisticated than work previously reported for MOOs [9, 14, 8]. However, the problem of anaphor resolution is recognized as a complex one, and our approach takes into account only the domain context but not the discourse context. Although an evaluation of the accuracy of our anaphor resolution in actual use has not been undertaken, we know that it has limitations. A more complete approach to anaphor resolution needs to take into account a variety of factors including discourse factors [7].

Besides the fact that coverage could be extended further, an obvious functionality currently lacking is the generation of natural language output. While it is possible to ask questions, the formulation of answers is presently limited to quite simple responses. Although this is often sufficient, a more sophisticated interaction is required in cases where, e.g., several possible referents are found for a referring expression by the user. Indeed, discourse management and dialogue strategies in general are missing from the system, since our work has been strongly focused on the analysis side.

Although this paper mainly presents the computational linguistics aspects of our work, it has also been driven by pedagogical motivations. We think that incorporating this natural language interaction mode — rather than using an artificial language — may be pedagogically desirable, especially in language learning systems. However, any such pedagogical effects remain to be tested; at this point we lay claim only to having created a system that will allow such testing. Our NLP extension of Dreistadt is operational and has been actively used by Norwegian learners of German at the University of Bergen, who looked at it favorably and were mainly critical of the somewhat slow response times, which are an implementation issue.

We do have a number of hypotheses concerning the reception and benefits of the system over a longer time span. On the one hand, we predict a synergy effect from the students using natural language both with the system and with other human users. This should manifest itself in better learning curves and a lack of command language artifacts in the student's conversational German. On the other hand, we fear that students will at times provide input that goes beyond the system's capabilities, and will in those cases be disappointed. In some cases, students may actually prefer an artificial command language to writing fully grammatical German, as an easy way out, even if it may not be to their educational benefit.

ACKNOWLEDGEMENTS

We would like to thank Carsten Jopp and Daniel Jung for providing an opportunity to work with Dreistadt and Sindre Sørensen for technical help. The parser, semantic processing and inferencing are based on Lisp programs written by Gerald Gazdar and Chris Mellish [2] and adapted by us.

REFERENCES

[1] Pavel Curtis, 'Not just a game: How lambdamoo came to exist and what it did to get back at me', in *High Wired*, eds., Cynthia Haynes and Jan Rune Holmevik, University of Michigan Press, (1998).

[2] Gerald Gazdar and Chris Mellish, *Natural Language Processing in LISP*, Addison-Wesley, Reading, 1989.

[3] *High Wired*, eds., Cynthia Haynes and Jan Rune Holmevik, University of Michigan Press, 1998.

[4] Carsten Jopp, 'Reell læring i virtuel by. bruk av virtual-realty-omgivelser i (språk)-læringen', in *IKT og læring i humanistisk perspektiv*, ed., Carsten Jopp, Cappelen, (2001).

[5] Till Lech and Koenraad de Smedt, 'Enhancing semantic annotation through coreference chaining: An ontology-based approach', in *Proceedings of the 5th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2005) at the 4rd International Semantic Web Conference (ISWC 2005)*, eds., Siegfried Handschuh and Thierry Declerck, CEUR Workshop Proceedings, Aachen, (2005). Technical University of Aachen.

[6] Till Lech and Koenraad de Smedt, 'Ontology extraction for coreference chaining', in *Proceedings of the Workshop on Anaphora Resolution, Mjølfjell, Norway, September 28-30, 2005*, (forthcoming).

[7] Ruslan Mitkov, 'Robust pronoun resolution with limited knowledge', in *Proceedings of the 18th International Conference on Computational Linguistics*, (1998).

[8] Doug Orleans, 'Natural language processing for multi-user virtual worlds', Project report, Dept. of Computer Science, Northeastern University, (1998).

[9] S. Rochefort, V. Dahl, and P. Tarau, 'Controlling virtual worlds through extensible natural language', in *AAAI Symposium on NLP for the WWW, Stanford University, CA, 1977*, (1997).

[10] Tor Jan Ropeid, 'Studenten, læreren og det elektroniske undervisningsrom', in *IKT og læring i humanistisk perspektiv*, Cappelen, Oslo, (2001).
[11] Peter Sells, *Lectures on Contemporary Syntactic Theories*, CSLI Publications, 1985.
[12] Stuart C. Shapiro, 'Generalized augmented transition network grammars for generation from semantic networks', *American Journal of Computational Linguistics*, **8**(1), 12–25, (1982).
[13] Stuart Shieber, *An Introduction to Unification-Based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes*, CSLI Publications, 1986.
[14] Paul Tarau, Koenraad De Bosschere, Veronica Dahl, and Stephen Rochefort, 'Logimoo: An extensible multi-user virtual world with natural language control', *Journal of Logic Programming*, **38**(3), 331–353, (1999).

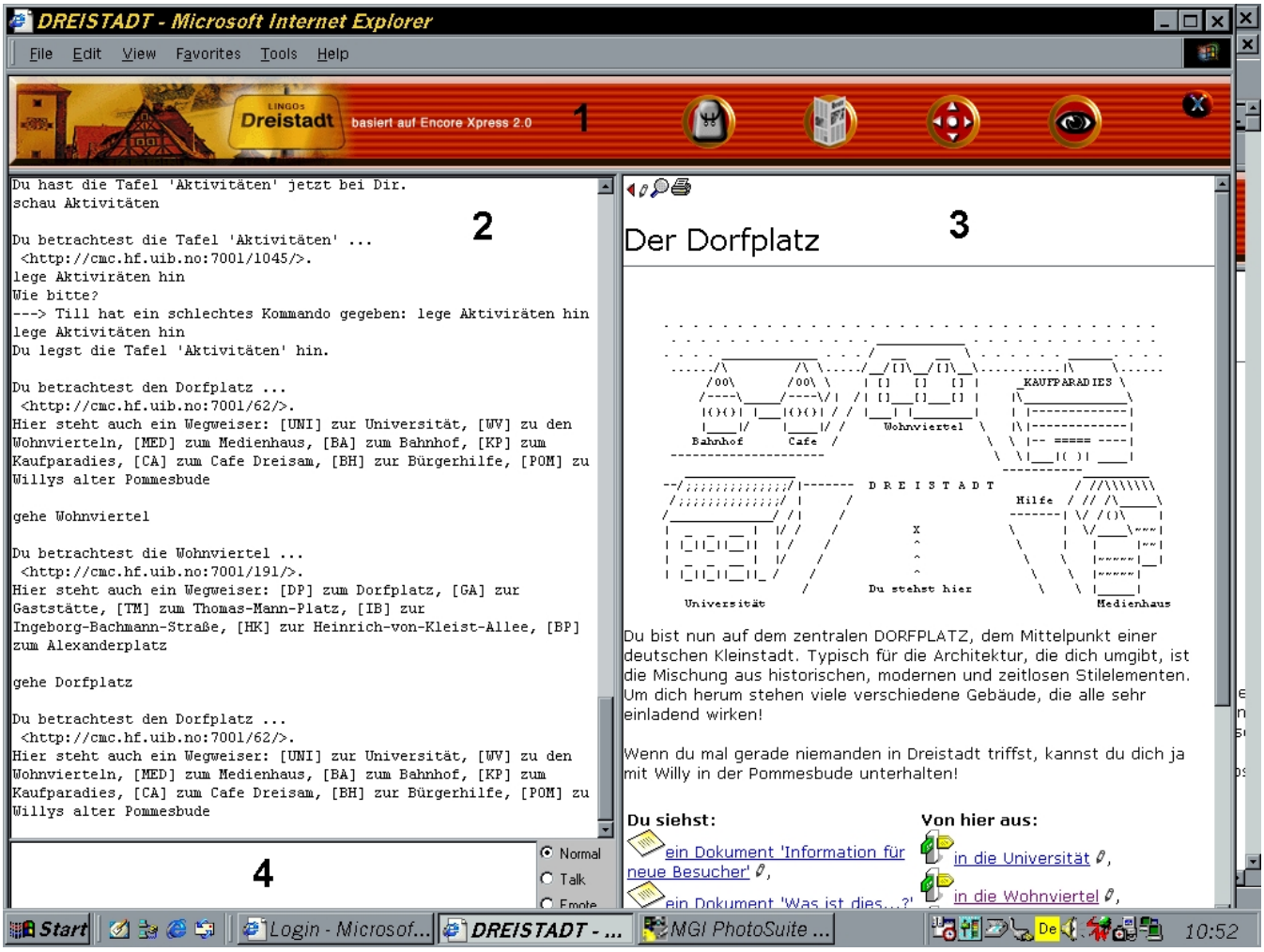

Figure 1. Dreistadt screen