Natural Language Understanding using Temporal Action Logic

Martin Magnusson
Department of Computer and Information Science
Linköping University, SE-581 83 Linköping, Sweden
email: marma@ida.liu.se
www.phlai.com/nl1

Abstract

We consider a logicist approach to natural language understanding based on the translation of a quasi-logical form into a temporal logic, explicitly constructed for the representation of action and change, and the subsequent reasoning about this semantic structure in the context of a background knowledge theory using automated theorem proving techniques. The approach is substantiated through a proof-of-concept question answering system implementation that uses a head-driven phrase structure grammar developed in the Linguistic Knowledge Builder to construct minimal recursion semantics structures which are translated into a Temporal Action Logic where both the SNARK automated theorem prover and the Allegro Prolog logic programming environment can be used for reasoning through an interchangeable compilation into first-order logic or logic programs respectively.

1 Introduction

A complex and poorly understood area in computational linguistics is the integration and use of background knowledge to aid parsing, interpretation and understanding of natural language. There is general agreement that background knowledge is needed, e.g. to select between ambiguous interpretations or to provide answers to questions, and that without at least a partial understanding of the world a system can never hope to approach full natural language understanding. As artificial intelligence research moves closer to applications there is an increasing risk that too many natural language projects concentrate on the robust performance that is required in real-world applications and that they, while realizing that background knowledge is important, tend to make its role peripheral instead of a solid base on which to build upon.

We describe a natural language understanding system based on a logicist knowledge representation foundation that serves as a research platform for experimentation with the interchange between computational linguistics and knowledge representation and reasoning. The focus is the representational and inferential adequacy of the underlying techniques, which have been selected for generality and extensibility, rather than on immediate applicability or the similarity with human dialogue characteristics. The techniques are brought together in a simple and clear architecture that holds great potential for development and experimentation. A novel integration of natural language technology, knowledge representation technology, and automated reasoning technology in a proof-of-concept question answering system, with the working title NL1, has been implemented and is available as open source[1].

2 Temporal Action Logic

The Temporal Action Logic (TAL) is a non-monotonic temporal logic developed specifically for reasoning about actions and dynamical domains. The logic has its origin in the Features and Fluents framework developed by Sandewall (1994) but was given a new characterization in terms of circumscription by Doherty (1994). Many extensions since have turned TAL into a very expressive language capable of representing, among other things, actions with durations,

[1] `http://www.phlai.com/nl1`

per1 $\forall t\ [Per(t, alive) \wedge Per(t, loaded)]$
acs1 $[t_1, t_2]\ Load \rightsquigarrow R((t_1, t_2]\ loaded)$
acs2 $[t_1, t_2]\ Fire \rightsquigarrow ([t_1]\ loaded \rightarrow R((t_1, t_2]\ \neg alive \wedge \neg loaded))$
obs1 $[0]\ \neg loaded \wedge alive$
occ1 $[1, 2]\ Load$
occ2 $[3, 4]\ Fire$

Figure 1: The Yale shooting scenario in TAL.

context-dependent and non-deterministic actions, concurrency, and action side-effects. It also provides solutions to the frame, ramification and qualification problems. For a more detailed introduction to TAL the reader is referred to (Doherty et al., 1998).

2.1 TAL Narratives

Domains are described in TAL using *fluents* that represent properties of the world that change over time. World laws governing the evolution of fluents are expressed in *narratives*, which are high-level descriptions of observations, action schemas, and action occurrences. Narratives receive a semantics through a translation to the base language, which is an order-sorted classical first-order logic together with a circumscription policy described below. Figure 1 shows the well-known Yale shooting scenario expressed as a TAL narrative. A persistence statement (labelled per1) constrains the fluents $alive$ and $loaded$'s values to persist unless they are affected by some action. Action schemas (acs1 and acs2) use the reassignment operator R to make $loaded$ true after performing the $Load$ action and to make both $loaded$ and $alive$ false after performing the $Fire$ action, but only if $loaded$ was true when the action was initiated. An observation statement (obs1) initializes $loaded$ and $alive$ to false and true respectively. Finally, the two occurrence statements (occ1 and occ2) describe a world history in which the $Load$ action is performed between time points 1 and 2, and the $Fire$ action is performed between 3 and 4.

2.2 Occlusion

The key to the solution of the frame problem in TAL lies in the use of *occlusion*. When narratives are translated into classical logic an $Occlude$ predicate is introduced and constrained to be true at those time points where fluents are forced to change their values due to reassignments. An additional *no-change* axiom is added that rules out any fluent changes at any time points when the fluent was not occluded. The final step is the minimization of the $Occlude$ predicate and the action occurrences, expressing the default assumption that no unexpected fluent changes or spurious actions occur unless explicitly specified. The minimization is accomplished through the circumscription of those parts of the translated theory that contain action schemas and action occurrences respectively.

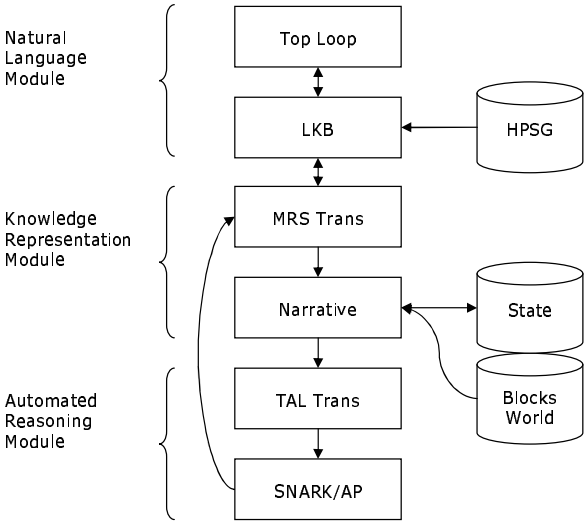

Figure 2: An overview of the NL1 architecture.

3 System Architecture

NL1 carries on an interactive natural language text dialogue with the user, executing commands and answering queries about a simulated blocksworld. An important emphasis of the system architecture, depicted in Figure 2, is the use of declarative knowledge structures in the hope of building a system that is both convenient to adapt to new usage scenarios and that has a great ultimate potential for extension while at the same time retaining the basic implementation components.

3.1 Natural Language Module

The natural language module is responsible for the parsing of input sentences and the construction of a quasi-logical form. User interaction consists of natural language input and output through a minimal user interface in the form of a text top-loop. Sentence are read from a prompt and passed as text strings to the Linguistic Knowledge Builder (LKB) component. The LKB chart parser uses a Head-driven Phrase Structure Grammar (HPSG),

based on the grammar in (Sag et al., 2003), to parse the input text string and build a feature structure representation. The grammar includes semantical relations, and the semantical part of the feature structure representation constitutes a Minimal Recursion Semantics (MRS) structure.

One of the benefits of this set-up is that natural language generation can be achieved by running the chart parser "in reverse" using the same grammar that was used when parsing. The task of generating a response is then reduced to the task of constructing a suitable MRS structure representing an answer to the user's request.

The HPSG grammar forms a declarative knowledge source that is easily adapted to new vocabularies, by changing the lexicon of words and semantical relations, and that has a great potential for extension, e.g., moving towards broad coverage as in the English Resource Grammar project (Copestake and Flickinger, 2000).

3.2 Knowledge Representation Module

The knowledge representation module holds a representation of both the basic world laws and the particulars of the current scenario, but also a history of the developments of the world during the ongoing dialogue.

The world model, together with action specifications and an initial state is encoded as TAL formulas and stored in the blocksworld knowledge base. A continually evolving TAL narrative of action occurrences represents the world history and is stored in the state knowledge base. Any sentence that enters the system reaches the knowledge representation module in the form of an MRS structure that needs to be transformed into a full logical form before it is used in reasoning. The MRS structure is passed to the translation component that implements a partial translation by performing a number of transformations, as described in Section 4, that result in a TAL formula. If the natural language module identified the sentence as an imperative command, the TAL formula is appended to the narrative, but if the sentence was identified as a proposition or a question, a reasoning problem is formed by combining the corresponding TAL formula with the background blocksworld theory.

By using TAL as our knowledge representation language we are able to express a wide range of common-sense reasoning scenarios and ensure that the system is not limited to the relatively simple blocksworld, even though it is used as an illustrative example.

3.3 Automated Reasoning Module

Performing automated reasoning in the Temporal Action Logic is not trivial since it is a non-monotonic logic with a semantics based on circumscription. When Sandewall developed the basis of the formalism he was concerned more with assessing its correctness rather than performing automated reasoning. Later Doherty and Łukaszewicz (1994) showed how the semantics could be expressed using circumscription and, in (Doherty, 1994), that under certain restrictions the circumscription policy could be reduced to predicate completion (defined by Lifschitz (1994)) which is computable and results in an equivalent first-order characterization.

These methods open up the possibility of reasoning in TAL using a very flexible compilation approach that fits in nicely with the TAL high-level narrative description to base language translation. The idea is to choose an existing domain-independent automated reasoning paradigm and develop a compilation step from the TAL base language into a language compatible with that paradigm. The most obvious example would be a compilation into regular first-order logic and the application of resolution theorem provers, but other options are certainly possible as will be described below. The flexibility lies in the postponing of the expressibility vs. tractability tradeoff decision to the time at which some specific application domain is fixed, or even later, to the time at which a specific reasoning problem needs to be addressed. For example, one would like to detect if the theory is Horn and then use a logic program compilation while still retaining the possibility to automatically fall back to first-order or even higher-order logic if necessary. Such a classification of the complexity of the reasoning problem is not possible in general, but even an incomplete version might be tremendously useful in certain domains.

Our implementation includes two different compilation mechanisms. One results in a first-order theory that is further encapsulated (adding explicit theorem proving strategies and other configurations) to form an input problem to SNARK (Stickel, 2005), a resolution theorem prover with support for, among other things, sorts and answer

(1) B1 is on the table.
Ok.
(2) The table supports B1.
Yes.
(3) Does the table support B2?
I don't know.
(4) B2 is on the table and B3 is on B1.
Ok.
(5) Does the table support B3?
No.
(6) What blocks are clear?
B3 is clear and B2 is clear.
(7) Pick up B3.
Ok.
(8) Put B3 on the table.
Ok.
(9) What blocks are on the table?
B2 is on the table and B1 is on the table and B3 is on the table.

Figure 3: An example dialogue from NL1.

extraction. The other compilation results in a logic program that is further specialized into a program for Allegro Prolog[2], a Prolog environment that is fully integrated in the Allegro Common-Lisp development environment. In both cases, depending on the question type, the result is the deduction of true or false, or the extraction of answer terms from a proof of the query formula. These answer formulas are passed back to the natural language module where they are used in the construction of an answer MRS that generates a natural language response, closing the loop.

3.4 Example Dialogue

The example dialogue in Figure 3 illustrates the working system and consists of numbered interactions between an operator and the software. The simplest sentence type is a proposition stating a fact about the world, as in sentence 1. NL1 parses the sentence, translates the result into a TAL formula, and adds it to the knowledge base. Since the knowledge is expressed using logic we can define new relations in terms of old relations using axioms, as in the *support* relation of sentence 2. Before accepting new knowledge NL1 uses its reasoning capabilities to determine if the new information contradicts the current knowledge or if it

[2] `http://www.franz.com/support/documentation/7.0/doc/prolog.html`

mrsToTal(MRS)
$R_{mrs} \leftarrow$ the list of relations in MRS
for each $r \in R_{mrs}$ do
  if r = the(x) or r = exists(x) then do $Q \leftarrow Q \cup \exists x$
  else if r = forall(x) then do $Q \leftarrow Q \cup \forall x$
  else if $r = rel(e_i, x_1, \ldots, x_n)$ then do
    $R_{tal} \leftarrow R_{tal} \cup$ [now] $rel(x_1, \ldots, x_n)$
  else if $r = rel(x)$ and rel is a background theory sort
    add $sort(x) = rel$ to the symbol table
  else do $R_{tal} \leftarrow R_{tal} \cup r$
return $q_1 \ldots q_i.[r_1 \wedge \ldots \wedge r_j]$
    where $q_1, \ldots, q_i \in Q$ and $r_1, \ldots, r_j \in R_{tal}$

Figure 4: The translation algorithm.

is already entailed and therefore redundant. That the table supports B1 is entailed by the previous fact and the axioms defining *support*, so the system simply replies yes. In sentence 3 we demonstrate the evaluation of a simple yes/no-question, although in this case the answer is not known since nothing has been said about block B2 and NL1 makes no closed world assumptions. Sentence 4 adds B2 to the table and B3 on top of B1 so that when the question in sentence 5 is evaluated it follows, from the fact that B3 is on B1 and the fact that a block cannot be in two places at the same time, that the table can not support B3. A much more complex type of questions is what-questions. Sentence 6 poses a what-question about another defined relation, *clear*. The system processes the question, as described in detail in Section 4, and constructs an answer MRS from which a complex natural language reply is generated. The sentences in 7 and 8 are recognized as imperative and are parsed into TAL action occurrences that are added to the world history. Finally, the question in sentence 9 demonstrates that the new blocksworld state conforms to the changes specified for the performed actions in the background action theory listed in Figure 7.

4 Sentence Processing

The sentence processing can be described by an (incomplete) algorithm, shown in figure 4, that translates a feature structure representation of an MRS into a TAL formula. To illustrate the algorithm let us assume that a user enters the question in Figure 5a. The LKB component parses the string and the MRS in Figure 5b is extracted from the resulting feature structure. The MRS Trans component uses the algorithm to interpret the sentence as a TAL formula, starting by con-

a. "What blocks are on the table?"

b.
$$\begin{bmatrix} \text{predication} \\ \text{mode} \;\; \text{wh-ques} \\ \text{index} \;\; e_1 \\ \text{liszt} \left\langle \begin{bmatrix} \text{on} \\ \text{ARG}_1 \;\; e_1 \\ \text{ARG}_2 \;\; x_1 \\ \text{ARG}_3 \;\; x_2 \end{bmatrix}, \begin{bmatrix} \text{table} \\ \text{ARG}_1 \;\; x_2 \end{bmatrix}, \begin{bmatrix} \text{the} \\ \text{ARG}_1 \;\; x_2 \end{bmatrix}, \right. \\ \left. \begin{bmatrix} \text{block} \\ \text{ARG}_1 \;\; x_1 \end{bmatrix}, \begin{bmatrix} \text{exists} \\ \text{ARG}_1 \;\; x_1 \end{bmatrix}, \begin{bmatrix} \text{answer} \\ \text{ARG}_1 \;\; x_1 \end{bmatrix} \right\rangle \end{bmatrix}$$

c. `(on e1 x1 x2) (table x2) (the x2)`
`(block x1) (exists x1) (answer x1)`

d.
```
(exists (x2 x1)
  (and (on e1 x1 x2) (table x2)
       (block x1) (answer x1)))
```

e.
```
(exists (x2 x1)
  (and (time now (on x1 x2))
       (table x2) (block x1)
       (answer x1)))
```

f.
```
(exists (x2 x1)
  (and (time now (on x1 x2))
       (answer x1)))
```

$sort(x1) = block, sort(x2) = table$

Figure 5: Input processing for a question.

a. `(answer b2) (answer b1)`

b.
```
(and
 (exists (x2) (time now (on b2 x2)))
 (exists (x4) (time now (on b1 x4))))
```

c. `(and e1 e2 e3)`
`(exists x2) (on e2 b2 x2)`
`(exists x4) (on e3 b1 x4)`

d. `(and e1 e2 e3) (the x2) (on e2 x1 x2)`
`(b2 x1) (table x2) (the x4)`
`(on e3 x3 x4) (b1 x3) (table x4)`

e.
$$\begin{bmatrix} \text{predication} \\ \text{mode} \;\; \text{prop} \\ \text{index} \;\; e_1 \\ \text{liszt} \left\langle \begin{bmatrix} \text{and} \\ \text{ARG}_1 \;\; e_1 \\ \text{ARG}_2 \;\; e_2 \\ \text{ARG}_3 \;\; e_3 \end{bmatrix}, \begin{bmatrix} \text{on} \\ \text{ARG}_1 \;\; e_2 \\ \text{ARG}_2 \;\; x_1 \\ \text{ARG}_3 \;\; x_2 \end{bmatrix}, \begin{bmatrix} \text{table} \\ \text{ARG}_1 \;\; x_2 \end{bmatrix}, \right. \\ \left. \begin{bmatrix} \text{the} \\ \text{ARG}_1 \;\; x_2 \end{bmatrix}, \begin{bmatrix} \text{b2} \\ \text{ARG}_1 \;\; x_1 \end{bmatrix}, \begin{bmatrix} \text{on} \\ \text{ARG}_1 \;\; e_3 \\ \text{ARG}_2 \;\; x_3 \\ \text{ARG}_3 \;\; x_4 \end{bmatrix}, \right. \\ \left. \begin{bmatrix} \text{table} \\ \text{ARG}_1 \;\; x_4 \end{bmatrix}, \begin{bmatrix} \text{the} \\ \text{ARG}_1 \;\; x_4 \end{bmatrix}, \begin{bmatrix} \text{b1} \\ \text{ARG}_1 \;\; x_3 \end{bmatrix} \right\rangle \end{bmatrix}$$

f. "B2 is on the table and B1 is on the table."

Figure 6: Answer processing and generation.

```
(domain block (b1 b2 b3))
(domain table (table1))
(domain surface (block table))
(fluent (holding block) boolean)
(fluent (on block surface) boolean)
(action (pickup block))
(action (putdown block surface))
(variable t time)
(variable block1 block)
(variable surface1 surface)
(variable surface2 surface)
(per (forall (t block1 surface1)
  (and (per t (holding block1))
       (per t (on block1 surface1)))))
(dep (forall (t block1 surface1 surface2)
  (-> (and (time t (on block1 surface1))
           (not (= surface1 surface2)))
   (time t (not (on block1 surface2))))))
(acs t1 t2 (pickup block1)
  (and (r (oc t1 t2) (holding block1))
    (forall (surface1)
      (r (oc t1 t2)
         (not (on block1 surface1))))))
(acs t1 t2 (putdown block1 surface1)
  (r (oc t1 t2)
     (and (not (holding block1))
          (on block1 surface1))))
(obs (forall (block1)
       (time 0 (not (holding block1)))))

(occ 0 1 (pickup b3))
(occ 1 2 (putdown b3 table1))
```

Figure 7: TAL representation of the blocksworld.

verting the MRS structure to a relation list, shown in Figure 5c. Our simple treatment of quantifiers assigns the scope of the entire formula while ordering each quantifier as they appear in the relation list and treating `the` as an existential quantifier, as in Figure 5d. In Figure 5e, the TAL time operator has been applied to create a temporal formula replacing the event variable `e1` with the designated constant `now` that is updated during the dialogue. Finally, to take advantage of the fact that TAL is an order-sorted logic, one-place relations that are identical to sorts defined in the current background theory are compiled into a symbol table and removed from the formula in Figure 5f.

At this point we have arrived at the TAL formula representing the user's question. Let us further assume that we are using the TAL blocksworld formalization shown in Figure 7, and that the current state of the dialogue interaction has produced the narrative history represented by the two final `occ` statements in the figure. These components together form a reasoning problem and any answer to the original question is an instantiation of the variable(s) selected by the special answer predicate in the question formula that satisfies the

blocksworld specification together with the current world history.

Whatever method chosen, to solve the specific reasoning problem, will produce one or more answer formulas as show in Figure 6a. The answers are used to instantiate the question formula from Figure 5f to form the TAL representation in Figure 6b, representing the answers to the question. Again the MRS Trans module applies a number of processing steps, this time to construct an MRS structure representing the answers. First, the TAL time operator is translated into event variables and the quantifiers flattened to relations in Figure 6c. In Figure 6d, the references to the different blocks are made explicit in the relation list and the existential quantifier relations are translated to `the` relations, assuming the answer recipient knows what instance is meant. Finally, the MRS feature structure in Figure 6e is built and passed to the LKB module which uses the HPSG grammar to generate a natural language sentence corresponding to it. This sentence, displayed in Figure 6f, represents the natural language answer to the natural language question posed in Figure 5a.

5 Discussion

Natural language understanding was among the first research topics of artificial intelligence and has continued to be of great importance. As a consequence a large number of natural language understanding and dialogue systems have been built. Even though our system does not contribute any new techniques, it is a novel combination of existing techniques that form an environment in which a variety of linguistic and knowledge representation problems can be attacked.

Both the HPSG grammar and the TAL background knowledge are declarative knowledge sources that can be updated, incrementally and iteratively extended, or adapted to other use scenarios, independent of each other and most of the implementation components. They are also very expressive, contributing to the generality of the approach and to the goal of achieving representational adequacy. Equally important to expressiveness are issues of efficiency. Through a flexible compilation scheme for automated reasoning the tradeoff between expressiveness and efficiency is not set in stone but can be adapted to the task at hand. Different compilation steps can be implemented from different subsets of TAL to different automated reasoners and the choice of which compilation step to use is postponed.

While we have defended our design decisions we also acknowledge that they do give rise to some disadvantages. Relying on deep parsing might result in an instance of what is often called the "brittleness" problem where sentences slightly outside the competence of the grammar is totally incomprehensible and any grammatical error has similar results. We agree to some extent with this objection but note that there is interesting work on a Robust Minimal Recursion Semantics (Copestake, 2003) that would help integrate deep and shallow parsing. However, it should be pointed out that certain problems can never be solved using shallow methods and the mixing of deep and shallow methods can only obscure their solutions. Although generality was listed as an advantage, we also include it as a system drawback and possible point of attack by critics. A more general system will always be worse at any given task than a less general system that is more specialized toward that task. This rule weakens as the complexity of the task grows, but holds for most problems dealt with in current AI research.

5.1 Evaluation

Ultimately deciding on the viability of a framework such as NL1 should depend in some way on evaluation, but we have yet to perform any structured evaluation attempts and it is far from obvious how they would be implemented. The emphasis is not on the immediate improvement in the measurable efficiency of some real-world task like natural language translation, or the maximum coverage of a large number of question answering tasks, but instead on the investigation and exploration of a potentially fruitful marriage between natural language and knowledge representation technologies. At the same time, it is clear that there are obvious improvements that could be made to the current NL1 system, and that one would like to formulate some measure that made what is intuitively obvious, objectively evident. To this end we propose evaluations through a problem collection.

Even a very simple system can answer an infinite number of questions correctly if all that is varied is some trivial question property such as the name of the blocks in a blocksworld. It is not sufficient to count the number of sentences that are answered without further classifying them

according to some complexity dimension, which is easier said than done. But even without this classification, one carefully selected sentence, illustrative of some complexity of language or reasoning, can still make or break a system. We propose that one creates and maintains an online collection of such examples, similar to the common-sense reasoning problem page (Morgenstern, 2005), the logically reasoning agents problem page (Thielscher, 2005), or the logic modelling workshop (Sandewall, 2005). New examples, illustrating new difficulties, would be submitted by different researchers and added to the problem page (with moderation). Evaluating different versions of the same system would then simply be accomplished by noting that the new version solves some additional question (ignoring, among others, issues of efficiency). Comparisons between systems would be entirely possible if one system subsumes the sentences correctly handled by the other system and possible with subjective results if the systems had a partial overlap.

5.2 Limitations and Future Work

We think this project has only scratched the surface of what is possible to accomplish using these techniques and that the proposed architecture has great potential. This section will point out the most important current limitations as well as our plans for continued development.

An obvious improvement is an extension of the HPSG grammar coverage to make the dialogue more varied and robust. The grammar does not at present even cover all the language constructs described in the book that it is based on (Sag et al., 2003) and there are certainly other HPSG grammar work that can be adapted to our system to further complement and extend coverage. An interesting experiment would be a coupling to the English resource grammar (Copestake and Flickinger, 2000).

Another extension that would put our approach to the test is a method of dealing with different forms of reference. We envision that, in addition to the domain background theory, the knowledge representation module will contain a model of the ongoing dialogue expressed using the same temporal action logic. Reference resolution would then be the solution of additional reasoning problems where objects that simultaneously fulfilled declarative constraints from dialogue factors, such as recency, and background knowledge, would be retrieved.

Time is a central concept in the system, yet at present this is not taken advantage of. Questions are all in present tense, even though everything is set up so as to support one talking to the system about past actions and time.

The current implementation recognizes commands of action and executes them, but such commands are restricted to simple atomic operations and the system can not by itself plan a sequence of actions in response to a user request. As part of another project, we are working on extending TAL to incorporate composite actions and action sequences or plans. Such work would fit naturally in the framework we have described here and would enable a seamless transition between requesting simple actions and requesting complex actions, possibly requiring the use of deductive planning, without extending the system architecture with a special purpose planner.

The compilation approach to automated TAL reasoning is inherently suitable for experimentation. While we have already performed work in this direction, developing several different compilations to first-order logic and another to logic programs, we do not expect to run out of ideas in this area. An especially interesting one is the use of deduction system alternatives to resolution, such as natural deduction, that might be more suitable for the kinds of inferences needed in the logicist approach to natural language understanding.

6 Related Work

An early and very impressive demonstration of natural language understanding was the SHRDLU system (Winograd, 1971). NL1 improves upon SHRDLU by using modern HPSG grammars instead of CFG grammars and declarative instead of procedural knowledge representation, but still falls short of the complexity of correctly executed dialogues. Though we are confident that our more general system architecture will catch up in the long run.

More recent work was carried out in the CLARE project (Alshawi et al., 1992) to provide natural language question answering interfaces for databases. The Core Language Engine parses questions into a quasi-logical form, aptly called QLF, that is interpreted and reasoned about in the context of background knowledge. In addition to

the choice of parsing environment and intermediate form, two differences from NL1 are that we explicitly avoid committing to a specific reasoning paradigm while CLARE is based on the logic programming paradigm, and that the scale of the CLARE project is simply vastly larger than ours.

The idea of using other theorem proving techniques than logic programming to aid natural language understanding has also been explored previously. The work in (Blackburn et al., 1998) uses Dynamic Logic as the semantical representation and a translation to a fragment of first-order logic together with the Bliksem theorem prover as the reasoning mechanism. The feasibility of the set-up is demonstrated by using it to resolve discourse ambiguities. Our approach is similar in the application of an automated theorem prover after a translation step, but our background knowledge is encoded in Temporal Action Logic, which endows the system with the power to perform actions and reason about their effects.

Other systems, such as the architecture described in (Allen et al., 2001), deal with dialogues in realistic scenarios where human users want to interact with the system as fluently as possible to accomplish a task. Such efforts strive for *human-like* behaviour while we consider the ultimate goal to be *human-level*, but possibly very artificial, behaviour and hypothesize that many issues in human dialogues might well be ignored. Our interest lies not in modelling dialogue phenomena, but in using knowledge representation techniques for natural language understanding in a system with a dialogue interface.

Acknowledgements

We would like to thank Lars Ahrenberg for guidance and helpful discussions.

References

James Allen, George Ferguson, and Amanda Stent. 2001. An architecture for more realistic conversational systems. In *Proceedings of the 6th International Conference on Intelligent User Interfaces IUI'01*, pages 1–8. ACM Press.

Hiyan Alshawi, David Carter, Richard Crouch, Steve Pulman, Manny Rayner, and Arnold Smith. 1992. CLARE: A contextual reasoning and cooperative response framework for the core language engine. Technical Report CRC-028, SRI International.

Patrick Blackburn, Johan Bos, Michael Kohlhase, and Hans de Nivelle. 1998. Automated theorem proving for natural language understanding. In *Problem-solving Methodologies with Automated Deduction (Workshop at CADE-15)*.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage english grammar using HPSG. In *Proceedings of the Second Conference on Language Resources and Evaluation LREC-2000*.

Ann Copestake. 2003. Report on the design of RMRS. Technical Report D1.1a, University of Cambridge.

Patrick Doherty and Witold Łukaszewicz. 1994. Circumscribing features and fluents. In *Proceedings of the 1st International Conference on Temporal Logic ICTL'94*, volume 827 of *Lecture Notes in AI*, pages 82–100. Springer.

Patrick Doherty, Joakim Gustafsson, Lars Karlsson, and Jonas Kvarnström. 1998. Temporal action logics (TAL): Language specification and tutorial. *Linköping Electronic Articles in Computer and Information Science*, 3(15).

Patrick Doherty. 1994. Reasoning about action and change using occlusion. In *Proceedings of the Eleventh European Conference on Artificial Intelligence ECAI'94*, pages 401–405. John Wiley and Sons.

Vladimir Lifschitz, 1994. *Circumscription*, volume 3 of *Handbook of Logic in Artificial Intelligence and Logic Programming*, chapter 6, pages 298–352. Oxford University Press.

Leora Morgenstern. 2005. The common-sense reasoning problem page. `http://www-formal.stanford.edu/leora/commonsense/`. Visited February 2006.

Ivan A. Sag, Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: A Formal Introduction (Second Edition)*. CSLI Publications.

Erik Sandewall. 1994. *Features and Fluents: The Representation of Knowledge about Dynamical Systems*, volume 1. Oxford University Press.

Erik Sandewall. 2005. Logic modelling workshop. `http://www.ida.liu.se/ext/etai/lmw/`. Visited December 2005.

Mark E. Stickel. 2005. SNARK - SRI's new automated reasoning kit. `http://www.ai.sri.com/~stickel/snark.html`. Visited December 2005.

Michael Thielscher. 2005. Logically reasoning agents problem page. `http://www.cl.inf.tu-dresden.de/~mit/LRAPP/`. Visited December 2005.

Terry Winograd. 1971. Procedures as a representation for data in a computer program for understanding natural language. Technical Report 235, MIT Artificial Intelligence Laboratory.