Automatic Translation of Natural Language System Specifications into Temporal Logic

Rani Nelken[1] and Nissim Francez[2]

[1] Tel-Aviv University, Tel-Aviv 69978, Israel
[2] Computer Science Department, The Technion, Haifa 32000, Israel

Abstract. This paper presents a method for automatically translating natural language specifications into temporal logic. Using this method, users may express complex specifications in relatively free natural language, allowing multi-sentence specifications, the use of pronouns instead of repeating the description of previously mentioned objects and complex temporal relations. These specifications are translated into temporal logic, while ensuring the correctness of the translation. This approach overcomes a well-known obstacle of applying model-checking industrially. In contrast to prior attempts, the translation is linguistically based on a modern formalism for discourse representation. An implementation of this translation method is presented in one of the modern computational linguistics systems.

1 Introduction

Temporal logic based verification using model-checking has proved to be a useful and effective method for verifying finite-state concurrent systems. A recognized problem in industrially applying this verification technique is making the specification formalism more intuitive and easy to use. In practice, designers of computerized systems either lack sufficient training in formal methods, or find the specification formalism un-intuitive and inconvenient. Thus, the formulation of specifications often becomes a two stage process:

1. Specifications are written in natural language (NL).
2. These specifications are manually translated into temporal logic (TL). This is done according to the intuitive understanding of the translator, who has to contend with the imprecision and ambiguity of NL and with the subtle interpretation of TL formulae.

The frequent occurrence of imprecision in this translation is discussed in [4], who propose an automatic translation tool, based on a direct mapping between TL formulae and NL constructs. This method allows designers to express specifications in NL, but severely restricts the source language according to the structure of TL. Thus, the full power attainable by computational linguistic methods is not attained. It is not clear whether their system is grounded on any sound linguistic theory.

In [5], the translation of NL specifications of logic programs into Prolog is discussed. Input specifications in 'controlled' NL are automatically translated into Prolog clauses through an intermediary representation in a linguistic theory called *Discourse Representation Theory* (DRT) [8]. These clauses serve as a Prolog knowledge base, which may be executed and queried, and also paraphrased back in NL.

In this paper, we describe a novel translation method, which also uses DRT as an intermediate representation level. We largely enhance the NL constructs which may be translated, abstracting away from the target TL, and allowing the use of much freer, more natural language. Instead of isolated formula-like sentences as in [4], we deal with sequences of inter-connected specifications, which we term *specification discourses* (SDs). Our method allows a treatment of complex NL constructs such as:

- NP anaphora - the use of pronouns instead of repeating a full description of an object.
- Complex temporal expressions, which are crucial for TL verification. Expressing such properties is one of the major difficulties, which hinders designers from using TL directly. This is a major advancement over the treatment of temporal expressions in [5], who use the temporal capabilities of DRT in a very limited way. They distinguish between events and states and relate them with the utterance time, but do not explore temporal relations between events or states in discourse.

We further introduce a correctness criterion for the (second stage of the) translation, and formally prove that the translation method fulfills this criterion. Such a criterion and proof were lacking.

An implementation of this method, in the form of an interactive program is described. It accepts SDs, parses them, constructing a representation in DRT, called a *Discourse Representation Structure* (DRS), and then translates the DRS into a TL formula. The generated formulae can subsequently serve as input to a model-checker.

1.1 A Running Example

To illustrate the translation method, consider an allocator A that allocates a single resource to m different customer processes $C_1, C_2, \ldots, C_m$[3]. The communication between each C_i and A is done by a pair of shared boolean variables r_i (request) and g_i (grant). Following are statements of properties, that a designer may wish to formally specify and verify about the operation of this system. Each specification is given in both NL and our target temporal logic, ACTL[4,5] [6], a subset of Computational Tree Logic (CTL) [2]. These specifications will illustrate the problems encountered in NL translation as described in Sec. 1.2.

[3] This example is based on one in [14].

[4] The target formalism of [4] is called ACTL too. Nevertheless, it is a different formalism.

[5] We allow the use of the following additional operators:

(1) a. One cycle After r_i is activated, g_i should be asserted. r_i is deactivated one to six cycles later. Afterwards, it should be deasserted. (Response to requests)
$\mathbf{AG}\ [rise(r_i) \rightarrow \mathbf{AX}\ [g_i \& \mathbf{ABF}_{1..6}\ (fall(r_i) \& \mathbf{A}\ [\neg fall(g_i) \mathbf{U}\ fall(g_i)])]]$

b. If r_i is active and g_i is asserted one cycle later, then eventually r_i will be inactive. (Conformance with the protocol)
$\mathbf{AG}\ [r_i \rightarrow \mathbf{AX}\ (g_i \rightarrow \mathbf{AF}\ \neg r_i)]$

c. Whenever r_i and g_i are inactive, they remain inactive, until r_i is activated and g_i remains inactive. (Conformance with the protocol)
$\mathbf{AG}\ [\neg r_i \& \neg g_i \rightarrow \mathbf{A}\ [\neg r_i \& \neg g_i \mathbf{U}\ (rise(r_i) \& \neg g_i)]]$

d. Once r_i is activated, if g_j is asserted, then g_i will be activated before it is asserted again. (1-Bounded overtaking)
$\mathbf{AG}\ [rise(r_i) \rightarrow \mathbf{AG}\ [g_j \rightarrow \mathbf{A}\ [\neg rise(g_j) \mathbf{U}\ rise(g_i)]]]$

1.2 Problems in the Analysis of NL Specifications

Following are characteristics of SDs, which should be taken into account when translating NL. They stem from NL in general, the specific structure of SDs and the gap between the source and target languages.

Ambiguity and imprecision are exhibited by SDs and must be resolved in the translation into TL.
Example: in sentence (1c) it is unclear which two clauses are conjoined by the conjunction 'and'. The second conjunct is 'g_i remains inactive', but the first is either: 'r_i is activated' or 'they remain inactive until r_i is activated'.

Inter-sentential links The natural way of expressing specifications involves multi-sentence discourses. These are not just lists of isolated sentences but rather coherent objects.
Example: the second sentence of example (1a) is crucially dependent on its predecessor, i.e. the phrase 'one to six cycles later' can be interpreted only by reference to the first sentence.

The temporal structure of specifications Specifications exhibit a unique temporal structure, causing SDs to often be expressed in a generic present or future tense:
- *Disconnection from the present:* SDs are often expressed in a timeless language.
Example: Even though the tense of the verbs in sentence (1b) is simple present, it does not refer to the present moment per se (e.g. the time of writing the sentence).

- $rise(p)$ and $fall(p)$ for $p \in AP$, where AP is the set of atomic propositions - to represent a change from $\neg p$ to p and vice versa.
- $\mathbf{AX}_i\ f \equiv \underbrace{\mathbf{AX}\ (\mathbf{AX}\ (\ldots(\mathbf{AX}\ f)))}_{\times i}$
- $\mathbf{ABF}_{i..j}\ f \equiv \mathbf{AX}_i\ (f \vee \underbrace{\mathbf{AX}\ (f \vee \ldots \mathbf{AX}\ (f \vee \mathbf{AX}\ f))}_{\times(j-i)})$

- *Quantification over events:* Specifications are usually concerned with recurring events in the course of a system's operation. Consequently, SDs contain either explicit or implicit quantification over events and time periods.
  Example: Sentence (1b) is understood as referring to *each* time r_i is active.

Non determinism Our target formalism is ACTL, a branching-time TL. In practice, designers tend to specify the behavior of systems in linear-time terms. When translating SDs into a branching-time TL, this linearity has to be introduced into the formalism. A simpler solution to this problem would be choosing a linear time TL, e.g. LTL, as our target formalism. The choice of ACTL in this paper is motivated by its widespread and successful use (e.g. [15, 1]).
Example: the meaning of sentence (1d) is clear in a linear-time model. However, it may have several branching-time TL interpretations, e.g. assume a branching structure at the root of which r_i is asserted, but where g_j is asserted along only one path from the root. Along which paths should g_i be activated ?

We solve these problems through the use of DRT, presented in Sec. 2, and a restriction on the generated formulae, explained in Sec. 4.2.

2 Discourse Representation Theory

DRT [8, 9, 10] is a linguistic theory of the semantic content of general NL, which studies discourses. It combines a static logical view of meaning with a dynamic cognitive view. DRSs are defined as formulae of a formal language $\mathcal{L}$ consisting of:

1. an infinite set R of typed *markers*: $x\,y\,z$ for signals, $s, s_1, s_2, \ldots s_n$ for states. $e, e_1, e_2, \ldots e_n$ for events and i, j for integers.
2. for each $n \in \mathbb{N}$ an infinite set P^n of n-place predicates. $V = \bigcup_n P^n$
3. identity

Definition 1. 1. A DRS $K = \langle U_K, Con_K \rangle$, where $U_K \subset R$ is finite and Con_K is a finite set of DRS-conditions.
2. Let K, K_1, K_2 be DRSs, $r_1, r_2, \ldots r_n \in R$ and $P \in P^n$. A DRS-condition may be one of: $r_1 = r_2$, $P(r_1, r_2, \ldots, r_n)$,$\neg K$, $K_1 \Rightarrow K_2$, $K_1 \vee K_2 \vee \ldots \vee K_n$

DRSs are interpreted as partial models. A DRS is verified by a model[6] M, iff it may be embedded in it as follows:

Definition 2. A model for $\mathcal{L}$ is $M = \langle U_M, \mathcal{EV}, \mathbb{N}, Pred_M \rangle$ where:

1. U_M is a non-empty set.

[6] In Sec. 4.2, we make some changes in the following definition of a model.

2. $\mathcal{EV}$ is an event structure (see [10]).
3. $\mathbb{N}$ is the set of natural numbers.
4. $Pred_M$ maps each n-place predicate of $\mathcal{L}$ onto an n-place relation over U_M.

Definition 3. Let K be a DRS, γ a DRS-condition, and let f be an **embedding** from some subset of V into M, i.e. $f : R' \rightarrow U_M$, where $R' \subseteq R$.

1. f verifies the DRS K in M ($M \models_f K$) iff f verifies each $\gamma \in Con_K$ in M.
2. f verifies the condition γ in M ($M \models_f \gamma$) iff
   (a) γ is of the form $r_1 = r_2$, $r_1, r_2 \in Dom(f)$ and $f(r_1) = f(r_2)$.
   (b) γ is of the form $P(r_1, r_2, \ldots, r_n)$, $r_1, r_2, \ldots r_n \in Dom(f)$ and $\langle f(r_1), f(r_2), \ldots, f(r_n)\rangle \in Pred_M(P)$
   (c) γ is of the form $\neg K'$ and there is no embedding $g : R \rightarrow U_M$, which extends f, s.t. $Dom(g) = Dom(f) \cup U_{K'}$ and $M \models_g K'$
   (d) γ is of the form $K_1 \Rightarrow K_2$ and for every embedding g s.t. $Dom(g) = Dom(f) \cup U_{K_1}$ and $M \models_g K_1$, there is an extension h of g s.t. $Dom(h) = Dom(g) \cup U_{K_2}$ and $M \models_h K_2$.
   (e) γ is of the form $K_1 \vee K_2 \vee \ldots \vee K_n$ and for some i $1 \leq i \leq n$ $M \models_f K_i$.

DRSs are constructed from NL discourses by the *DRS-construction algorithm* (Fig. 1), which processes sentences one by one, incrementally updating a DRS according to a set of *DRS-construction rules*[7]. The algorithm models the way in which a human listener processes a discourse, understanding sentences one at a time. DRT provides an analysis of the 'glue' that holds a discourse together, most prominently, it is able to resolve the meaning of anaphoric pronouns in discourse. Full DRT also provides a thorough analysis of temporal relations within discourse [7, 17, 10, 16]. While this analysis is not described in this paper, it is illustrated through the DRS-construction for example (1a).

```
Input: Specification Discourse D = ⟨S_1, S_2, ..., S_n⟩
ContextDRS ← EmptyDRS
i ← 0
repeat
        DRS_i ← parse(S_i)
        ContextDRS ← UpdateContext(ContextDRS,DRS_i)
        i ← i + 1
until i = n
Output: ContextDRS
```

Fig. 1. DRS-construction algorithm

[7] These rules, i.e. the details of *UpdateContext* are given in [10] and are not presented here for lack of space.

3 DRS-Construction for NL Specifications

We illustrate the construction of DRSs for SDs through the stepwise construction of a DRS for example (1a), repeated here as (2).

(2) a. One cycle after r_i is activated, g_i should be asserted.
b. r_i is deactivated one to six cycles later.
c. Afterwards, it should be deasserted.

DRSs are depicted using a graphical box-notation. The top of the box corresponds to the universe of the DRS, and the rest to the DRS-conditions. The DRS-construction algorithm constructs the DRS[8] shown in Fig. 2.

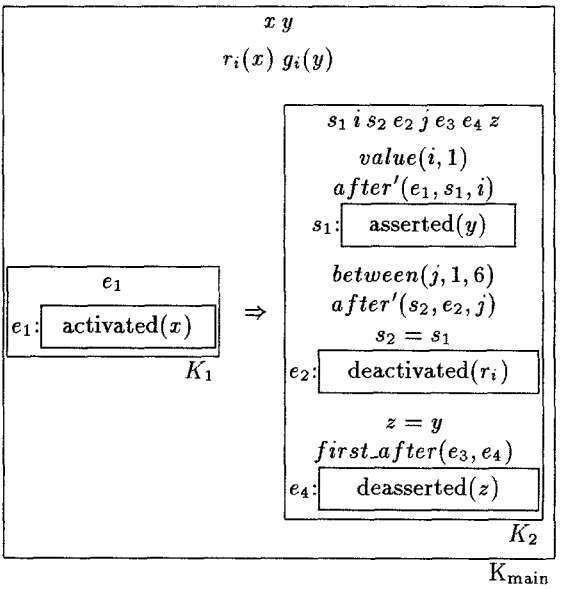

Fig. 2. Example DRS

a. Processing sentence (2a) introduces the discourse markers x, y into $U_{K_{main}}$, the first two DRS-conditions naming x and y r_i and g_i and the implication condition $K_1 \Rightarrow K_2$, excluding the part of K_2 following the condition: s_1:[...]. The embedding conditions of an implication $K_1 \Rightarrow K_2$ determine that it induces universal quantification over the discourse referents of U_{K_1} and existential over those of U_{K_2}, giving an interpretation that for each event e_1, in which x is activated, there is a state s_1, in which y is asserted,

[8] For reference purposes DRSs are labeled. The labeling is part of the meta-language used to discuss DRSs.

which follows the occurrence of e_1 by one time cycle. The introduction of the implication condition is due to the implicit quantification inherent in (2a).

b. Parsing sentence (2b) produces a new DRS. This DRS is combined with the DRS for the previous sentence, adding into K_2 the discourse markers s_2, e_2, j and the DRS-conditions $between(\ldots)$, $after'(s_2, e_2, j)$, e_2:[$\cdots$]. When combining the DRSs, the 'anonymous' eventuality marker s_2 is identified with s_1, thus determining that 'later' here means 'after the assertion of g_i'.

c. Parsing sentence (2c) generates an additional DRS, which is combined with the compound DRS of sentences (2a) and (2b) to form the completed DRS of Fig. 2. The treatment of 'afterwards' is identical to that of 'later'. This sentence also contains the pronoun 'it', which causes the introduction of the marker z, which is later identified with y by a similar technique.

We introduce a set of restrictions on the structure of DRSs generated for SDs, not described here for lack of space. A DRS thus restricted is called a *Specification DRS* (SPDRS), and the set of SPDRSs, SPDRT. These restrictions are a result of the subset of NL accepted by the translation method and the DRS-construction rules.

4 Translating DRSs into ACTL

4.1 The Translation Method

We sketch the translation procedure $trans : SPDRT \Rightarrow ACTL$. We illustrate its operation on the DRS of Fig. 2:

Translating K_{main}: $K_1 \Rightarrow K_2$ generates a formula starting in **AG** . This reflects the universal quantification on the elements of U_{K_1} conferred by the embedding[9].

Translating $K_1 \Rightarrow K_2$: The temporal relation[10] $after'(e_1, s_1, i)$ generates an operator $\mathbf{AX}_i$. Let f_{K_2} be the translation of the remaining conditions of K_2. The translation is $\mathbf{AG}\,[rise(r_i) \rightarrow \mathbf{AX}\,(g_i \& f_{K_2})]$, where $rise(r_i)$ is the translation of the *event* e_1, and g_i that of the *state* s_1.

Translating the remaining conditions of K_2: The translation is driven by the $after'$ and $first\_after$ conditions. The translation of the $first\_after$ generates the formula $f_2 = fall(r_i) \& \mathbf{A}\,[\neg fall(g_i) \mathbf{U}\, fall(g_i)]$. The translation of the first one generates the formula $g_i \& \mathbf{ABF}_{1..6}\,(fall(r_i) \& f_2)$.

The resulting translation is therefore the following, which is equivalent to (1a):

$$\mathbf{AG}\,[rise(r_i) \rightarrow \mathbf{AX}\,(g_i \& \mathbf{ABF}_{1..6}\,(fall(r_i) \& fall(r_i) \& \mathbf{A}\,[\neg fall(g_i) \mathbf{U}\, fall(g_i)]))]$$

[9] In general, the main DRS may contain several implications, in which case its translation is the conjunction of such formulae.

[10] Different temporal relations generate different operators.

4.2 Correctness

The translation method consists of two major transitions: from NL to DRT, and from DRT to TL. We define and prove a correctness criterion only for the second stage of the translation. No correctness criterion can be defined for the first stage of the translation. Given a NL SD and DRS, such a criterion would determine whether the DRS correctly represents the meaning of the NL specification. This criterion would require an alternative formal interpretation of the NL SD, the validity of which should also be somehow defined and proved. Therefore, we cannot hope for a mathematical correctness criterion for the DRS-construction algorithm. We can only check whether the constructed DRSs conform to our linguistic knowledge about the meaning of the NL SDs they are meant to represent. Thus, we benefit from the wealth of linguistic research dedicated for the purpose of providing a semantic analysis of NL.

Kripke Structure Induced DRT Models In order to ensure the correctness of the translation from DRT to TL, we link the models used for the interpretation of both theories. Through this linking, we solve the dichotomy between the linear-time interpretation of SDs and DRSs, and the branching-time interpretation of ACTL formulae.

Definition 4. Given a path $\pi = s_0, s_1, \ldots$ in a Kripke structure P, a *single path structure* $P(\pi)$ is a tuple $\langle S_\pi, s'_0, R_\pi, L_\pi \rangle$ such that

- $S_\pi = \{s'_0, s'_1, \ldots\}$ is an infinite set of (pairwise distinct) states.
- $R_\pi = \{(s'_i, s'_{i+1}) | i \geq 0\}$
- $\forall i \geq 0\ L_\pi(s'_i) = L(s_i)$

We expand the definition of satisfaction of CTL* formulae (defined only for structures having a finite set of states) to also include satisfaction of formulae by single path structures.

Definition 5. For any Kripke structure P, its induced DRT model M_P is constructed as follows:

- For each $p \in AP$,a binary signal of M_P
- For each each path $\pi \in \Pi_P$, the set of paths of P, a DRT path-model M_π as follows: for each state of $P(\pi)$, a state of M_π, and for each pair of consecutive states, an event.

Verification for DRT path models is as in Definition 3. We define $M_P \models K$ iff each of the induced path-models $M_\pi \models K$.

Based on this construction, we present the following correctness criterion:

Definition 6. An ACTL formula f is a correct translation of a DRS K iff for every Kripke structure P and its induced DRT model M_P: $M_P \models K \iff P \models f$

Theorem 7 shows that the translation method conforms to this correctness criterion. The proof of this theorem is based on the reduction of correctness to single paths.

Theorem 7. *Let K be an SPDRS, and $f = trans(K)$. f is a correct translation of K into ACTL.*

Reduction of Correctness to Single Paths In [6] three linearity properties for branching time temporal logics are defined: *strong linearity*, *sub-linearity* and *equi-linearity*. We take advantage of the strongest of these properties, strong-linearity, by restricting the formulae generated by the translation to the subset of strong-linear formulae.

Definition 8. [6] A formula $f \in$ ACTL is strong-linear, iff there is an ω-regular language $\mathcal{L}_f$, such that for every Kripke structure P, $P \models f \iff \mathcal{L}(P) \subseteq \mathcal{L}_f$

Lemma 9. *If K is an SPDRS, then $trans(K)$ is a strong-linear ACTL formula.*

Lemma 10 asserts that for a strong-linear formulae f, the satisfaction of f by a Kripke structure P depends only on the satisfaction of f by each individual path of P, regardless of the way they are interleaved.

Lemma 10. *If $f \in$ ACTL is strong-linear, then for every Kripke structure P, $[\forall \pi \in \Pi_P, P(\pi) \models f] \iff P \models f$*

Strong-linearity allows us to reduce the correctness of the translation of an SPDRS by a Kripke structure to its correctness relative to single path structures, as in the following lemma illustrated by Fig. 3.

Lemma 11. *Let f be a strong-linear ACTL formula and K an SPDRS. If for every Kripke structure P and its induced DRT model M_P: $\forall \pi \in \Pi_P[M_\pi \models K \iff P(\pi) \models f]$ where M_π is the path model associated with the single path structure $P(\pi)$, then f is a correct translation of K.*

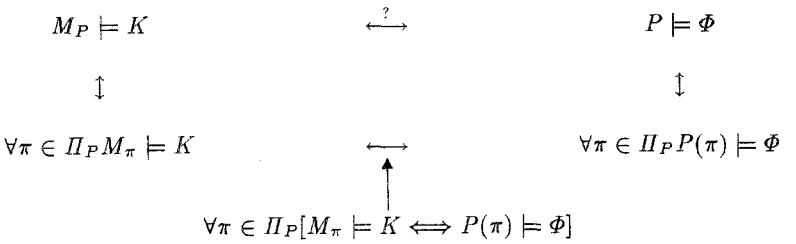

Fig. 3. Reduction of the correctness to correctness for paths

5 The Implementation

The translation method described above has been implemented as an interactive program **SpecTran 1.0**, which receives as input SDs, parses them, constructs DRSs and generates an ACTL formula from the DRS. The parser is written using the LexGram system [11, 12], which is based on a synthesis of ideas from *Lambek Categorial Grammar* [13] and *Head Driven Phrase structure Grammar* (HPSG) [18]. LexGram is written in a unification-based grammar formalism, CUF [3], and in Prolog.

In [11] a German grammar is implemented. The system parses sentences, constructing syntactic tree representations and semantic representations in the form of DRSs for them. In **SpecTran** the German grammar was replaced by an English one. The syntax part of this grammar was written by Esther König. **SpecTran**, parses input SDs sentence after sentence, processing each new sentence and updating a single DRS. In cases of ambiguity, the parser produces all the alternative parses of a sentence and their related DRSs.

SpecTran consults the user with respect to the resolution of the meaning of pronouns (e.g. 'it', 'they'), allowing a choice between a set of appropriate alternatives. A similar consultation is done with respect to the resolution of the meaning of words such as 'later'.

The completed DRS after the parsing of the full discourse, is passed in the form of a CUF data structure into the DRT to TL translation module. This module, written in CUF as well, implements the translation procedure described above. It accepts SPDRSs and translates them into strong-linear ACTL formulae.

6 Conclusion

In this paper, we have presented a translation method from NL specifications into TL, for the purpose of verification. Through the use of computational linguistic methods, we allow the expression of complex specifications in NL. While completely unrestricted NL is beyond the reach of current technology, and is arguably an undesirable medium for expressing specifications, we allow the use of relatively convenient language within certain restrictions. It is still necessary for designers to write specifications in precise and concrete language, but some tolerance is allowed in the use of flexible syntactic structure, multi-sentence discourse, pronominal anaphora and complex inner-sentential and inter-sentential temporal relations. By drawing on current linguistic research in the analysis of NL discourses, we enhance the applicability of an NL interface to a model-checker. It is our belief that the use of such an interface may facilitate the verification process in industrial practice, without harming the correctness of the verification. By introducing and proving a correctness criterion for the (second stage) of the translation and drawing on linguistic research for the first part of the translation, we are able to guarantee that the transformation from NL into TL does not introduce errors. Such a guarantee is lacking for the manual translation

process often exercised in practice, and from previous attempts of automatic translation.

Acknowledgments

The work of the second author was partially supported by a grant from the Israeli ministry of science "Programming languages induced computational linguistics", and by the fund for the promotion of research in the Technion. We also wish to thank the following people: Uwe Reyle, Esther König, Orna Grumberg, Ilan Beer and Shoham Ben-David.

References

1. E. M. Clarke, O. Grumberg, H. Hiraishi, S. Jha, D. E. Long, K. L. McMillan, and L. A. Ness. Verification of the futurebus+ cache coherence protocol. In L. Claesen, editor, *Proceedings of the Eleventh International Symposium on Computer Hardware Description Langugages and their Applications*, North Holland, 1993.
2. E.M. Clarke, E.A. Emerson, and A.P. Sistla. Expressibility results for linear time and branching time logic. *ACM transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
3. J. Dörre and M. Dorna. Cuf - a formalism for linguistic knowledge representation. In J. Dörre, editor, *Computational Aspects of Constraint - Based Linguistic Description*, volume I. ILLC/Department of Philosophy, University of Amsterdam, Amsterdam, 1993. DYANA 2 deliverable R.1.2.A.
4. A. Fantechi, S. Gnesi, G. Ristori, M. Carenini, M. Vanocchi, and P. Moreschini. Assisting requirement formalization by means of natural language translation. *Formal Methods in System Design*, 4:243–263, 1994.
5. N. E. Fuchs and R. Schwitter. Specifying logic programs in controlled natural language. In *Proceedings of the Workshop on Computational Logic for Natural Language Processing. A Joint COMPULOGNET/ELSNET/EAGLES Workshop*, Edinburgh, 1995.
6. O. Grumberg and R.P. Kurshan. How linear can branching-time be? In D. M. Gabbay and H. J. Ohlbach, editors, *First International Conference on Temporal Logic (ICTL'94). Lecture Notes in Artificial Intelligence 827*, pages 180–194, Bonn, Germany, 1994. Springer-Verlag.
7. E. W. Hinrichs. Temporale anaphora im englischen. Unpublished Statexamen Thesis. University of Tuebingen., 1981.
8. H. Kamp. A theory of truth and semantic representation. In T. Janssen J. Groenendijk and M. Stokhof, editors, *Formal Methods in the Study of Language*, pages 277–322. Mathematical Center, Amsterdam, 1981.
9. H. Kamp and U. Reyle. A calculus for first order discourse representation structures. Bericht Nr.16-1991 Arbeitspapier des Sonderforschungsbereich 340, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, 1991.
10. H. Kamp and U. Reyle. *From Discourse to Logic*, volume 42 of *Studies in Linguistics and Philosophy*. Kluwer Academic Publishers, 1993.
11. E. König. A study in grammar design. Arbeitspapier 54 des Sonderforschungsbereich 340. Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, 1994.

12. E. König. Lexgram - a practical categorial grammar formalism. In *Proceedings of the Workshop on Computational Logic for Natural Language Processing. A Joint COMPULOGNET/ELSNET/EAGLES Workshop*, Edinburgh, Scotland, 1995.
13. J. Lambek. The mathematics of sentence structure. *American mathematical monthly*, 65:154–170, 1958.
14. Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems.* Springer - Verlag, 1991.
15. K. L. Mcmillan. *Symbolic Model Checking: An Approach to the State Explosion Problem.* PhD thesis, Carnegie Mellon University, 1992.
16. R. Nelken and N. Francez. Splitting the reference time:temporal anaphora and quantification. In *Proceedings of the EACL '95 - The seventh meeting of the European Chapter of the Association for Computational Linguistics*, Dublin, Ireland, 1995. Also available as Technical Report LCL-94-10 of the Laboratory for Computational Linguistics, The Technion IIT.
17. B. Partee. Nominal and temporal anaphora. *Linguistics and Philosophy*, 7:243–286, 1984.
18. C. Pollard and I. A. Sag. *Head Driven Phrase Structure Grammar.* University of Chicago Press, Chicago, 1994.