

Abductive reasoning on temporal information

Sven Verdoolaege¹, Marc Denecker¹, Ness Schelkens²,
Danny De Schreye¹ and Frank Van Eynde²

¹Department of Computer Science and

²Centre for Computational Linguistics

K.U.Leuven

Abstract

Texts in natural language contain a lot of temporal information, both explicit and implicit. Verbs and temporal adjuncts carry most of the explicit information, but for a full understanding general world knowledge and default assumptions have to be taken into account. We will present a theory for describing the relation between, on the one hand, verbs, their tenses and adjuncts and, on the other, the eventualities and periods of time they represent and their relative temporal locations, while allowing interaction with general world knowledge.

The theory is formulated in an extension of first order logic and is a practical implementation of the concepts described in Van Eynde (2000a) and Schelkens et al. (2000). We will show how an abductive resolution procedure can be used on this representation to extract temporal information from texts.

1 Introduction

This article presents some work conducted in the framework of Linguaduct, a project on the temporal interpretation of Dutch texts by means of abductive reasoning.

A natural language text contains a lot of both explicit and implicit temporal information, mostly in verbs and adjuncts. The purpose of the theory presented here is to represent how this information is available in Dutch texts with the aim of allowing extraction of that information from particular texts.¹ The extracted information contains the temporal relations between the eventualities described in the text as well as relations to periods of time

¹In this paper, we only deal with sentences.

explicitly or implicitly described in the text. To arrive at this information some (limited) reasoning needs to be performed on the representation.

The theory is an adaptation of the theory described in Van Eynde (1998), Van Eynde (2000b) and Van Eynde (2000a), which integrates a DRT-style analysis (Kamp and Reyle 1993) of tense and aspect into HPSG (Pollard and Sag 1994). The adaptation concerns a reformulation of the typed feature structures in terms of first order logic. This facilitates the modeling of the interaction between linguistic knowledge and world knowledge.

The extraction of temporal information is performed by constructing a model of the logical theory, for which we use an existing abductive procedure. In case of ambiguities, several models exist and each can be generated.

We start off with some preliminaries about the adapted linguistic theory. We will then describe the knowledge representation language that we are going to use (essentially first order logic), followed by an extensive explanation of the representation of the linguistic theory in logic. We finish off with a short description of the reasoning procedure used and some examples of how it can be used on the theory to extract temporal information.

2 Conceptual framework

We assume familiarity with the following concepts regarding the semantics of tense, aspect and temporal modification. They correspond largely to what is found in Kamp and Reyle (1993).

An *eventuality* is either an event, a state or a process. We will not make use of these three kinds of eventualities, but only of their defining properties, i.e. *stative* vs. *dynamic* and *telic* vs. *atelic*. The *eventuality time* is the period of time that the eventuality takes place on. Eventualities are introduced by verbs, but not all verbs introduce an eventuality. The ones that do are called *substantive*, those that do not are called *vacuous* or *non-substantive* (Van Eynde 2000b).

Each eventuality also introduces a *location time*. The location time is a time with respect to which the eventuality is located. It is similar to Reichenbach's reference time (Reichenbach 1947) and it is used here to express the difference in effect of both tenses and frame adjuncts on the eventuality time of telic and atelic eventualities.

The *utterance time* is the time within which the utterance takes place and, in our theory, it is assumed to be constant throughout the text. To explain some phenomena (e.g. transposition, flashback), a *temporal perspective time* is sometimes used, but in this paper, you can assume that it is part of

the time of utterance.² Note that it is used in the theory because the actual implementation does deal with some of these phenomena (Verdoolaege et al. 2000).

As to adjuncts, our theory is mainly based on Schelkens et al. (2000). We distinguish between *frame* or *locating adjuncts* indicating when an eventuality occurs, and *durational adjuncts* expressing how long it takes.³ Amongst the frame adjuncts, we further distinguish between *deictic* adjuncts that refer to the utterance time (or more generally, to the temporal perspective time) and *independent* adjuncts that refer directly to the time axis. In this paper, we do not deal with the so-called anaphoric adjuncts.

3 Representation language

As already mentioned, the theory is represented in a language that is essentially first order logic (FOL). It is extended with some axioms and notational conveniences, which will be explained in this section.

First of all, we want different constants to represent different entities and, more generally, we want different functors and functors with differing arguments to represent different entities. This means that constants and functors identify objects; functors can be seen as constructors. To accomplish this, the so-called Unique Names Axioms (UNA) are added to the theory (Reiter 1980) (Clark 1978). Since there are an infinite number of such axioms, they are (implicitly) built into the solver.

Sometimes, however, we want to use open functions, that is, functions that do not identify an object, but rather whose result can be equal to the result of another function. The particular solver we use, assumes UNA for every functor, so we cannot represent an n -ary open function with a functor. Instead we can use an $(n + 1)$ -ary predicate, with the extra argument representing the result of the function and with an axiom ensuring the existence and uniqueness of the function result in function of its arguments. For example, a binary function mapping a country and a year to the person that is or was the president of that country in that year, would be represented by a ternary predicate, e.g. *pres(USA, 2000, Bill)*.

We use a special notation for open functions that also expresses that the arguments and the result each satisfy a predicate. For example, the *pres* function mapping a country and a year to a person is represented as follows,

²See Kamp and Reyle (1993) chapter 5, section 4 or Van Eynde 1998, p. 245.

³We do not deal with frequency adjuncts in this paper.

where the **of** indicates the introduction of an open function.

$$(1) \quad \text{of } \textit{pres} : \textit{country}(_), \textit{year}(_) \rightarrow \textit{person}(_).$$

The above declaration is equivalent to a set of axioms, but it is shorter and easier to understand and it allows reasoning procedures working on a specification to handle such constraints more efficiently. The following is the more long-winded version:

$$(2) \quad \begin{array}{l} \text{fol} \quad \forall(C, Y) : \textit{country}(C) \ \& \ \textit{year}(Y) \\ \quad \Rightarrow \exists(P) : \textit{person}(P) \ \& \ \textit{pres}(C, Y, P). \\ \text{fol} \ \forall(C, Y, P1, P2) : \textit{pres}(C, Y, P1) \ \& \ \textit{pres}(C, Y, P2) \Rightarrow P1 = P2. \\ \text{fol} \quad \forall(C, Y, P) : \textit{pres}(C, Y, P) \Rightarrow \textit{country}(C) \ \& \ \textit{year}(Y). \end{array}$$

The **fol** marker indicates that what follows is a first order logic formula. These axioms express, first, that for each country and each year, there is a person that is the president of that country in that year, i.e. *pres* is a total function. Second, that for a given country and year, there is at most one president of that country in that year. Third, that for each instantiation of the president predicate, the first two arguments are a country and a year respectively.

While FOL is ideally suited to represent assertional knowledge, that is, facts and axioms, it does not fare so well when it comes to definitional knowledge, i.e. to defining concepts. A definition of a concept is an exhaustive enumeration of the cases in which some object belongs to the concept. We use a notation borrowed from logic programming. For example, the concept *uncle* is defined as either a male sibling (i.e. brother) of a parent or a male spouse of a (presumably female) sibling of a parent:

$$(3) \quad \begin{array}{l} \textit{uncle}(S, C) \leftarrow \textit{male}(S) \ \& \ \textit{sibling}(S, P) \ \& \ \textit{parent}(P, C). \\ \textit{uncle}(S, C) \leftarrow \textit{male}(S) \ \& \ \textit{married}(S, A) \ \& \\ \quad \textit{sibling}(A, P) \ \& \ \textit{parent}(P, C). \end{array}$$

Predicates that are not defined are called open predicates. Open functions are always open predicates.

For the simple, non-recursive, definitions we use in our theory, such a definition is equivalent to its completion (Clark 1978).⁴ The completion of a definition states that a predicate defined in it, holds if and only if one of its cases holds. That is, the defined predicate, which is the head of each rule, is equivalent to the disjunction of the bodies of the rules. In order to be able to

⁴The equivalence also holds for a certain subset of recursive definitions.

take the disjunction of the bodies, the heads have to be identical, of course. To accomplish this, all terms in an argument position of the predicate in the head are first moved to the body as an equality to the variable representing that argument, and all variables local to the body are quantified in the body. That is the set of m rules defining $p/n: \forall x_i : p(t_i) \leftarrow F_i$ is turned into the following equivalence:

$$(4) \quad \forall \mathbf{z} : p(\mathbf{z}) \leftrightarrow \left\{ \begin{array}{l} (\exists x_1 : \mathbf{z} = \mathbf{t}_1 \wedge F_1) \\ \vee \\ \dots \\ \vee \\ (\exists x_m : \mathbf{z} = \mathbf{t}_m \wedge F_m) \end{array} \right.$$

Denecker (2000) presents the extension of classical logic with a more general notion of (inductive) definitions.

4 Representation

The way in which information can be extracted from a theory is of course largely dependent on how the theory is represented. We will therefore first discuss this representation and only in the following sections will we present the extraction part.

This section will mainly focus on the sentence “[_S [_{NP} Ik] [_{VP} [_{V-AUX} ben] [_{VP} [_{ADV} gisteren] [_{VP} [_{ADJ} ziek] [_{V-MAIN} geweest]]]]]” (I have been sick yesterday), showing how this sentence is represented and showing the parts of the theory needed to extract information from it.⁵

4.1 Input

The sentence “Ik ben gisteren ziek geweest” contains three interesting words when it comes to temporal information, viz. “ben”, “geweest” and “gisteren”. The first two are both forms of the verb “zijn” (to be), the last is a temporal adjunct.

To be able to refer to different occurrences of the same word (such as “zijn” in the example, which occurs both in its past participle form and in its present tense form), we make use of tokens which are arbitrarily chosen constants that represent these occurrences. In our example sentence ($s1$),

⁵Note that in general, there is no direct correspondence between Dutch and English tenses.

we have verb tokens $w1$ for the main verb and $w2$ for the auxiliary verb, and an adjunct token $a1$.

To express which word is associated with a token, we use the *verbt_word* and *adjt_word* predicates. For verbs, we additionally use the *vform* predicate to indicate the verb form. Since all of these are exhaustive enumerations, they are defined predicates. For our example sentence we have:

$$\begin{aligned}
 (5) \quad & \textit{verbt_word}(w1, \textit{zijn}) && \leftarrow \textit{true.} \\
 & \textit{verbt_word}(w2, \textit{zijn}) && \leftarrow \textit{true.} \\
 & \textit{adjt_word}(a1, \textit{gisteren}) && \leftarrow \textit{true.} \\
 & \textit{vform}(w1, \textit{past_participle}) && \leftarrow \textit{true.} \\
 & \textit{vform}(w2, \textit{present_tense}) && \leftarrow \textit{true.}
 \end{aligned}$$

To express further that $s1$ is a clause, that $w1$ is the main verb of $s1$, that $w2$ is an auxiliary verb with $w1$ as its complement and that $a1$ is an adjunct in $s1$, we use the following:

$$\begin{aligned}
 (6) \quad & \textit{clause}(s1) && \leftarrow \textit{true.} \\
 & \textit{main_verb}(s1, w1) && \leftarrow \textit{true.} \\
 & \textit{aux_verb}(w2, w1) && \leftarrow \textit{true.} \\
 & \textit{s_adjunct}(s1, a1) && \leftarrow \textit{true.}
 \end{aligned}$$

Here again, we are dealing with exhaustive enumerations and thus definitions.

4.2 Periods of time

Part of the objective is to interrelate the different times associated with a text, so it is necessary that they are all comparable to each other. The allowed kinds of periods of time are intervals, denoted by the *int* predicate, and points, denoted by the *point* predicate.

Intervals refer directly to the time axis. They are represented by a pair of points on the time axis as arguments to an *int* functor. This does not preclude intervals from having unknown end points. The points on the time axis themselves can be left unspecified. The relations between intervals (*overlap*, *within*, *before* and *meets*) and some other properties about intervals that we will see later (e.g. *day_a* and *hour*) are defined in terms of relations and properties of these end points. We will not show them here.

As to the actual representation of points on the time axis, it is not trivial to find one that allows efficient processing for all the different kinds of constraints that a text can place on the intervals that it (implicitly or

explicitly) deals with. For now, we have contented ourselves with a fairly simple one that is easy to process by humans. Each point on the axis is represented by a function (ts) of four integers, the year, the month, the day of the month and the hour.⁶ For example, the whole of May the 21st 1976 is represented as $int(ts(1976, 5, 21, 0), ts(1976, 5, 22, 0))$. The relations and properties of these point are in turn defined in terms of relations on their composing parts.

Most of the theory is not concerned with the exact representation of periods of time, but instead only expresses relations between pairs of such periods through the following predicates. The relations that can be specified form only a subset of those in Allen (1983), but are sufficient for this application.

- *overlap*, which specifies that its arguments have a non-empty intersection,
- *within*, a special case of overlap which specifies that its first argument is situated completely within its second argument,
- *before*, which places its first argument completely before its second argument, and
- *meets*, a special case of before which specifies that its first argument immediately precedes its second argument.

4.3 Tenses and auxiliaries

The Dutch language has two simple tenses, the simple present and the simple past, and several others that require auxiliaries. Each is identified by a sequence of auxiliaries, each having the next as a complement (the last has the main verb as its complement), and the tense of the first auxiliary. Actually, it is a sequence of classes of auxiliaries (e.g. an auxiliary of the perfect or an auxiliary of the future). In our example sentence, “Ik ben gisteren ziek geweest”, there is one auxiliary ($w2$, “zijn”) and it is one of the perfect. The tense of this auxiliary is the present tense.

Only some verbs are auxiliaries and there are several kinds of auxiliaries (e.g. perfect, future). This information is represented by the *verb_aux_kind* predicate. For example, “zijn” has (at least) three uses: one as the main verb of a clause (v_zijn , such as $w1$), one as a temporal auxiliary (t_zijn , such

⁶The rather low resolution is due to technical limitations.

as *w2*) and one as an aspectual auxiliary (*a-zijn*). Both uses of “zijn” as an auxiliary are auxiliaries of the perfect.

$$(7) \quad \begin{array}{l} \text{verb_aux_kind}(t_zijn, \text{perfect}) \leftarrow \text{true.} \\ \text{verb_aux_kind}(a_zijn, \text{perfect}) \leftarrow \text{true.} \end{array}$$

Each meaning of “zijn” refers to the same verb lexeme. The *verb_lex* predicate enumerates the lexemes and we naturally only show part of its definition here.

$$(8) \quad \begin{array}{l} \text{verb_lex}(t_zijn, \text{zijn}) \leftarrow \text{true.} \\ \text{verb_lex}(a_zijn, \text{zijn}) \leftarrow \text{true.} \\ \text{verb_lex}(v_zijn, \text{zijn}) \leftarrow \text{true.} \end{array}$$

It determines of which verb a verb token can be an occurrence, by placing a constraint on the open function *token_verb* that maps verb tokens to their corresponding verbs. The constraint is that the verb is one of the possible meanings of the word associated with the verb token, i.e. that the word of the verb token is the same as the word of the verb.

$$(9) \quad \begin{array}{l} \text{of } token_verb : \text{verb_token}(_) \rightarrow \text{verb}(_). \\ \text{fo1 } \forall(T, V, W, L) : token_verb(T, V) \ \& \\ \text{verb_t_word}(T, W) \ \& \text{verb_lex}(V, L) \Rightarrow W = L. \end{array}$$

The *verb_token* and *verb* predicates merely enumerate the verb tokens and verbs respectively.

For each predicate representing one of the other properties of verbs (such as *verb_aux_kind* above), there is a corresponding predicate for verb tokens, that is defined to be that of the verb for which it is a token. The predicates have the same name, with an extra *verb_* prefix for the one on verbs. For example:

$$(10) \quad \text{aux_kind}(W) \leftarrow \exists(V) : token_verb(W, V) \ \& \ \text{verb_aux_kind}(V).$$

For *aux_kind*, we need to place an extra (indirect) constraint on the *token_verb* predicate. The tokens that appear as an auxiliary in the sentence (i.e. $\exists(W2) : \text{aux_verb}(W, W2)$) should be precisely those whose verb is an auxiliary (i.e. $\exists(F) : \text{aux_kind}(W, F)$).

$$(11) \quad \text{fo1 } \forall(W) : (\exists(W2) : \text{aux_verb}(W, W2)) \Leftrightarrow (\exists(F) : \text{aux_kind}(W, F)).$$

Another distinction among auxiliaries made in the adapted theory is the one between temporal (e.g. *t-zijn*) and aspectual (e.g. *a-zijn*) auxiliaries. Temporal auxiliaries have a purely temporal meaning and do not introduce

an eventuality (i.e. they are vacuous). Aspectual auxiliaries, on the other hand, have a temporal influence only indirectly through the eventualities they introduce. For example, the auxiliaries of the perfect can be either temporal or aspectual. The temporal perfect auxiliary requires the location time of the verb it has as a complement to be situated before the temporal perspective time, whereas the aspectual perfect auxiliary introduces a new eventuality that represents the resulting state of what its complement represents.

Substantivity is another one of those properties on verbs (*verb_subst*) that is inherited by verb tokens (*subst*). Due to the low number of vacuous verbs, it is simpler to list those instead of all the substantive ones and then to define the substantive verbs as those that are not vacuous. We only show part of the enumeration of vacuous verbs.

$$(12) \quad \text{verb_subst}(V) \leftarrow \neg \text{verb_vacuous}(V).$$

$$(13) \quad \begin{array}{l} \text{verb_vacuous}(t_hebben) \leftarrow \text{true.} \\ \text{verb_vacuous}(t_zijn) \leftarrow \text{true.} \\ \text{verb_vacuous}(t_zullen) \leftarrow \text{true.} \end{array}$$

Substantivity determines which verb tokens are associated with an eventuality. The eventuality is indicated by the unary *evt* functor and an additional *isevt* predicate indicates which *evts* represent eventualities.

$$(14) \quad \text{isevt}(\text{evt}(W)) \leftarrow \text{subst}(W).$$

Each eventuality has both an eventuality time and a location time associated with it. The eventuality time is expressed through the *evtttime* predicate. Similarly, the location time uses the *loc* predicate. The second argument of both is an interval and is required to be unique for a given eventuality. This can trivially be expressed by a set of axioms that have been omitted here. In general, an overlap relation holds between the two (expressed in the first of the following axioms). The second axiom expresses the fact that for telic eventualities this overlap is narrowed down to inclusion.

$$(15) \quad \begin{array}{l} \text{fol} \quad \quad \quad \forall(E) : \text{isevt}(E) \Rightarrow \\ (\exists(L, T) : \text{loc}(E, L) \ \& \ \text{evtttime}(E, T) \ \& \ \text{overlap}(T, L)). \\ \text{fol} \quad \quad \quad \forall(E) : \text{telic}(E) \Rightarrow \\ (\exists(L, T) : \text{loc}(E, L) \ \& \ \text{evtttime}(E, T) \ \& \ \text{within}(T, L)). \end{array}$$

$$(16) \quad \begin{array}{l} \text{fol} \quad \quad \quad \forall(W) : \text{vform}(W, \text{past_participle}) \\ \Rightarrow (\exists(L) : \text{subst}(W) \ \& \ \text{loc}(\text{evt}(W), L) \ \& \ \text{bounded}(L)). \end{array}$$

As already explained, the axiom states that the aspectual (substantive) perfect introduces the resulting state of the eventuality of its complement. Additionally, in the theory, there is a requirement for the complement to be non-stative, which is why $w2$ cannot be aspectual, since in our example sentence, $w1$ is stative. The definition of the *stative* predicate is similar to *subst*.

The *result* predicate in the above axiom is an open predicate expressing that the second argument represents the resulting state of the first argument. This implies that the resulting state immediately follows the eventuality that it is a resulting state for.

$$(20) \quad \text{fol } \forall(E, E2, L, L2) : \text{result}(E, E2) \ \& \ \text{evtime}(E, L) \ \& \ \text{evtime}(E2, L2) \Rightarrow \text{meets}(L, L2).$$

4.4 Adjuncts

We associate with each adjunct token the period of time that it refers to. For each adjunct, there is an axiom that specifies or somehow constrains this period of time. For durational adjuncts, which we will not discuss further in this paper, only the length of the period of time is constrained by the adjunct.

For example, the adjunct “gisteren” (yesterday), is a frame adjunct that refers to the day before the day that includes the temporal perspective time.

$$(21) \quad \text{fol} \quad \forall(A, Y, P) : \text{adjt_word}(A, \text{gisteren}) \ \& \ \text{adjtime}(A, Y) \ \& \ \text{adjt_ppp}(A, P) \Rightarrow (\exists(T) : \text{day_a}(Y), \text{day_a}(T), \text{within}(P, T) \ \& \ \text{meets}(Y, T)).$$

Here, *adjtime* is another open function linking an adjunct to its associated period of time and *adjt_ppp* does the same for the temporal perspective time. The first is an open function from “adjunctoids”, which refers to anything that occurs as a first argument to the *adjt_word* predicate, to intervals. The second is defined as the temporal perspective time of the modified verb. We will show the use of adjunctoids later in this subsection.

$$(22) \quad \text{adjunctoid}(A) \leftarrow \exists(L) : \text{adjt_word}(A, L). \\ \text{of } \text{adjtime} : \text{adjunctoid}(_) \rightarrow \text{int}(_).$$

The effect of a frame adjunct is that the location time of the modified verb is required to be within the frame period:

$$(23) \quad \text{fol } \forall(A, T, W, L) : \text{adjunct_verb}(A, W) \ \& \ \text{frame}(A) \ \& \ \text{adjtime}(A, T) \ \& \ \text{loc}(\text{evt}(W), L) \Rightarrow \text{within}(L, T).$$

The *adjunct_verb* predicate is an open function that maps adjuncts (anything that appears as a second argument to *s_adjunct*) to one of the substantive verbs in the clause. This mapping is not specified in the input, since in general, when a clause contains more than one (possibly) substantive verb, we do not know a priori which of these verbs is modified by the adjunct. The *frame* predicate lists all the frame adjuncts. Part of its definition is as follows:

$$(24) \quad \text{frame}(A) \leftarrow \text{adjt_word}(A, \text{gisteren}).$$

This setting also allows for some more complicated uses, for example, *na* (after) followed by something that could be used as a frame adjunct. The use of *na* indicates that the location time of the modified verb is situated after the period of time of the complement of the preposition. This can be modeled as inclusion in a frame interval that immediately succeeds the frame interval of the complement.

$$(25) \quad \text{fol } \forall(A, B, T, F, P) : \text{adjt_word}(A, \text{na}(B)) \ \& \ \text{adjtime}(A, T) \ \& \ \text{frame}(B) \ \& \ \text{adjtime}(B, F) \Rightarrow \text{meets}(F, T)$$

The definition of *frame* is extended accordingly:

$$(26) \quad \text{frame}(A) \leftarrow \text{adjt_word}(A, \text{na}(B)) \ \& \ \text{frame}(B).$$

In the input, *na gisteren* would be specified as follows:

$$(27) \quad \begin{array}{ll} \text{adjt_word}(a1, \text{gisteren}) & \leftarrow \text{true.} \\ \text{adjt_word}(a2, \text{na}(a1)) & \leftarrow \text{true.} \\ \text{s_adjunct}(s1, a2) & \leftarrow \text{true.} \end{array}$$

Schelkens et al. (2000) also discusses so-called *point-like* frame adjuncts such as “om x uur” (at x o’clock) and argues that they affect both the location time and the eventuality time, in that the point described by the adjunct is included in both. As this is different from the behaviour of frame adjuncts belonging to the *frame* predicate, we introduce another one, *point_frame*, which lists all the point-like frame adjuncts. The effect these have is described by the following axiom.

$$(28) \quad \begin{array}{l} \text{fol} \quad \forall(A, T, W, L, E) : \\ \text{point_frame}(A) \ \& \ \text{adjtime}(A, T) \ \& \ \text{verb_adjunct}(W, A) \ \& \\ \text{loc}(\text{evt}(W), L) \ \& \ \text{evtime}(\text{evt}(W), E) \Rightarrow \\ \text{within}(T, L) \ \& \ \text{within}(T, E). \end{array}$$

5 Reasoning procedure

Up to this point, we have presented a logic theory consisting of FOL axioms and definitions in which some predicates were defined and others open, but we have not shown how to derive information from this theory. Since the defined predicates are known, the only uncertainty lies in the open predicates. We need to find an interpretation for these open predicates that is consistent with the theory. In other words, we need to generate a model for the open predicates. To do this we use an existing abductive procedure, called SLDNFA (Denecker and Van Nuffelen 1999), which operates on theories written in the knowledge representation language outlined in section 3.

Abduction is a form of non-monotonic reasoning that is used to explain observations. In this case, an explanation consists of a model for the open predicates. To explain some observation, we sometimes have to assume (abduce) other information. This form of reasoning is called non-monotonic, because new information may invalidate previously drawn conclusions.

We cannot give a detailed explanation of how the SLDNFA procedure works within this limited space, but we do want to give an idea of it. In short, the procedure tries to make the conjunction of all axioms and the (possibly empty) observation (query) hold by abducting a set of atoms, according to the following rules. A conjunction holds if all of its conjuncts hold; for a disjunction, it is sufficient that one of its disjuncts holds, so each one is tried out until one is found that holds. Negation is distributed over disjunctions and conjunctions. A defined predicate is replaced by its completion (see section 3).

When an open predicate occurs negatively, the atom is (temporarily) assumed not to hold if this does not conflict with earlier made assumptions. If it is part of a disjunction, the remaining disjuncts are remembered in case it turns out we want it to hold after all. When an open predicate occurs positively, the atom is assumed to hold and any applicable remembered disjunction is required to hold as well. If the atom was already assumed not to hold and there were no remaining disjuncts at that point, the procedure backtracks to the latest disjunction with remaining disjuncts. Finally, an equality unifies its arguments. If this fails, the procedure backtracks as well.

Numerical operations and comparisons are translated into CLP (Constraint Logic Programming) constraints and handed over to an efficient constraint solver. They occur mainly in the definitions and axioms pertaining to points on the time axis, which we have not shown in this paper. At the end, all numerical variables are labeled, which means that they get a value assigned to them that satisfies all constraints.

6 Deriving temporal information

Applying the procedure of section 5 to our theory with an empty observation (that is, we just want a consistent interpretation for the open predicates) for our example sentence “Ik ben gisteren ziek geweest”, we get the following result, which lists for each open predicate precisely its model:

```

adjt_ppp: [adjt_ppp(a1,int(ts(1999,1,2,0),ts(1999,1,2,1)))]
adjtime: [adjtime(a1,int(ts(1999,1,1,0),ts(1999,1,2,0)))]
adjunct_verb: [adjunct_verb(a1,w1)]
evtttime: [evtttime(evt(w1),int(ts(1999,1,1,0),ts(1999,1,2,0))),
            evtttime(utt,int(ts(1999,1,1,0),ts(1999,1,3,0)))]
loc: [loc(evt(w1),int(ts(1999,1,1,0),ts(1999,1,2,0)))]
s_ppp: [s_ppp(s1,int(ts(1999,1,2,0),ts(1999,1,2,1)))]
token_verb: [token_verb(w2,t_zijn),token_verb(w1,v_zijn)]

```

We will now show how this result can be obtained from the given axioms and definitions, without, however, following the exact procedure as presented in section 5 as that would be too tedious.

The model we construct should satisfy each axiom. Arguably the simplest axiom is the one requiring the existence of an utterance time (17), which requires us to abduce an instance of the *evtttime* predicate, viz. one with *utt* as first argument and *some U* as second argument. Further constraints will narrow down the possible values for this *U*.

Next, we look at the correspondence between verb tokens and verbs, i.e. *token_verb*. Equation (9) shows that *token_verb* is a total function from verb tokens to verbs with the restriction that they refer to the same lexeme. The input (5) shows that both verb tokens (*w1* and *w2*) refer to *zijn*, for which equation (8) allows three possibilities. Now, we know from (6) that *w1* is not an auxiliary, so, based on (7), it cannot be *t_zijn* or *a_zijn*, so it must be *v_zijn*. Similarly, *w2* is an auxiliary and since the axiom on the aspectual perfect (19) requires non-stativity of the complement, whereas *v_zijn* is stative, it must be temporal, i.e. *t_zijn*.⁷ The abductive procedure would abduce each possibility in turn and would only reconsider when it hits a contradiction such as mentioned above.

Since *v_zijn* is substantive (it is not listed as vacuous (13)), it introduces an eventuality (14), and therefore it has overlapping eventuality and location times (15). Since *w1* is furthermore a past participle, the location time is bounded (16) and thus even *includes* the eventuality time. As to the relation

⁷The definition of *stative* is not shown, but is similar to *subst* and includes *v_zijn*.

with the temporal perspective time, the *loc_ppp* definition (18) specifies a before relation for temporal (non substantive) auxiliaries of the perfect.

Finally, we look at the information contained in the adjunct. As mentioned in section 4.4, *adjunct_verb* associates adjunct tokens to tokens of substantive verbs of the same clause. In this case there is only one substantive verb (*w1*), so the adjunct can only modify that verb. Since we are dealing with the adjunct “gisteren”, the right-hand side of implication in axiom (21) has to hold, namely that the adjunct refers to a day that immediately precedes the day that contains the temporal perspective time. Axiom (23) then further ensures that this period of time includes the location time of *evt(w1)*.

At this stage, all the axioms are satisfied, but our model is underspecified in that some abduced atoms still contain variables, notably those that refer to time intervals. We can then choose these time intervals, of course taking into account the constraints that have been placed on them. In this case, the abductive procedure has chosen the temporal perspective time to be the first hour of January the second, 1999. It is included in the utterance time, which, as you can see, is rather large, stretching over two days.⁸ The adjunct “gisteren” (yesterday) then of course refers to January the first and this includes the location time of the eventuality associated with *w1* (the whole day, here), which in turn includes the eventuality time (also the whole day, here).

When more information is given, the system does not have that much choice. Suppose, for example, that you know that the sickness lasted from 6 o'clock until 8 o'clock in the evening on May the 21st, 2000 and that the utterance time lasted an hour, you would give the following observation (query) to the system:

$$(29) \quad \text{utt}(U) \ \& \ \text{hour}(U) \ \& \ \text{evttime}(\text{evt}(w1), \text{int}(\text{ts}(2000, 5, 21, 18), \text{ts}(2000, 5, 21, 20)))$$

The result is, as you would expect, that the utterance happened on May the 22nd. We only show the result for the *evttime* predicate; the other time intervals are changed accordingly.

```
evttime: [evttime(utt,int(ts(2000,5,22,0),ts(2000,5,22,1))),
          evttime(evt(w1),int(ts(2000,5,21,18),ts(2000,5,21,20)))]
```

An example of another kind of query, is the following. Is it possible for the utterance to have taken place before the eventuality described by *w1* ?

$$(30) \quad \text{utt}(U) \ \& \ \text{evttime}(\text{evt}(w1), T) \ \& \ \text{before}(U, T)$$

⁸Maybe some general restriction should be placed on the extent of the utterance time.

The result:

no

The example sentence used is the previous queries, gave rise to at most one model (apart from the possible choice of time periods). We can also analyze sentences with ambiguities. For example, (?) argues that sentence 29, reproduced below, has two possible interpretations, depending on which of the verbs is taken to be modified by the adjunct.

- (29) Ze had om vier uur al een hamburger gegeten.
she have.PAST at four hour already a hamburger eat.PSP
'She had already eaten a hamburger at four o'clock.'

The input is a straightforward transliteration:

- (31)
- | | | |
|-----------------------------------|---|-------|
| <i>main_verb(s1, w1)</i> | ← | true. |
| <i>aux_verb(w2, w1)</i> | ← | true. |
| <i>verbt_word(w1, eten)</i> | ← | true. |
| <i>verbt_word(w2, hebben)</i> | ← | true. |
| <i>vform(w1, past_participle)</i> | ← | true. |
| <i>vform(w2, past_tense)</i> | ← | true. |
| <i>s_adjunct(s1, a1)</i> | ← | true. |
| <i>adjt_word(a1, om(4))</i> | ← | true. |

When given this sentence as input, we do indeed get two significantly different models, of which we only show the relevant parts. Both have identified *w2* as an aspectual perfect. According to (19) and (20), this means that the eventuality of having eaten (*evt(w2)*) immediately follows the eventuality of eating (*evt(w1)*). The first model then takes the adjunct to modify the main verb and (28) forces the time referred to by the adjunct to included in the eating.

```
adjtime: [adjtime(a1,int(ts(1999,1,1,4),ts(1999,1,1,5)))]
adjunct_verb: [adjunct_verb(a1,w1)]
evtttime: [evtttime(evt(w1),int(ts(1999,1,1,0),ts(1999,1,2,0))),
           evtttime(evt(w2),int(ts(1999,1,2,0),ts(1999,1,3,0)))]
token_verb: [token_verb(w2,a_hebben),token_verb(w1,v_eten)]
```

Similarly, in the second model the auxiliary is modified by the adjunct and it is the having eaten that includes the time referred to by the adjunct.

```

adjtime: [adjtime(a1,int(ts(1999,1,2,4),ts(1999,1,2,5)))]
adjunct_verb: [adjunct_verb(a1,w2)]
evtttime: [evtttime(evt(w1),int(ts(1999,1,1,0),ts(1999,1,2,0))),
            evtttime(evt(w2),int(ts(1999,1,2,0),ts(1999,1,3,0)))]
token_verb: [token_verb(w2,a_hebben),token_verb(w1,v_eten)]

```

7 Conclusions

In this paper, we have shown a formalisation in first order logic of an existing theory about temporal information in Dutch texts. Although to a large extent, this theory had already been formalised in HPSG, it had not been practically implemented as such.

The representation shown in this paper has effectively been implemented and results of using an abductive reasoning procedure on it were presented. Although the problems dealt with in this paper were limited, the experiments show that abduction may be viable for natural language processing.

A representation in logic is not only very flexible and extendible, a good representation also requires the use of well thought out concepts, because of logic's clear and formal semantics. This representation has already helped in clearing out the meaning of some concepts used in the temporal analysis of sentences.

The research that led to this paper has also been an exercise in knowledge representation and will contribute toward a better knowledge representation methodology.

References

- Allen, J. (1983). Maintaining Knowledge About Temporal Intervals. *Communications of the ACM* 26(11), 832–843.
- Clark, K. (1978). Negation as failure. In H. Gallaire and J. Minker (Eds.), *Logic and Databases*, pp. 293–322. Plenum Press.
- Denecker, M. (2000, April 9-11). Extending classical logic with inductive definitions. In C. Baral and M. Truszczynski (Eds.), *Proceedings of NMR'2000*, pp. 1–15. 8th Intl. Workshop on Non-Monotonic Reasoning.
- Denecker, M. and B. Van Nuffelen (1999). Experiments for integration CLP and abduction. In K. R. Apt, A. C. Kakas, E. Monfroy, and

- F. Rossi (Eds.), *Proceedings of the 1999 ERCIM/COMPULOG workshop on Constraints*, pp. 1–15.
- Kamp, H. and U. Reyle (1993). *From Discourse to Logic*. Kluwer Academic Publishers.
- Pollard, C. and I. A. Sag (1994). *Head-Driven Phrase Structure Grammar*. The University of Chicago Press.
- Reichenbach, H. (1947). *Elements of Symbolic Logic*. The Free Press, New York. reprint 1966.
- Reiter, R. (1980). Equality and domain closure in first-order databases. *JACM* 27, 235–249.
- Schelkens, N., F. Van Eynde, and S. Verdoolaege (2000). The semantics of temporal adjuncts. In P. Monachesi (Ed.), *Computational Linguistics in the Netherlands 1999*. Utrecht: University of Utrecht.
- Van Eynde, F. (1998). Tense, Aspect and Negation. In F. Van Eynde and P. Schmidt (Eds.), *Linguistic Specifications for Typed Feature Structure Formalisms*, pp. 209–280. European Communities.
- Van Eynde, F. (2000a). A constraint-based semantics for tenses and temporal auxiliaries. To be published in R. Cann, C. Grover & P. Miller (eds.), *Grammatical Interfaces in HPSG*. CSLI-Stanford.
- Van Eynde, F. (2000b). Figure Heads in HPSG. In F. Van Eynde, I. Schuurman, and N. Schelkens (Eds.), *Computational Linguistics in the Netherlands 1998*, Number 29 in *Language and Computers: Studies in Practical Linguistics*, pp. 161–178. Rodopi, Amsterdam.
- Verdoolaege, S., M. Denecker, N. Schelkens, D. De Schreye, and F. Van Eynde (2000). Semantic interpretation of temporal information by abductive inference. In P. Monachesi (Ed.), *Computational Linguistics in the Netherlands 1999*. Utrecht: University of Utrecht.