Semantic interpretation of temporal information by abductive inference

Sven Verdoolaege[1], **Marc Denecker**[1], **Ness Schelkens**[2], **Danny De Schreye**[1] **and Frank Van Eynde**[2]
[1]Department of Computer Science and [2]Centre for Computational Linguistics
K.U.Leuven

Abstract

Besides temporal information explicitly available in verbs and adjuncts, the temporal interpretation of a text also depends on general world knowledge and default assumptions. 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.

The theory is formulated in logic and is a practical implementation of the concepts described in Ness Schelkens et al. (this volume). 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 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 2000 and Van Eynde (ms.), which integrates a DRT-style analysis (Kamp and Reyle 1993) of tense and aspect into HPSG (Pollard and Sag 1994). Although from inception, HPSG was aimed at representing not only syntax, but also semantics and pragmatics, when it comes to practical implementation, the focus has clearly been on syntax. In contrast, the adapted theory deals mainly with semantics and pragmatics. Furthermore, recent research suggests that a direct implementation of HPSG may not be computationally viable (Richter, Sailer and Penn 1999).

For these reasons and because of our experience with it, we have chosen to formulate the theory in logic, rather than trying to extend an existing HPSG solution. As such we do not use typed feature structure, but represent much the same information in a different way. A formulation in logic should also allow the linguistic knowledge to be easily extended with additional 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

[1]In this paper, we only deal with sentences.

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* (Van Eynde 2000).

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. It turns out that by mediating these effects through the location time, only the relationship between eventuality time and location time needs to be telicity dependent.

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), we use the *temporal perspective time*, which can change from clause to clause.[2] We will from time to time refer to the temporal perspective time, but it is not essential for understanding this paper. In most circumstances, it can be assumed to be part of the utterance time. It is only in the transposed case that it will precede the utterance time.

As to adjuncts, our theory is mainly based on Schelkens et al. (this volume). 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.

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

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

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, where the `of` indicates the introduction of an open function.

(1) `of` $pres : country(\_), year(\_) \rightarrow person(\_)$.

The above declaration is equivalent to a set of axioms, but it is shorter to write down (and easier to understand at first glance) and it allows reasoning procedures working on a specification to handle such constraints more efficiently. The more long-winded version is shown in figure 1. 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

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

Figure 1: The open function *pres*.

in that year. 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:

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

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 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:

$$
(3)\quad \forall \mathbf{z} : p(\mathbf{z}) \leftrightarrow
\begin{cases}
(\exists x_1 : \mathbf{z} = \mathbf{t}_1 \wedge F_1) \\
\qquad \vee \\
\qquad \ldots \\
\qquad \vee \\
(\exists x_m :\ \mathbf{z} = \mathbf{t}_m \wedge F_m)
\end{cases}
$$

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 "$[_{\text{S}}\ [_{\text{NP}}$ Ik$]$ $[_{\text{VP}}\ [_{\text{V-AUX}}$ ben$]$ $[_{\text{VP}}\ [_{\text{ADV}}$ gisteren$]$ $[_{\text{VP}}$ $[_{\text{ADJ}}$ ziek$]$ $[_{\text{V-MAIN}}$ 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" en "gisteren". The first two are both forms of the verb "zijn" (to be), the last is a temporal adjunct.

Since our main interest is in semantic interpretation, we assume that any syntactic information that is unambiguous and thus independent of any further semantic analysis, is available as input. For this sentence, we have that "geweest" is the past participle form of "zijn", and that it is the main verb of the clause; "ben" is a present tense of "zijn" and is an auxiliary to "geweest"; and finally, "gisteren" is an adjunct in the same clause. Note that 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.

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

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

To be able to refer to different occurrences of the same word (such as “zijn” in the example), we make use of tokens which are arbitrarily chosen constants that represent these occurrences. In our example sentence (*s1*), 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 *morf* predicate to indicate the verb form. Since all of these are exhaustive enumerations, they are defined predicates. For our example sentence we have:

$$(4)\quad \begin{array}{lcl} \mathit{verbt\_word}(\mathit{w1}, \mathit{zijn}) & \leftarrow & \texttt{true}. \\ \mathit{verbt\_word}(\mathit{w2}, \mathit{zijn}) & \leftarrow & \texttt{true}. \\ \mathit{adjt\_word}(\mathit{a1}, \mathit{gisteren}) & \leftarrow & \texttt{true}. \\ \mathit{morf}(\mathit{w1}, \mathit{past\_participle}) & \leftarrow & \texttt{true}. \\ \mathit{morf}(\mathit{w2}, \mathit{present\_tense}) & \leftarrow & \texttt{true}. \end{array}$$

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:

$$(5)\quad \begin{array}{lcl} \mathit{clause}(\mathit{s1}) & \leftarrow & \texttt{true}. \\ \mathit{main\_verb}(\mathit{s1}, \mathit{w1}) & \leftarrow & \texttt{true}. \\ \mathit{aux\_verb}(\mathit{w2}, \mathit{w1}) & \leftarrow & \texttt{true}. \\ \mathit{s\_adjunct}(\mathit{s1}, \mathit{a1}) & \leftarrow & \texttt{true}. \end{array}$$

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

4.2 Eventualities and verbs

An eventuality associated with a verb token is indicated by the unary *evt* functor. Not every such construct really represents an eventuality, though. Only in the cases where the argument is a token for a substantive verb, does it refer to an eventuality. We therefore need an additional predicate *isevt* to indicate which *evt*s represent eventualities.

$$(6)\quad \mathit{isevt}(\mathit{evt}(W)) \quad \leftarrow \quad \mathit{subst}(W).$$

Now we still have to define *subst* itself. There is only a limited number of verbs that are not substantive, but we cannot discriminate between substantive and vacuous verbs on the basis of the lexeme. 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 as *w2*) and one as an aspectual auxiliary (*a_zijn*). Of these, only the use as a temporal auxiliary is vacuous. Section 4.4 deals with the distinction between temporal and aspectual auxiliaries.

We consider these three ways of using “zijn” to be three different verb meanings of the same verb lexeme. For each verb (meaning), the lexicon specifies whether it is substantive or not, as well as any other property of verbs that is independent of the actual use of the verb in a sentence, e.g. whether it is stative or dynamic. Since the properties of a verb hold for each occurrence of that verb, they are inherited by the verb tokens.

One of the properties of a verb, is the lexeme used. The *verb_lex* predicate enumerates them and we naturally only show part of its definition here.

$$(7)\quad \begin{array}{lcl} \mathit{verb\_lex}(\mathit{t\_zijn}, \mathit{zijn}) & \leftarrow & \texttt{true}. \\ \mathit{verb\_lex}(\mathit{a\_zijn}, \mathit{zijn}) & \leftarrow & \texttt{true}. \\ \mathit{verb\_lex}(\mathit{v\_zijn}, \mathit{zijn}) & \leftarrow & \texttt{true}. \end{array}$$

This property is not really inherited, but rather 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.

$$(8)\quad \begin{array}{l} \texttt{of}\ \mathit{token\_verb} : \mathit{verb\_token}(\_) \rightarrow \mathit{verb}(\_). \\ \texttt{fol}\ \forall(T, V, W, L) : \mathit{token\_verb}(T, V)\ \& \\ \quad \mathit{verbt\_word}(T, W)\ \&\ \mathit{verb\_lex}(V, L). \\ \qquad \Rightarrow W = L. \end{array}$$

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, 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, substantivity of verb tokens (*subst*) is defined as the substantivity of the corresponding verb (*verb_subst*).

$$(9)\quad \begin{array}{l} \mathit{subst}(W) \\ \quad \leftarrow \quad \exists(V) : \mathit{token\_verb}(W, V)\ \& \\ \qquad \mathit{verb\_subst}(V). \end{array}$$

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.

(10) $\textit{verb\_subst}(V) \quad \leftarrow \quad \neg\textit{verb\_vacuous}(V).$

(11)
$$\begin{array}{lll} \textit{verb\_vacuous}(\textit{t\_hebben}) & \leftarrow & \texttt{true}. \\ \textit{verb\_vacuous}(\textit{t\_zijn}) & \leftarrow & \texttt{true}. \\ \textit{verb\_vacuous}(\textit{t\_zullen}) & \leftarrow & \texttt{true}. \end{array}$$

4.3 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 no known 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 $\textit{int}(\textit{ts}(1976, 5, 21, 0), \textit{ts}(1976, 5, 22, 0))$. 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.

The eventuality time of an eventuality is associated with it through the *evt*time 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 is expressed by the following axioms:[7]

(12)
$$\begin{array}{l} \texttt{fol}\ \forall(E,T) : \textit{evttime}(E,T) \Rightarrow \textit{int}(T). \\ \texttt{fol}\ \ \forall(E,L) : \textit{loc}(E,L) \Rightarrow \textit{int}(L). \\ \texttt{fol}\ \ \forall(E, T1, T2) : \textit{evttime}(E, T1)\ \& \\ \qquad \textit{evttime}(E, T2) \Rightarrow (T1 = T2). \\ \texttt{fol}\ \ \ \forall(E, L1, L2) : \textit{loc}(E, L1)\ \& \\ \qquad\quad \textit{loc}(E, L2) \Rightarrow (L1 = L2). \end{array}$$

Each eventuality has both an eventuality time and a location time associated with it. Next, we express the overlap relation that generally holds between these two. The second axiom expresses the fact that for telic eventualities this overlap is narrowed down to inclusion.

(13)
$$\begin{array}{c} \texttt{fol}\ \forall(E) : \textit{isevt}(E) \Rightarrow (\exists(L,T) : \\ \textit{loc}(E,L)\ \&\ \textit{evttime}(E,T)\ \& \\ \textit{overlap}(T,L)). \\ \texttt{fol}\ \forall(E) : \textit{telic}(E) \Rightarrow (\exists(L,T) : \\ \textit{loc}(E,L)\ \&\ \textit{evttime}(E,T)\ \& \\ \textit{within}(T,L)). \end{array}$$

The utterance time is expressed through the unary *utt* predicate and is required to exist. Essentially, it is just the eventuality time of some

[6]The rather low resolution is due to technical limitations.

[7]The careful reader will have noticed that these are prime candidates for specification as open functions. The current solver, however, places rather severe restrictions on the domain predicate (*isevt* in this case, as shown in the next set of axioms), which are not met here.

"*utt*" eventuality.

(14)
$$\begin{array}{lcl} \texttt{fol} & & \exists(U) : utt(U). \\ utt(U) & \leftarrow & evttime(utt, U). \end{array}$$

Since the temporal perspective time can only change from clause to clause, each clause has a unique temporal perspective time and therefore it can be represented as an (open) function of clauses.

(15) $\texttt{of}\ s\_ppp : clause(\texttt{Z}) \rightarrow point(\_).$

In the rest of the theory we will often want to refer to the temporal perspective time of the clause that contains a given verb. For convenience, we therefore define another predicate *v_ppp*, relating verbs to temporal perspective times. In this definition we use the *s_verb* predicate, which enumerates all the verbs in a clause.

(16)
$$\begin{array}{l} v\_ppp(W, P) \\ \quad \leftarrow \quad \exists(Z) : s\_verb(Z, W)\ \& \\ \qquad\quad s\_ppp(Z, P). \end{array}$$

4.4 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, both uses of "zijn" as an auxiliary are auxiliaries of the perfect.

(17)
$$\begin{array}{lcl} verb\_aux\_kind(t\_zijn, perfect) & \leftarrow & \texttt{true}. \\ verb\_aux\_kind(a\_zijn, perfect) & \leftarrow & \texttt{true}. \end{array}$$

As with the other properties of verbs, the auxiliary kind of a verb is inherited by the verb tokens (as the *aux_kind* predicate). Here, 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. $\exists(W2) : aux\_verb(W, W2)$) should be precisely those whose verb is an auxiliary (i.e. $\exists(F) : aux\_kind(W, F)$).

(18)
$$\begin{array}{l} \texttt{fol}\ \forall(W) : (\exists(W2) : aux\_verb(W, W2)) \\ \qquad \Leftrightarrow (\exists(F) : aux\_kind(W, F)). \end{array}$$

This brings us to another distinction among auxiliaries made in the adapted theory and that 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.

Both the temporal auxiliaries and the tenses stipulate a relation between the location time of an eventuality and the temporal perspective time. For temporal auxiliaries it is the eventuality of the complement whose location time is constrained, for the tenses it is the eventuality of the tensed verb, in case it is substantive.

In our current theory, there are three kinds of relations that can hold between a location time and the temporal perspective time, which we have succinctly called *before*, *not_before* and *after*. Note that these names give only a rough idea. Further down in this section, we will show what exactly they imply. It turns out that the temporal perfect and the past tense have the same effect in this regard, namely the before relation.

There is a slight complication though, and that is that in case of transposition, the past tense specifies a before relation between the temporal perspective time and the utterance time, between which, as you may remember, normally holds an inclusion relation. Due to this, the transposed past tense does not specify a before relation between location time and temporal perspective time, but behaves rather

$$
\begin{array}{lll}
\textit{evt\_ppp}(W, \textit{before}) & \leftarrow & \textit{subst}(W)\ \&\ \textit{morf}(W, \textit{past\_tense})\ \&\ \neg\textit{transposed}(W) \vee \\
& & (\exists(A) : \textit{aux\_kind}(A, \textit{perfect})\ \&\ \neg\textit{subst}(A)\ \&\ \textit{aux\_verb}(A, W)). \\
\textit{evt\_ppp}(W, \textit{not\_before}) & \leftarrow & \textit{subst}(W)\ \&\ (\textit{morf}(W, \textit{past\_tense})\ \&\ \textit{transposed}(W)) \vee \\
& & \qquad \textit{morf}(W, \textit{present\_tense}). \\
\textit{evt\_ppp}(W, \textit{after}) & \leftarrow & \exists(A) : \textit{aux\_kind}(A, \textit{futurate})\ \&\ \textit{aux\_verb}(A, W).
\end{array}
$$

Figure 2: Relation between temporal perspective time and eventuality time.

like the present tense on this level, which we have called the "*not_before*" relation. For completeness, we mention that the futurate auxiliary specifies an after relation. All this leads to the definition for the relation between location time and temporal perspective time shown in figure 2.

As you can see, for each clause there is exactly one eventuality whose location time is related to the temporal perspective time, either that of the tensed verb in case it is substantive or that of the complement of the (vacuous) temporal auxiliary. You may wonder what happens when a temporal auxiliary has the past tense. The above definition seems to allow that this past tense has no effect (the present tense has indeed no effect, but you would not be surprised about that). As it happens, the past tense for a temporal auxiliary, can only be a transposed past tense and in this case the other relation, i.e. the one between temporal perspective time and utterance time comes into play, as we discussed above. We give the definition of this relation and the axiom enforcing the transposedness:

$$
\begin{array}{l}
\textit{ppp\_utt}(W, \textit{before}) \\
\quad \leftarrow \quad \textit{morf}(W, \textit{past\_tense})\ \&\ \textit{transposed}(W). \\
\textit{ppp\_utt}(W, \textit{at}) \\
\quad \leftarrow \quad \textit{morf}(W, \textit{present\_tense}) \vee \\
\qquad\quad \textit{morf}(W, \textit{past\_tense})\ \&\ \neg\textit{transposed}(W). \\
\texttt{fol}\ \forall(W) : \textit{morf}(W, \textit{past\_tense})\ \&\ \neg\textit{subst}(W) \\
\qquad\qquad \Rightarrow \textit{transposed}(W).
\end{array}
$$

(19)

We cannot go into the circumstances under which transposition arises, we merely define it here as an "abnormality", where *abnormal* is an open predicate:

$$
\textit{transposed}(W) \leftarrow \textit{abnormal}(\textit{ab\_transposed}(W)). \tag{20}
$$

As we already hinted at in section 4.2, the auxiliary (*w2*) in our example sentence can only be interpreted as a temporal auxiliary, but we have shown no rule that would preclude it from being interpreted as an aspectual one, simply because we have not shown the rule for the aspectual perfect. So here it is:

$$
\begin{array}{c}
\texttt{fol}\ \forall(W, C) : \textit{aux\_kind}(W, \textit{perfect})\ \& \\
\textit{subst}(W)\ \&\ \textit{aux\_verb}(W, C) \\
\Rightarrow \neg\textit{stative}(\textit{evt}(C))\ \& \\
\textit{result}(\textit{evt}(C), \textit{evt}(W)).
\end{array} \tag{21}
$$

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 the *subst* one shown in section 4.2.

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.

$$
\begin{array}{c}
\texttt{fol}\ \forall(E, E2, L, L2) : \textit{result}(E, E2)\ \& \\
\textit{evttime}(E, L)\ \&\ \textit{evttime}(E2, L2) \\
\Rightarrow \textit{meets}(L, L2).
\end{array} \tag{22}
$$

Now, as promised, the implications of the *evt_ppp* and *ppp_utt* relations. For the *ppp_utt* relation, *before* means exactly what you would expect and *at* means inclusion of the temporal perspective time in the utterance time.

$$
\begin{array}{c}
\texttt{fol}\ \forall(U, W, P) : \textit{ppp\_utt}(W, \textit{before})\ \& \\
\textit{v\_ppp}(W, P)\ \&\ \textit{utt}(U) \\
\Rightarrow \textit{before}(P, U). \\
\texttt{fol}\ \ \forall(U, W, P) : \textit{ppp\_utt}(W, \textit{at})\ \& \\
\textit{v\_ppp}(W, P)\ \&\ \textit{utt}(U) \\
\Rightarrow \textit{within}(P, U).
\end{array} \tag{23}
$$

For the *evt_ppp* relation, *not_before* means non precedence. (There are some further specialisations of this relation in restricted circumstance, but we will not show them here.) The *before* and *after* cases should be intuitively clear.

$$\begin{array}{l}\texttt{fol}\ \forall(W,P,L): \mathit{evt\_ppp}(W,\mathit{before})\ \&\\ \quad \mathit{loc}(\mathit{evt}(W),L)\ \&\ \mathit{v\_ppp}(W,P)\\ \qquad \Rightarrow \mathit{before}(L,P).\\ \texttt{fol}\ \forall(W,P,L): \mathit{evt\_ppp}(W,\mathit{not\_before})\ \&\\ \quad \mathit{loc}(\mathit{evt}(W),L)\ \&\ \mathit{v\_ppp}(W,P)\\ \qquad \Rightarrow \neg\mathit{before}(L,P).\\ \texttt{fol}\ \forall(W,P,L): \mathit{evt\_ppp}(W,\mathit{after})\ \&\\ \quad \mathit{loc}(\mathit{evt}(W),L)\ \&\ \mathit{v\_ppp}(W,P)\\ \qquad \Rightarrow \mathit{before}(P,L).\end{array} \tag{24}$$

4.5 Adjuncts

Although we currently deal with only a few adjuncts, we have tried to be as general as possible in our treatment of them. Our main design decisions were to have compositionality and to avoid redundancy. This means that we associate with an adjunct (token) only what the adjunct itself adds to the meaning of the sentence and we want to treat what is common to several adjuncts, separately.

For example, the adjunct "gisteren" (yesterday), is a frame adjunct that refers to the day before the day that includes the temporal perspective time. That is, the adjunct specifies a *relation* between some external period of time (in case of deictic adjuncts such as "gisteren", the temporal perspective time) and the period of time that it introduces. Anaphoric adjuncts, which, as we already mentioned, we do not deal with in this paper, specify a relation between some other period of time and the period of time that they introduce.

We therefore generalise that for frame adjuncts, the meaning is this relation between a contextual parameter and the introduced period of time. To represent this, a tuple consisting of the two periods of time is associated with the adjunct and the relation between them is enforced by an axiom. This is the axiom that expresses the "gisteren" relation as explained above:

$$\begin{array}{l}\texttt{fol}\ \forall(A): \mathit{adjt\_word}(A,\mathit{gisteren})\\ \Rightarrow (\exists(P,Y,T): \mathit{day\_a}(Y)\ \&\ \mathit{day\_a}(T)\ \&\\ \quad \mathit{within}(P,T)\ \&\ \mathit{meets}(Y,T)\ \&\\ \mathit{time\_period}(A,\mathit{ta\_frame}(\mathit{deictic}(P),Y))).\end{array} \tag{25}$$

The *ta_frame* functor specifies that the adjunct is a frame adjunct. Its first argument, the contextual parameter, is a functor *deictic*, which indicates that its only argument (ultimately) refers to the temporal perspective time of some clause. Note that the axiom only requires the existence of this period P, which only at a later stage will be identified with some temporal perspective time, by an axiom shown later in this subsection. Independent frame adjuncts will have the constant *independent* instead of the *deictic* functor.

In general, each adjunct has some (unique) construct associated with it. This means that *time_period* is an open function of an adjunct. Since the different axioms specify what the function result is, it is left unspecified in the declaration of *time_period* as an open function.

$$\begin{array}{lcl}\mathit{adjunctoid}(A) & \leftarrow & \exists(L): \mathit{adjt\_word}(A,L).\\ \mathit{anything}(\_) & \leftarrow & \texttt{true}.\\ \multicolumn{3}{l}{\texttt{of}\ \mathit{time\_period}: \mathit{adjunctoid}(\_) \rightarrow \mathit{anything}(\_).}\end{array} \tag{26}$$

As you can see, we do not only associate a time period with adjuncts, but with anything that "looks" like an adjunct (*adjunctoid*), which is anything that occurs as a first argument to the *adjt_word* predicate. We will show the use of that later in this subsection.

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

$$\begin{array}{l}\texttt{fol}\ \forall(W,T,R,L):\\ \mathit{verb\_time}(W,\mathit{ta\_frame}(R,T))\ \&\ \mathit{loc}(\mathit{evt}(W),L)\\ \qquad \Rightarrow \mathit{within}(L,T).\end{array} \tag{27}$$

Here, again, we use an auxiliary predicate *verb_time*, which associates the time period information to the verb that is modified:

$$\begin{array}{l}\mathit{verb\_time}(W,P)\\ \quad \leftarrow\ \exists(A): \mathit{verb\_adjunct}(W,A)\ \&\\ \qquad \mathit{time\_period}(A,P).\end{array} \tag{28}$$

When it is a deictic adjunct that modifies a verb, we still have to require that the contextual parameter is in fact the temporal perspective time:

$$\begin{array}{l}\texttt{fol}\ \forall(V,A,K,P):\\ \mathit{verb\_time}(V,\mathit{ta\_frame}(\mathit{deictic}(A),K))\ \&\\ \quad \mathit{v\_ppp}(V,P) \Rightarrow A = P.\end{array} \tag{29}$$

The *time_period* 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 modelled as inclusion in a frame interval that immediately succeeds the frame interval of the complement. The contextual dependency is determined by the complement and therefore it is left unchanged.

$$\texttt{fol}\ \forall(A,B,C,F) : \mathit{adjunct}(A, \mathit{na}(B))\ \&\ \mathit{time\_period}(B, \mathit{ta\_frame}(C,F)) \Rightarrow (\exists(T) : \mathit{meets}(F,T)\ \&\ \mathit{time\_period}(A, \mathit{ta\_frame}(C,T))). \quad (30)$$

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

$$\begin{array}{lll} \mathit{adjt\_word}(\mathit{a1}, \mathit{gisteren}) & \leftarrow & \texttt{true}. \\ \mathit{adjt\_word}(\mathit{a2}, \mathit{na}(\mathit{a1})) & \leftarrow & \texttt{true}. \\ \mathit{s\_adjunct}(\mathit{s1}, \mathit{a2}) & \leftarrow & \texttt{true}. \end{array} \quad (31)$$

As for durational adjuncts, we can only represent so called in-adjuncts so far, and not the for-adjuncts. The cause of this is that for-adjuncts require the eventuality of the verb they modify to be atelic in itself, but combined with the for-adjunct, the eventuality would be telic. Now, one and the same eventuality can obviously not be both telic and atelic, so the adjunct would have to introduce a new eventuality and we do not handle such things yet.

Handling *in* is done in much the same way as *na*, except that there is a change in type instead of a change in interval, since the complement of *in* when used by itself is not an in-adjunct.

$$\texttt{fol}\ \forall(A,B,C,T) : \mathit{adjunct}(A, \mathit{in}(B))\ \&\ \mathit{time\_period}(B, \mathit{ta\_dur}(T)) \Rightarrow \mathit{time\_period}(A, \mathit{ta\_in}(T)). \quad (32)$$

An in-adjunct requires its corresponding eventuality to have taken at most as much time as specified by its complement.

$$\texttt{fol}\ \forall(W,T,L) : \mathit{verb\_time}(W, \mathit{ta\_in}(T))\ \&\ \mathit{evttime}(\mathit{evt}(W), L) \Rightarrow \mathit{telic}(\mathit{evt}(W))\ \&\ \mathit{within}(L,T). \quad (33)$$

This is again similar to the frame adjunct constraint, the difference being that it requires a telic eventuality and that, here, the eventuality time is constrained directly, instead of indirectly through the location time. The location time *could* of course be used here as well, since, for telic eventualities, the eventuality time is restricted to be within the location time, but there is no need to here.

An example of a complement of *in* would be *een uur* (an hour):

$$\texttt{fol}\ \forall(A) : \mathit{adjt\_word}(A, \mathit{een\_uur}) \Rightarrow (\exists(T) : \mathit{hour}(T)\ \&\ \mathit{time\_period}(A, \mathit{independent}, \mathit{ta\_dur}(T))). \quad (34)$$

Here, the *hour* predicate means that its argument has a duration of exactly one hour.

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. As a trivial example, take the definition of *transposed* in section 20. Suppose we somehow observe *transposed*(*evt*(*w1*)), then we can explain this by assuming (abducing) *abnormal*(*ab_transposed*(*evt*(*w1*))). This form of reasoning is called non-monotonic, because new information may invalidate previously drawn conclusions. (This in contrast to deduction, where any conclusion that was drawn, can always still be drawn after the addition of new information.) For example, should we add the axiom

$$\texttt{fol}\ \neg \mathit{abnormal}(\mathit{ab\_transposed}(\mathit{evt}(\mathit{w1}))). \quad (35)$$

then we would not be able to draw the same conclusion, as it would be inconsistent. (This is not a problem for deduction, since we can deduce anything from an inconsistent theory.)

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 abducing 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 labelled, 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:

```
adjunct_verb : [adjunct_verb(a1,w1)]
evttime :
[evttime(evt(w1),int(ts(1999,1,1,0),
                        ts(1999,1,2,0))),
 evttime(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)))]
time_period : [time_period(a1,
 ta_frame(deictic(
  int(ts(1999,1,2,0),ts(1999,1,2,1))),
  int(ts(1999,1,1,0),ts(1999,1,2,0))))]
token_verb : [token_verb(w2,t_zijn),
              token_verb(w1,v_zijn)]
```

The *w1* and *w2* tokens were abduced to be tokens for the verb *v_zijn* and the temporal auxiliary *t_zijn*, respectively. As explained in section 4.4 this is in fact the only possibility for the *token_verb* predicate for this particular sentence. The only adjunct (*a1*) in the sentence was taken to modify the main verb (*w1*), which is again the only possibility here, since the temporal auxiliary is vacuous. This model is unique upto the choice of the intervals. In case of ambiuguities, several models are generated.

Since no information was given about the time, the time intervals are chosen rather arbitrarily, although they do of course satisfy the constraints. The temporal perspective time is 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 temporal perspective time is further the deictic argument relative to which "gisteren" (yesterday) is positioned. This January the first 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).

Suppose 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:

$$
\begin{array}{ll}
 & \mathit{utt}(U)\ \&\ \mathit{hour}(U)\ \& \\
(36) & \mathit{evttime}(\mathit{evt}(\mathit{w1}), \mathit{int}(\mathit{ts}(2000,5,21,18) \\
 & \qquad \mathit{ts}(2000,5,21,20)))
\end{array}
$$

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

The result is, as you would expect, that the utterance happened on May the 22nd. We only show the result for the *evt*time 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* ?

(37) $$utt(U)\ \&\ evttime(evt(w1),T)\ \&\ before(U,T)$$

The result:

```
no
```

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 extensible, 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 towards 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*, Plenum Press, pp. 293–322.

Denecker, M. (2000). Extending classical logic with inductive definitions, *in* C. Baral and M. Truszczynski (eds), *Proceedings of NMR'2000*, 8th Intl. Workshop on Non-Monotonic Reasoning, pp. 1–15.

Denecker, M. and Van Nuffelen, B. (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 Reyle, U. (1993). *From Discourse to Logic*, Kluwer Academic Publishers.

Pollard, C. and Sag, I. A. (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.

Richter, F., Sailer, M. and Penn, G. (1999). A Formal Interpretation of Relations and Quantification in HPSG, *in* G. Bouma, E. Hinrichs, G.-J. M. Kruijff and R. T. Oehrle (eds), *Constraints and Resources in Natural Language Syntax and Semantics*, CSLI Publications, pp. 281–298.

Schelkens, N., Verdoolaege, S. and Van Eynde, F. (2000). The semantics of temporal adjuncts, this volume.

Van Eynde, F. (1998). Tense, Aspect and Negation, *in* F. Van Eynde and P. Schmidt (eds), *Linguistic Specifications for Typed Feature Structure Formalisms*, European Communities.

Van Eynde, F. (2000). 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*, Rodopi, Amsterdam.

Van Eynde, F. (n.d.). A constraint-based semantics for tenses and temporal auxiliaries, Submitted to HPSG99 proceedings.