Time, Tense and Aspect in Natural Language Database Interfaces*

Ion Androutsopoulos[1] Graeme Ritchie[2] Peter Thanisch[3]

[1]Language Technology Group, Microsoft Research Institute,
Macquarie University, Sydney NSW 2109, Australia
e-mail: `ion@mri.mq.edu.au`

[2]Department of Artificial Intelligence, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, Scotland, U.K.
e-mail: `G.D.Ritchie@ed.ac.uk`

[3]Department of Computer Science, University of Edinburgh
King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, Scotland, U.K.
e-mail: `pt@dcs.ed.ac.uk`

*This paper reports on work that was carried out while the first author was in the Department of Artificial Intelligence, University of Edinburgh, supported by the Greek State Scholarships Foundation.

Abstract

Most existing natural language database interfaces (NLDBs) were designed to be used with database systems that provide very limited facilities for manipulating time-dependent data, and they do not support adequately temporal linguistic mechanisms (verb tenses, temporal adverbials, temporal subordinate clauses, etc.). The database community is becoming increasingly interested in temporal database systems, that are intended to store and manipulate in a principled manner information not only about the present, but also about the past and future. When interfacing to temporal databases, supporting temporal linguistic mechanisms becomes crucial.

We present a framework for constructing natural language interfaces for temporal databases (NLTDBs), that draws on research in tense and aspect theories, temporal logics, and temporal databases. The framework consists of a temporal intermediate representation language, called TOP, an HPSG grammar that maps a wide range of questions involving temporal mechanisms to appropriate TOP expressions, and a provably correct method for translating from TOP to TSQL2, TSQL2 being a recently proposed temporal extension of the SQL database language. This framework was employed to implement a prototype NLTDB.

1 Introduction

Since the 1960s, natural language database interfaces (NLDBs) have been the subject of much attention in the natural language processing community (Perrault and Grosz 1988), (Copestake and Sparck Jones 1990), (Androutsopoulos, Ritchie, and Thanisch 1995b). NLDBs allow users to access information stored in databases by formulating requests in natural language. Most existing NLDBs were designed to interface to database systems that provide very limited facilities for manipulating time-dependent data. Consequently, most NLDBs also provide very limited temporal support. In particular, they are intended to answer questions that make only a superficial use of time, in the form of timestamp values that are just treated as a form of numerical data by the database language, and do not adequately support temporal linguistic mechanisms (verb tenses and aspects, temporal adverbials, temporal subordinate clauses, etc.): users are allowed to use very few (if any) of these mechanisms, and their semantics are typically over-simplified or even ignored.

The database community is becoming increasingly interested in *temporal* database systems. These are intended to store and manipulate in a principled manner information not only about the present, but also about the past and future (Tansel, Clifford, Gadia, Jajodia, Segev, and Snodgrass 1993), (Tsotras and Kumar 1996). In questions directed to temporal databases (e.g. (1) – (3)), it becomes crucial for NLDBs to interpret correctly temporal linguistic expressions.

(1) What was the salary of each engineer while ScotCorp was building bridge 5?

(2) Did anybody leave site 4 before the chief engineer had inspected the control room?

(3) Which systems did the chief engineer inspect on Monday after the auxiliary generator was in operation?

Temporal database systems currently exist only in the form of research prototypes (e.g. Boehlen (1995)), but we expect that commercially available temporal database systems will appear in the near future, because: (i) data storage is constantly becoming cheaper, and hence retaining huge quantities of information about the past is becoming more cost effective, (ii) there is a strong demand for tools to store and manipulate time-dependent data (e.g. in stock market and data mining applications), and (iii) research on temporal databases is maturing, to the extent that consensus proposals are beginning to appear (Jensen 1993), (Snodgrass 1995). It is, therefore, desirable to investigate how interfaces to temporal database systems can be built.

Typically, NLDBs have used an intermediate representation language (usually some form of logic) to encode the meanings of natural language requests, with the resulting intermediate language expressions being available for translation into a suitable database language, often SQL (Melton and Simon 1993). (The benefits of this include generality, modularity, portability (Androutsopoulos *et al.* 1995b).) Building on this architecture, we propose a framework for constructing NLDBs for temporal databases (NLTDBs) that draws on ideas from tense and aspect theories (see Comrie (1976, 1985) for an introduction), temporal logics (van Benthem 1983),(Gabbay, Hodkinson, and Reynolds 1994), and temporal databases. We emphasise that we do not set out to formulate an improved general tense and aspect theory or temporal logic. Our aim is a more practical one: to explore how ideas from these areas can be integrated in a framework that leads to practical NLTDBs. We also concentrate on keyboard-entered English questions that refer to the past or the present, ignoring the following more difficult issues: questions about the future (that may require more complex semantic models),

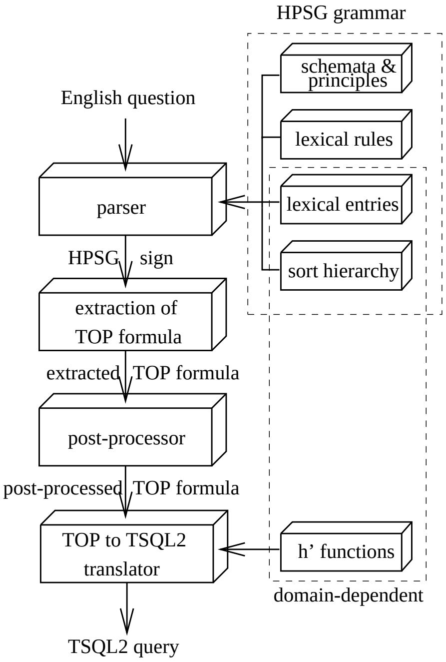

Figure 1: Architecture of the prototype NLTDB

speech input (although our work is in principle compatible with the text having been spoken), and requests to update the database (see Davidson and Kaplan (1983) for an outline of the difficulties with updates).

Ignoring some details, our framework consists of a formal temporal intermediate representation language, called TOP, an HPSG grammar (Pollard and Sag 1994) that maps a wide range of English questions involving temporal mechanisms to appropriate TOP expressions, and a provably correct mapping to translate from TOP to TSQL2, TSQL2 being a recently proposed extension of SQL (Snodgrass 1995). To demonstrate that this framework is workable, we have employed it to implement a small prototype NLTDB, using ALE (Carpenter 1992; Carpenter and Penn 1994) and Prolog.[1] Figure 1 shows the architecture of that system.

Each English question is first parsed using the HPSG grammar, generating an HPSG *sign* (a feature-structure that represents syntactic and semantic properties of the question; signs and the components of the HPSG grammar

[1]The prototype NLTDB is available from `http://www.dai.ed.ac.uk/groups/nlp`.

will be discussed in section 4). Multiple signs are generated for questions that the parser understands to be ambiguous. A TOP formula is then extracted from each sign and the extracted formula undergoes an additional post-processing phase (to be discussed in section 4.8).

The TOP formulae that are generated at the end of the post-processing capture what the NLTDB understands to be the possible readings of the English question. (The prototype NLTDB makes no attempt to determine which reading the user has in mind.) These formulae are then translated into TSQL2. (The h' functions of figure 1 map primitive TOP expressions to TSQL2 expressions. They will be discussed in section 6.1.)

The prototype NLTDB prints all the resulting TSQL2 queries and the corresponding TOP formulae.

The generated TSQL2 queries would be executed by the underlying database management system (DBMS) to retrieve the information requested by the user. A prototype database system, called TIMEDB, that supports a version of TSQL2 now exists.[2] During the work that we report here, however, TIMEDB was not yet available. As a result of this, our prototype NLTDB has not been linked to TIMEDB. Hence, the TSQL2 queries are currently not executed, and no answers are produced. We hope that future work will link the two systems. This task is complicated by the fact that the two systems adopt different versions of TSQL2, since we had to introduce some modifications to TSQL2 (see section 5), and TIMEDB also introduces some modifications.

The lexicon entries and sort hierarchy of the grammar, and the definitions of the h' functions of the TOP to TSQL2 translator are the only parts of the prototype NLTDB that need to be revised when configuring the NLTDB for a new database.

The parser, the grammar, and the formula extractor were implemented using ALE (Carpenter 1992; Carpenter and Penn 1994). ALE provides a chart-parser (which is the one used in the prototype NLTDB), and a formalism for expressing typed feature-structures, Prolog-like rules, and unification-based grammars.[3] The post-processor and the TOP to TSQL2 translator

[2] TIMEDB was developed by Andreas Steiner at ETH Zürich. It is available from `http://www.cs.auc.dk/ general/DBS/tdb/TimeCenter`. TIMEDB compiles TSQL2 statements into sequences of conventional SQL statements, which are then executed by a commercial database system (ORACLE).

[3] The ALE grammar of the prototype NLTDB is based on previous ALE encodings of HPSG grammars by Penn and Carpenter at Carnegie Mellon University, and Manandhar and Grover at the University of Edinburgh. We are grateful to all four of them. ALE is available from `http://macduff.andrew.cmu.edu/ale`.

were implemented in Prolog.

We emphasise that the prototype NLTDB is intended only to demonstrate that our theoretical framework is implementable. Several modules would have to be added to the prototype NLTDB, if it were to be used in real-life applications (see section 8).

The remainder of this paper is organised as follows: section 2 discusses some ideas from tense and aspect theories that are used in our framework; section 3 describes TOP, our intermediate representation language; section 4 presents the HPSG-based mapping from English to TOP; section 5 offers a brief introduction to TSQL2; section 6 highlights the TOP to TSQL2 mapping; section 7 compares our approach to previous proposals on NLTDBs; section 8 concludes and proposes directions for further research.

Although several issues (summarised in section 8) remain to be addressed, we argue that our framework constitutes a significant improvement over previous proposals on NLTDBs.

2 Aspectual classes

It is common in tense and aspect theories to classify natural language expressions or situations described by natural language expressions into *aspectual classes* (or "aktionsarten"), a practice often attributed to Vendler (1967). Numerous aspectual taxonomies have been proposed (see, for example, Dowty (1986), Moens and Steedman (1988), Mourelatos (1978), Parsons (1990), Vlach (1993)). These differ in the number and names of the classes, and the nature of the classified objects. What is common to all of them is that they try to capture intuitions about the way that native speakers view situations, with respect to issues such as whether the situation is instantaneous or more drawn out, whether the situation involves an event which is completed or not, etc. Although most of this work has been motivated by the goal of capturing linguistic generalisations, these nuances of meaning are also relevant to ensuring that a NLTDB operates in a correct and comprehensible manner.

2.1 Our aspectual taxonomy

For our purposes, we have found a taxonomy that distinguishes between *states*, *points*, *culminating activities*, and *activities* to be adequate. Our four classes correspond to ways of viewing situations that people seem to use when communicating in English: a situation can be viewed as involving no action (state view), as an instantaneous action (point view), as an action with a *climax* (an inherent completion point; culminating activity view), or as an action with no climax (activity view). (The use of "culminating activities" is derived from Moens and Steedman (1988).) For example, *"Tank 5 contains oil."* is typically uttered with a state view in mind, while *"Tank 5 exploded."* would be used in most contexts with a point view. *"ScotCorp built a new bridge."* would be typically uttered with a culminating activity view. In this case, the building situation is seen as having an inherent completion point (the point where the entire bridge has been built). If the building stops before reaching this point, the building situation is considered incomplete. When a verb is used in the simple past in a sentence uttered with a culminating activity view, the speaker usually implies that the climax was reached (the building of the bridge was completed). In contrast, *"Robot 2 moved."* would probably be uttered with an activity view. In this case, there is no climax (the movement can stop at any point without being any more or less complete), and the simple past carries no implication that any climax was reached. If, however, a particular destination were provided, as in *"Robot 2 moved to the exit."*, the sentence would be uttered with a culminating activity view, and the climax would be the point where the robot reaches the exit.

Determining which view the speaker has in mind is important to understand what the speaker means. For example, when an *"at ... "* temporal adverbial is attached to a clause uttered with a state view (e.g. *"Tank 5 contained oil at 5:00pm."*), the speaker usually means that the situation of the clause simply holds at the adverbial's time. Excluding special contexts that do not arise in NLDBs, there is normally no implication that the situation starts or stops holding at the adverbial's time. In contrast, with clauses uttered with culminating activity views, *"at ... "* adverbials (if felicitous) usually specify the time where the situation starts or reaches its climax (e.g. (4)).

(4) Robot 2 moved to the exit at 5:00pm.

Some linguistic markers seem to signal which view the speaker has in mind. For example, the progressive usually signals a state view (e.g. unlike (4), (5) is typically uttered with a state view, and the movement is simply ongoing at 5:00pm).

(5) Robot 2 was moving to the exit at 5:00pm.

Often, however, there are no such explicit markers. The processes employed in those cases by hearers to determine the speaker's view are not yet fully understood. In a NLTDB, however, where questions refer to a restricted domain, reasonable guesses can be made by observing that in each domain, each verb tends to be associated mainly with one particular view. Certain agreements about how situations are to be viewed (e.g. that some situations are to be treated as instantaneous – point view) will have also been made during the design of the database. These agreements provide additional clues about the main view of each verb in the domain.

More precisely, we adopt the following approach. Whenever the NLTDB is configured for a new application domain, the base form of each verb is assigned to one of the four aspectual classes, using criteria (to be discussed below) intended to determine the main view of each verb in the particular domain. Aspectual class is treated as a property not only of verbs, but also of verb phrases, clauses, and sentences. Normally, all verb forms inherit the aspectual classes of the corresponding base forms. Verb phrases, clauses, and sentences normally inherit the aspectual classes of their main verb forms. Some linguistic mechanisms (e.g. the progressive), however, may cause the aspectual class of a verb to differ from that of the base form, or the aspectual class of a verb phrase, clause, or sentence to differ from that of its main verb form.

In the case of a verb like *"to run"*, that typically involves a culminating activity view when used with an expression determining a specific destination or distance (e.g. *"to run to the station/five miles"*), but an activity view otherwise, we assume that there are two different homonymous verbs *"to run"*: one has a culminating activity base form, and requires a complement determining a specific destination or distance; the other has an activity base form and requires no such complement. A similar distinction is introduced with verbs whose aspectual class depends on whether the verb's object denotes a countable or mass entity (e.g. *"to drink a bottle of wine"* vs. *"to drink wine"*) (Mourelatos 1978). Similarly, when a verb can be used in an application domain with both habitual and non-habitual meanings (e.g. *"BA737 departs from Gatwick."* (habitually, perhaps every day) vs. *"BA737* (actually) *departed from Gatwick five minutes ago."*), we distinguish between homonyms with habitual and non-habitual meanings. Our aspectual criteria classify the base forms of habitual homonyms as states. The aspectual classes of non-habitual homonyms depend on the verb and the domain.

In the rest of this paper, we call verbs whose *base* forms are classified as states, activities, culminating activities, and points, *state, activity, culminating activity, and point verbs* respectively. This does not rule out other (non-base) forms of these verbs belonging to other classes.

2.2 Aspectual criteria

It is important to realise that our aspectual classes play a purely formal role, in the sense that their whole purpose is to allow the systematic translation of English into our semantic representation, in a way that achieves plausible effects within a NLTDB. Despite our attempts, in section 2.1 above, to provide an informal meaning for our classes, they are not accessible to linguistic intuition. Verbs are located in the classification scheme using particular criteria, which we outline below, not by considering whether or not they appear to meet some semantic condition suggested by the class name. For example, we have found it convenient to group "habitual" usage with "states", but this does not embody an empirical claim that "habitual" sentences do not describe actions.

The criteria we use to classify the base verb forms in each application domain are three, and they are applied sequentially. First, the *simple present criterion* is used: if the simple present of a verb can be used in the particular domain in single-clause questions with non-futurate meanings, it is a state verb; otherwise, it is a point, activity, or culminating activity verb. For example, in domains where (6) is possible, *"to contain"* is a state verb.

(6) Does any tank contain oil?

The simple present is sometimes used to refer to scheduled situations. For example, (7) could refer to a particular scheduled assembling. We consider this meaning of (7) a futurate one. Hence, (7) does not constitute evidence that *"to assemble"* is a state verb.

(7) When does J.Adams assemble engine 5?

As explained in section 2.1, when verbs have both habitual and non-habitual uses, we distinguish between habitual and non-habitual homonyms. Excluding scheduled meanings, (8) can only have a habitual meaning, i.e. it can only involve the habitual homonym of *"to land"*. Therefore, in domains where (8) is possible, the habitual *"to land"* is a state verb. (8) does not constitute evidence that the non-habitual *"to land"* is a state verb.

(8) Which flight lands on runway 2?

Next, the *point criterion* is applied to distinguish point verbs from activity or culminating activity verbs. The point criterion is based on the fact that some verbs describe kinds of situations that are always modelled in the database as instantaneous. If a verb describes situations of this kind, it is a point verb; otherwise it is an activity or culminating activity verb.

Many of our illustrative examples come from a hypothetical airport domain (described in Androutsopoulos (1996)). In that domain, we assume the database does not distinguish between the times at which a flight starts or stops entering an airspace sector: entering a sector is modelled as instantaneous. Also, in our airport domain *"to enter"* is used only to refer to flights entering sectors. Consequently, in that domain *"to enter"* is a point verb. (If *"to enter"* were also used to refer to groups of passengers entering planes, and if situations of this kind were modelled as non-instantaneous, one would have to distinguish two homonymous *"to enter"*: one used with flights entering sectors, and one with groups of passengers entering planes. Only the the first homonym would be a point verb.)

Finally, the *imperfective paradox criterion* is applied to decide if the remaining verbs are activity or culminating activity ones. The criterion is based on the "imperfective paradox" (Dowty 1977), (Lascarides 1988). Yes/no questions containing the past continuous and the simple past of the verbs (e.g. (9) – (12)) are considered. (The past continuous questions must not be read with futurate meanings, e.g. (9) must not be taken to ask if John was *going* to run.) If an affirmative answer to the past continuous question implies an affirmative answer to the simple past question (as in (9) – (10)), the verb is an activity verb; otherwise (as in (11) – (12)), it is a culminating activity verb. ((11) does not imply (12): John may have abandoned the building before completing the house.)

(9) Was John running?

(10) Did John run?

(11) Was John building a house?

(12) Did John build a house?

We show below how some base verb forms are classified in our airport domain.

states: contain, be (non-auxiliary), arrive (habitually), depart (habitually), service (habitually)

activities: circle (waiting permission to land), taxi (no destination), queue (for runway)

culminating activities: land, take off, inspect, board, service (actually), taxi (to destination)

points: enter, start, begin, stop, finish, leave, arrive (actually), depart (actually)

We stress that the classification depends on the uses of the verbs in the particular domain, and on how the situations of the verbs are modelled in the database. For example, our hypothetical airport database models arrivals and departures as instantaneous, and hence the corresponding verbs are classified as point verbs. If arrivals and departures were not modelled as instantaneous, the corresponding verbs would probably be classified as culminating activity ones.

3 The TOP language

We now turn to TOP, our intermediate representation language. Although we avoid referring to it as a "logic" (on the grounds that it has no proof theory), it is in many respects similar to, and closely based on, traditional logics, and it has a model theory much as one would define for a genuine temporal logic. However, it is far from being a full logic, as the emphasis in its design has been on reflecting natural language semantics rather than covering inferential phenomena. A formal summary of TOP is given in the appendix; although this is rather terse owing to space limitations, a fuller account can be found in of Androutsopoulos (1996, chapter 3).

TOP contains constants and variables, and atomic formulae are constructed by applying predicate symbols to arguments. Formulae can be conjoined, but there are currently no negation or disjunction connectives. (We did not need these for the linguistic phenomena that we focused on. We plan to include these in future TOP versions.) Similarly, there are currently no quantifier symbols, with variables essentially being treated as existentially quantified. The most relevant constructs for the current discussion are the temporal operators (which were influenced by those of Crouch and Pulman (1993)). For example, (13) is represented as (14). Roughly speaking, the *Past* operator (introduced by the verb tense) requires $contain(tk2, water)$ to be true at some past time e^v (the "^{v}" suffix marks variables), and the *At* operator (introduced by the adverbial) requires that time to fall within *1/10/95*. (Dates are shown in the day/month/year format.)

(13) Did tank 2 contain water (some time) on 1/10/95?

(14) $At[\mathit{1/10/95}, Past[e^v, contain(tk2, water)]]$

As we are not primarily interested in the structure of referring expressions, we will make the simplifying assumption that particular items in the domain such as *"tank 2"* are associated with simple constants such as $tk2$ rather than some more complex logical expression. The prototype NLTDB currently requires *"tank 2"* to be typed as a single word, which is associated in the lexicon with $tk2$. Notice that $tk2$ is a TOP constant, *not* a database value.

3.1 TOP basics

TOP assumes that time is linear, discrete, and bounded (van Benthem 1983). Following the Reichenbachian tradition (Reichenbach 1947), TOP formulae are evaluated with respect to more than one times: st (*speech time*, the time where the question is submitted to the NLTDB), et (*event time*, the time where the situation denoted by the formula holds), and lt (*localisation time*, a temporal window within which the event time must be placed (cf. Crouch and Pulman (1993); this is different from Reichenbach's "reference time" and similar to the "location time" of Kamp and Reyle (1993)). While st is always a time-point, et and lt are generally periods, and hence TOP can be considered period-based. (To avoid conflicts with TSQL2 terminology, we use the term "period" to refer to what temporal logicians usually call "intervals", i.e. convex sets of time-points.)

Although the aspectual classes of natural language expressions affect how these expressions are represented in TOP, it is not always possible to tell the aspectual class of a natural language expression by examining its TOP representation. The approach here is different from those of Dowty (1977), Dowty (1986), Kent (1993), and Lascarides (1988), where aspectual class is a property of formulae or denotations of formulae.

Returning to (13), the answer will be affirmative if (14) evaluates to true. When evaluating TOP formulae, lt initially covers the entire time axis, though it may be narrowed down subsequently by temporal operators. In (14), the At operator narrows lt to its intersection with 1/10/95. This causes lt to become 1/10/95. The $Past$ operator then narrows lt to its intersection with $[t_{first}, st)$, where t_{first} is the earliest time-point. (In this particular example, if 1/10/95 is entirely in the past, the $Past$ operator does not restrict lt any further.) The overall formula evaluates to true iff there is an event time period et, where $contain(tk2, water)$ is true, such that $et \sqsubseteq lt$. (p_1 is a subperiod of p_2, written $p_1 \sqsubseteq p_2$, iff p_1, p_2 are periods and $p_1 \subseteq p_2$.)

Although here we provide only informal descriptions of the semantics of TOP formulae, TOP has formally defined model-theoretic semantics (see Appendix A and Androutsopoulos (1996)). Among other things, a TOP model (which is ultimately defined in terms of database constructs), specifies the maximal event time periods where the situations denoted by TOP predicates hold (e.g. in the case of $contain(tk2, water)$, it specifies the maximal periods where tank 2 contained water). TOP predicates are always *homogeneous*. This means that if a predicate is true at an event time et, it is also true at any event time $et' \sqsubseteq et$. (Various versions of homogeneity have been used in Allen (1984), Kent (1993), Lascarides (1988), Richards, Bethke, van der Does, and Oberlander (1989), and elsewhere.) In our example, if tank 2 contained water from 29/9/95 to 2/10/95, $contain(tk2, water)$ is true at any event time that is a subperiod of that period. Hence, there will be an et where $contain(tk2, water)$ is true, such that et is also a subperiod of 1/10/95 (the lt), and the answer to (13) will be affirmative.

3.2 Interrogative quantifiers

In (14), the semantics of the *Past* operator bind the variable e^v to et (the event time, where tank 2 contained water). The e^v argument of the *Past* operator is useful, for example, in time-asking questions like (15), expressed as (16). The interrogative quantifier ($?_{mxl}$) of (16) reports the *maximal* past ets where $contain(tk2, water)$ is true, such that et is a subperiod of 1/10/95.

(15) When on 1/10/95 was tank 2 empty?

(16) $?_{mxl}e^v \ At[\mathit{1/10/95}, Past[e^v, empty(tk2)]]$

A simpler version of the interrogative quantifier is used in non-time asking questions like (17), expressed as (18).

(17) What did tank 2 contain on 1/10/95 at 5:00pm?

(18) $?w^v \ \ Part[\mathit{5\_00pm}^g, f^v] \wedge At[\mathit{1/10/95}, At[f^v, Past[e^v, contain(tk2, w^v)]]]$

The $\mathit{5\_00pm}^g$ in (18) denotes the set of all 5:00pm-periods (5:00:00-59 on 2/2/95, 5:00:00-59 on 3/2/95, etc.). The "^{g}" suffix stands for "gappy partitioning": the set of all 5:00pm-periods is gappy, in the sense that the union of its elements does not cover the entire time-axis. $Part[\mathit{5\_00pm}^g, f^v]$ means that f^v must be a 5:00pm-period. The two At operators cause lt to become the particular 5:00pm-period of 1/10/95. e^v, the event time where $contain(tk2, w^v)$ holds, must fall in the past and within f^v.

3.3 More on At, Before, and After

Two additional operators, *Before* and *After*, are used to express adverbials like "*before 1/10/95*" or "*after 5:00pm*". Like At, these operators narrow

lt. The first arguments of At, *Before*, and *After* can also be formulae. This form of the operators is used when expressing subordinate clauses introduced by "*while*", "*before*", or "*after*" respectively. In (20), which represents one possible reading of (19), the At operator causes $Past[e2^v, free(run2)]$ to be evaluated with respect to a new lt'. lt' is the intersection of the original lt (the whole time-axis) with a maximal event time period $e1^v$ where $Past[e1^v, circling(ba737)]$ is true. (20) evaluates to true iff there is a past event time $e2^v \sqsubseteq lt'$ where $free(run2)$ is true.

(19) Was runway 2 free while BA737 was circling?

(20) $At[Past[e1^v, circling(ba737)], Past[e2^v, free(run2)]]$

3.4 The Fills operator

(20) represents the reading of (19) whereby runway 2 must have been free *some* time while BA737 was circling. There is, however, a stricter reading, that requires runway 2 to have been free *throughout* BA737's circling. (Some speakers report another, preferred, reading in which the runway had to be free for *most* of the time that BA737 was circling. As a simplification, we will not handle that dialect variant, which would be semantically complicated.) Readings of this kind are very common when states combine with adverbials or clauses that are understood as specifying non-instantaneous periods. For example, (21) has a reading where tank 2 must have been empty *some* time on 1/10/95, and a stricter (and probably preferred) one where the tank must have been empty *throughout* that day.

(21) Was tank 2 empty on 1/10/95?

For simplicity, the stricter readings are currently ignored in our English to TOP mapping (and the rest of this paper). These readings, however, *can* be expressed in TOP using the *Fills* operator, which requires the event time to cover exactly the localisation time. For example, the stricter reading of (19) can be expressed as (22). The *Fills* operator of (22) requires $e2^v$ (event time where runway 2 was free) to be set to lt' (maximal past period where BA737 was circling). The stricter reading of (21) is expressed as (23).

(22) $At[Past[e1^v, circling(ba737)], Past[e2^v, Fills[free(run2)]]]$

(23) $At[\mathit{1/10/95}, Past[e^v, Fills[empty(tk2)]]]$

3.5 The Culm operator

In the case of activity, state, or point verbs (e.g. *"to circle"*, *"to contain"*, and *"to depart"* in our airport domain), progressive and non-progressive forms receive the same TOP representations. For example, both (24) and (25) are represented as (26), which requires an *et* where BA737 was circling to exist in the past. (Progressive forms of state verbs are typically unfelicitous, but for our purposes it is harmless to allow them. The use of point verbs in the progressive, as in (27), usually signals that the user is not aware that the situation of the verb is modelled as instantaneous. A warning to the user should be generated in the latter case. Our prototype NLTDB, however, currently does not generate such messages.)

(24) Was BA737 circling?

(25) Did BA737 circle?

(26) $Past[e^v, circling(ba737)]$

(27) Was BA737 departing at 5:00pm?

With culminating activity verbs (e.g. *"to inspect"* in the airport domain), the representation of the non-progressive contains an additional *Culm* operator. (28) and (30) are represented as (29) and (31) respectively. The *Culm* operator requires *et* to begin at the start-point of the earliest maximal period where the situation of its argument holds (the argument of *Culm* is always an atomic formula), and to end at the end-point of the latest of those maximal periods. Furthermore, the situation of the predicate must reach its climax at the end-point of *et*.

(28) Was J.Adams inspecting BA737?

(29) $Past[e^v, inspecting(ja, ba737)]$

(30) Did J.Adams inspect BA737?

(31) $Past[e^v, Culm[inspecting(ja, ba737)]]$

Let us assume, for example, that J.Adams started to inspect BA737 at 9:00pm on 1/10/95, stopped the inspection at 1:00pm, resumed it at 2:00pm, and finished it at 4:00pm on the same day. The maximal periods where the situation of $inspecting(ja, ba737)$ holds are from 9:00pm to 1:00pm and from 2:00pm to 4:00pm on 1/10/95. Let us call p_{mxl1} and p_{mxl2} these maximal periods. (29) requires an *et* where the predicate holds (i.e. a subperiod of

p_{mxl1} or p_{mxl2}) to be a subperiod of $[t_{first}, st)$ (the lt). That is, there must be some past time where J.Adams was inspecting BA737. (31), on the other hand, would, with the database values assumed here, require et to be the period from 9:00pm to 4:00pm on 1/10/95, the inspection to reach its climax at the end of et, and $et \sqsubseteq [t_{first}, st)$. That is, the entire inspection must be in the past. If the questions were submitted at 3:00pm on 1/10/95, (29) would be true, but (31) would be false. This accounts for the fact that (30) requires the inspection to have been completed, while (28) does not.

We note that our approach does not run into problems with the "imperfective paradox" ((31) implies (29), but not the opposite; cf. the logic of Crouch and Pulman (1993)), and this is achieved without invoking branching-time or possible worlds (cf. Dowty (1977), Kent (1993)), that are difficult to model in practical temporal database systems.

3.6 Episode identifiers

With culminating activity verbs, we often use an extra *episode-identifying* argument (the variable ep^v in (33); ep^v should also, for correctness, be present in (29) and (31)). This is needed in sentences like (32), if J.Adams was involved in several different inspections of BA737. (We treat assertions like (32) as yes/no questions; variables can be thought of, informally, as existentially quantified.)

(32) J.Adams inspected BA737 on 1/10/95.

(33) $At[\mathit{1/10/95}, Past[e^v, Culm[inspecting(ep^v, ja, ba737)]]]$

Let us assume, for example, that apart from the inspection of 1/10/95, J.Adams was also involved in another inspection of BA737, that started at 11:00pm on 1/1/95, and stopped (without ever reaching its completion) at 4:00am on 2/1/95. Let us call p_{mxl3} this period. We use the episode-identifying argument to distinguish between the two inspections: $inspecting(ep1, ja, ba737)$ is true for $et \sqsubseteq p_{mxl1}$ or $et \sqsubseteq p_{mxl2}$, while $inspecting(ep2, ja, ba737)$ is true for $et \sqsubseteq p_{mxl3}$.

The preferred reading of (32) is that an entire inspection (from start to completion) of BA737 by J.Adams occurred within 1/10/95. Without the episode-identifying argument, this reading is not captured correctly by (33): (33) would require et to extend from the beginning of p_{mxl3} (on 1/1/95) to the end of p_{mxl2} (on 1/10/95), and et to be a subperiod of 1/10/95. Since this is impossible, the answer would (incorrectly) be false. With ep^v, in contrast, (33) requires et to extend from the beginning of the earliest maximal period that corresponds to a particular value of ep^v ($ep1$ or $ep2$) to the end of the latest maximal period that corresponds to the *same* ep^v value. et must also be a subperiod of 1/10/95, and the inspection that corresponds to ep^v's value must reach its climax at the end of et. For $ep^v = ep2$, all these conditions are satisfied, and hence the answer would (correctly) be affirmative.

3.7 The For operator

Duration adverbials are represented using the *For* operator. For example, (34) is mapped to (35). day^c and $minute^c$ represent the sets of all day- and minute-periods respectively. (The "^{c}" suffix stands for "complete partitioning": the sets of all day- and minute-periods are complete, in the sense that the union of each set's elements is the entire time-axis.) The *For* operator of (35) requires et (the time of the circling) to cover 40 consecutive minutes.

(34) On which day was BA737 circling for forty minutes?

(35) $?d^v\ Part[day^c, d^v] \land At[d^v, For[min^c, 40, Past[e^v, circling(ba737)]]$

3.8 The Begin and End operators

Verbs like *"to start"*, *"to stop"*, and *"to finish"* are expressed using the *Begin* and *End* operators. (36) and (38) are mapped to (37) and (39) respectively. (37) requires the end-point of a maximal period where the inspection is ongoing to be located at some past 5:00pm-minute. (39) requires the *completion* of an inspection to fall within a past 5:00pm-minute.

(36) J.Adams stopped inspecting BA737 at 5:00pm.

(37) $Part[\mathit{5{:}00pm}^g, f^v] \land At[f^v, Past[e^v, End[inspecting(ep^v, ja, ba737)]]]$

(38) J.Adams finished inspecting BA737 at 5:00pm.

(39) $Part[\mathit{5{:}00pm}^g, f^v] \land At[f^v, Past[e^v, End[Culm[inspecting(ep^v, ja, ba737)]]]]$

3.9 The Pres and Perf operators

The past perfect is expressed using a *Past* followed by a *Perf* operator. For example, (40) is mapped to (41). The *Perf* operator introduces a new event time (denoted by $e2^v$ in (41)), that has to precede the original one ($e1^v$). (41) requires the time of the departure ($e2^v$) to precede another past time

($e1^v$). The latter corresponds to Reichenbach's *reference time* (Reichenbach 1947), a time from which the departure is viewed. With culminating activity verbs, a *Culm* operator is also inserted (e.g. (42) receives (43)).

(40) BA737 had departed.

(41) $Past[e1^v, Perf[e2^v, depart(ba737)]]$

(42) Had J.Adams inspected BA737?

(43) $Past[e1^v, Perf[e2^v, Culm[inspecting(ep^v, ja, ba737)]]]$

A sentence like (44) has two readings: *"on 1/1/95"* may specify the reference time, or it may refer directly to the inspection. (In the latter case, (44) is similar to *"Did J.Adams inspect BA737 on 1/1/95?"*, except that it creates an impression of a longer distance between the inspection and the present.) We represent the two readings as (45) and (46) respectively.

(44) Had J.Adams inspected BA737 on 1/1/95?

(45) $At[1/1/95, Past[e1^v, Perf[e2^v, Culm[inspecting(ep^v, ja, ba737)]]]]$

(46) $Past[e1^v, Perf[e2^v, At[1/1/95, Culm[inspecting(ep^v, ja, ba737)]]]]$

(45) requires the reference time ($e1^v$) to be a subperiod of 1/1/95, and to follow an entire inspection of BA737 by J.Adams. Apart from introducing a new *et*, the *Perf* operator also resets *lt* to the entire time-axis. Hence, in (45), $e2^v$ does not need to be a subperiod of 1/1/95. In (46), the *At* operator narrows *lt* after the *Perf* operator has reset it. This requires the inspection to be located within 1/1/95.

It has often been claimed (e.g. Blackburn, Gardent, and de Rijke (1994), Moens and Steedman (1988), Vlach (1993)) that the English present perfect asserts that some consequence of a past situation holds at the present. For example, (47) seems to imply that the engine is still on fire, or that it was damaged by the fire and has not been repaired. (A similar connection may exist between the past situation and the reference time in some uses of the past perfect.) This implication does not seem to be present (at least not as strongly) in (48).

(47) Engine 5 has caught fire.

(48) Engine 5 caught fire.

Although these claims are intuitively appealing, it is difficult to see how they could be used in a NLTDB. Perhaps in (49) the NLTDB should check not only that the landing was completed, but also that some consequence of the landing still holds. But what should this consequence be? Should the NLTDB check that the plane is still at the airport, and should the answer be negative if the plane has departed again? Should the NLDB check that the passengers are still at the airport? Given this uncertainty, we do not require the past situation to have present consequences.

(49) Has BA737 landed?

The present perfect could be expressed using a *Pres* and a *Perf* operator. (The *Pres* operator simply requires *st* to be located within *et*.) For example, (49) would be represented as (50). Given, however, that we do not require the past situation to have present consequences, there are very few differences between the present perfect and the simple past for our purposes. For simplicity, we represent the present perfect in the same way as the simple past, i.e. both (49) and (51) receive (52).

(50) $Pres[Perf[e^v, Culm[landing(ep^v, ba737)]]]$

(51) Did BA737 land?

(52) $Past[e^v, Culm[landing(ep^v, ba737)]]]$

3.10 The Ntense operator

We end the presentation of TOP with the *Ntense* operator (borrowed from Crouch and Pulman (1993)). This is useful in sentences like (53), where "president" may refer either to the present (the current president visited Rome, perhaps before becoming president), or to the time of the visit (the visitor was the president of that time). In (54), the *Ntense* operator requires p^v to be the president at *st* (indicated by now^*). In contrast, in (55) the *Ntense* requires p^v to be the president at the time of the visit (e^v).

(53) The president visited Rome in 1991.

(54) $Ntense[now^*, president(p^v)] \land At[y1991, Past[e^v, visiting(p^v, rome)]]$

(55) $Ntense[e^v, president(p^v)] \land At[y1991, Past[e^v, visiting(p^v, rome)]]$

We discuss (53) further in section 4.6.

4 From English to TOP

The English requests are mapped to TOP formulae using an HPSG-based grammar (Pollard and Sag 1994). An HPSG grammar consists of *schemata*, *principles*, *lexical rules*, *lexicon entries*, and a *sort hierarchy*. Our grammar is very close to that of Pollard and Sag, with the main difference being that we use TOP expressions instead of situation-theoretic constructs. We have also added an ASPECT feature and principle to accommodate our aspectual taxonomy, and lexical signs and rules for verb forms, temporal adverbials, and other temporal mechanisms.

This section attempts to offer a taste of how our HPSG-based mapping from English to TOP works (the full details can be found in Androutsopoulos (1996)). Readers with previous exposure to modern unification-based grammars should be able to follow most of the discussion. Some of the details, however, may be unclear to readers not familiar with HPSG.

4.1 Verb forms

In HPSG, each word and syntactic constituent receives a *sign*, a feature-structure of a particular form, that provides information about the word or syntactic constituent. (56) shows the (lexical) sign of the base form of *"to inspect"* in the airport domain. (To save space, we ignore some uninteresting features, and we write "SS" instead of "SYNSEM".)

(56)
$$
\left[\begin{array}{ll}
\text{PHON} & \langle \textit{inspect} \rangle \\
\text{SS} \mid \text{LOC} \mid \text{CAT} & \left[\begin{array}{ll}
\text{HEAD} & {}_{verb}\left[\begin{array}{ll}\text{VFORM} & bse \\ \text{AUX} & - \end{array}\right] \\
\text{ASPECT} & culmact \\
\text{SUBJ} & \langle \text{NP}_{\boxed{1}} \rangle \\
\text{COMPS} & \langle \text{NP}_{\boxed{2}} \rangle
\end{array}\right] \\
\text{SS} \mid \text{CONT} & inspecting(ep^v, \boxed{1}, \boxed{2})
\end{array}\right]
$$

According to (56), *"inspect"* is the base form of a non-auxiliary verb, it is a culminating activity, and it requires a noun-phrase as its subject, and one as its object. The base form introduces a TOP expression of the form $inspecting(ep^v, \boxed{1}, \boxed{2})$, where $\boxed{1}$ and $\boxed{2}$ correspond to the entities denoted by the subject and object respectively. (More precisely, $\boxed{1}$ and $\boxed{2}$ stand for HPSG *indices*. In our HPSG version, indices represent TOP constants and variables. Our indices have a special feature, not shown here, that distinguishes constant- from variable-representing indices.) The CONT values are actually feature-structures that represent TOP expressions. For simplicity, however, here we show directly the corresponding TOP expressions.

The signs for non-base forms of non-auxiliary verbs are generated automatically from the corresponding base form signs by lexical rules. For example, the simple past (57) is generated from (56) by a lexical rule that inserts a *Past* and a *Culm* in the CONT of the base sign. The lexical rule also modifies PHON for simple past morphology, and changes VFORM to *fin* (finite form, a form that requires no auxiliary verb). The simple past signs of state, point, and activity verbs are generated by a similar rule, which does not insert, however, a *Culm* operator.

$$(57)\quad \begin{bmatrix} \mathrm{PHON} & \langle \textit{inspected} \rangle \\ \mathrm{SS}\,|\,\mathrm{LOC}\,|\,\mathrm{CAT} & \begin{bmatrix} \mathrm{HEAD} & {}_{\textit{verb}}\begin{bmatrix}\mathrm{VFORM} & \textit{fin} \\ \mathrm{AUX} & - \end{bmatrix} \\ \mathrm{ASPECT} & \textit{culmact} \\ \mathrm{SUBJ} & \langle\ \mathrm{NP}_{\boxed{1}}\ \rangle \\ \mathrm{COMPS} & \langle\ \mathrm{NP}_{\boxed{2}}\ \rangle \end{bmatrix} \\ \mathrm{SS}\,|\,\mathrm{CONT} & Past[e^v, Culm[inspecting(ep^v, \boxed{1}, \boxed{2})]] \end{bmatrix}$$

When *"inspected"* combines with its subject and object in (58), the CONT value of (57) becomes (59), a value which is inherited by the sign of the overall sentence. Roughly speaking, the TOP representation of a sentence is the CONT value of the sentence's sign. (We will refine this statement in the following paragraphs.)

(58) J.Adams inspected BA737.

(59) $Past[e^v, Culm[inspecting(ep^v, ja, ba737)]]$

Present participle signs are the same as the corresponding base ones, except that their VFORM is *prp*. In (60), for example, apart from its VFORM, the sign for *"inspecting"* is the same as (56) (i.e. *"inspecting"* is also a culminating activity form). (61) shows the sign of *"was"* that is used in past continuous forms.

(60) J.Adams was inspecting BA737.

(61)
$$\begin{bmatrix} \text{PHON} & \langle \textit{was} \rangle \\ \text{SS} \mid \text{LOC} \mid \text{CAT} & \begin{bmatrix} \text{HEAD} & {}_{\textit{verb}}\begin{bmatrix}\text{VFORM} & \textit{fin} \\ \text{AUX} & + \end{bmatrix} \\ \text{ASPECT} & \textit{state} \\ \text{SUBJ} & \langle \boxed{1} \rangle \\ \text{COMPS} & \langle \text{VP}\begin{bmatrix}\text{SUBJ} & \langle\boxed{1}\rangle \\ \text{VFORM} & \textit{prp}\end{bmatrix} : \boxed{2} \rangle \end{bmatrix} \\ \text{SS} \mid \text{CONT} & \textit{Past}[e^v, \boxed{2}] \end{bmatrix}$$

According to (61), *"was"* requires a present participle verb phrase complement (e.g. *"inspecting BA737"*). The subject of *"was"* ($\boxed{1}$) is the same as the subject of the complement. The CONT of *"was"* is also the same as the CONT of the verb phrase complement ($\boxed{2}$), but with an additional *Past* operator. In (60), when *"was"* combines with its complement (*"inspecting BA737"*) and its subject (*"J.Adams"*), the CONT of (61) becomes (62). The sign of the overall (60) inherits the CONT of (61) (i.e. (60) is mapped to (62)).

(62) $Past[e^v, inspecting(ep^v, ja, ba737)]$

The propagation of the ASPECT feature is controlled by our ASPECT principle: syntactic constituents inherit the ASPECT of their head-daughter (the *"was"* in *"was inspecting"*), except if the head-daughter combines with an adjunct-daughter (a modifier), in which case the mother syntactic constituent inherits the ASPECT of the adjunct-daughter (an example of the latter case will be given in section 4.3). In (60), the *"was inspecting"* (and then the overall sentence) inherits the *state* ASPECT of *"was"*. In effect, the progressive has caused an aspectual shift to state (*"inspecting"* was a culminating activity). Along with our treatment of *"at ... "* temporal adverbials (discussed in section 4.3 below), this shift accounts for the fact that in *"J.Adams was inspecting BA737 at 5:00pm."* the inspection was probably simply ongoing at 5:00pm (cf. *"J.Adams inspected BA737 at 5:00pm."*). A similar shift is assumed in Dowty (1986), Moens and Steedman (1988), Vlach (1993), and elsewhere.

4.2 Habituals and non-habituals

In the case of verbs with multiple homonyms (section 2), the lexicon provides multiple signs for the base forms. (63) and (64), for example, correspond to the habitual and non-habitual homonyms respectively of *"to service"*. (In our airport domain, each flight is habitually (normally) serviced by particular servicing companies, though sometimes a flight may be serviced by a company other than its normal one.)

(63) $$\begin{bmatrix} \textsc{phon} & \langle \textit{service} \rangle \\ \textsc{ss} \mid \textsc{loc} \mid \textsc{cat} & \begin{bmatrix} \textsc{head} & {}_{\textit{verb}}\begin{bmatrix} \textsc{vform} & \textit{bse} \\ \textsc{aux} & - \end{bmatrix} \\ \textsc{aspect} & \textit{state} \\ \textsc{subj} & \langle \textsc{np}_{\boxed{1}} \rangle \\ \textsc{comps} & \langle \textsc{np}_{\boxed{2}} \rangle \end{bmatrix} \\ \textsc{ss} \mid \textsc{cont} & \textit{hab\_serv}(\boxed{1}, \boxed{2}) \end{bmatrix}$$

(64) $$\begin{bmatrix} \textsc{phon} & \langle \textit{service} \rangle \\ \textsc{ss} \mid \textsc{loc} \mid \textsc{cat} & \begin{bmatrix} \textsc{head} & {}_{\textit{verb}}\begin{bmatrix} \textsc{vform} & \textit{bse} \\ \textsc{aux} & - \end{bmatrix} \\ \textsc{aspect} & \textit{culmact} \\ \textsc{subj} & \langle \textsc{np}_{\boxed{1}} \rangle \\ \textsc{comps} & \langle \textsc{np}_{\boxed{2}} \rangle \end{bmatrix} \\ \textsc{ss} \mid \textsc{cont} & \textit{servicing}(ep^v, \boxed{1}, \boxed{2}) \end{bmatrix}$$

(63) classifies the habitual homonym as a state verb, and introduces a predicate form $hab\_serv(\boxed{1}, \boxed{2})$, which is intended to hold at times where $\boxed{1}$ (the servicer) has the habit of servicing $\boxed{2}$ (the flight). (64) classifies the non-habitual homonym as a culminating activity verb, and introduces a predicate of the form $servicing(ep^v, \boxed{1}, \boxed{2})$, which is intended to hold at times where $\boxed{1}$ is actually servicing $\boxed{2}$.

Simple present signs are generated by a lexical rule that requires the base form to be a state. Thus, no simple present signs are generated for activity, culminating activity, or point verbs. This reflects the fact that (ignoring futurate meanings, which we do not address) these verbs are unlikely to be used in the simple present in a NLDB context. For example, it is unlikely that (65) would refer to a currently ongoing service (i.e. with a reading that involves the non-habitual culminating activity verb); the present continuous would have been used instead. (65) typically has a habitual meaning (that involves the habitual state verb).

(65) Which company services BA737?

In (65), this arrangement assigns only a habitual sign to *"services"* (the habitual sign is similar to (63), with an additional *Pres* operator). This causes (65) to receive only (66), which asks for the current habitual servicer of BA737.

(66) $?c^v\ Ntense[now^*, company(c^v)] \wedge Pres[hab\_serv(c^v, ba737)]$

In contrast, the simple past lexical rules generate both habitual and non-habitual simple past signs (derived from (63) and (64)). Hence, (67) receives both (68) and (69). These correspond to two possible readings of (67): (68) asks for the 1990 habitual servicers of BA737, while (69) asks for companies that actually serviced (possible just once) BA737 in 1990. (The role of the *Ntense* operators will be discussed in section 4.6.)

(67) Which companies serviced BA737 in 1990?

(68) $?c^v\ Ntense[e^v, company(c^v)] \wedge At[y1990,\ Past[e^v, hab\_serv(c^v, ba737)]]$

(69) $?c^v\ Ntense[e^v, company(c^v)] \wedge At[y1990, Past[e^v, Culm[servicing(ep^v, c^v, ba737)]]]$

4.3 Temporal adverbials

Let us now consider the adverbial of (70). The lexicon provides multiple signs for prepositions that introduce temporal adverbials. (71) shows the "*at*" sign that is used when an "*at* … " adverbial modifies a state or point expression. (The HEAD of (71) is shown separately in (72).)

(70) J.Adams was inspecting BA737 at 5:00pm.

(71)
$$\begin{bmatrix} \textsc{phon} & \langle at \rangle \\ \textsc{ss} \mid \textsc{loc} \mid \textsc{cat} & \begin{bmatrix} \textsc{head} & (72) \\ \textsc{subj} & \langle\rangle \\ \textsc{comps} & \langle \textsc{np}_{\ min\_per\boxed{1}} \rangle \\ \textsc{aspect} & point \end{bmatrix} \\ \textsc{ss} \mid \textsc{cont} & At[\boxed{1}, \boxed{2}] \end{bmatrix}$$

(72)
$$\begin{bmatrix} \textsc{mod} \mid \textsc{loc} \mid \textsc{cat} \mid \textsc{aspect} & state \vee point \\ \textsc{mod} & \textsc{s}\begin{bmatrix}\textsc{vform}\ fin\end{bmatrix}{:}\boxed{2} \vee \textsc{vp}\begin{bmatrix}\textsc{vform}\ psp\end{bmatrix}{:}\boxed{2} \end{bmatrix}_{prep}$$

According to (71), "*at*" requires as its complement a noun phrase that denotes a minute-period.[4] Requiring the complement of (71) to denote a

[4]For readers familiar with HPSG, *min_per* is a subsort of *index*. In our HPSG version, indices are divided into subsorts that correspond to types of world entities. These subsorts are used to enforce selectional restrictions.

minute-period ensures that (71) will not be used with a noun phrase complement like *"Monday"* or *"gate 2"*. (There are other *"at"* signs for uses of *"at"* in sentences like *"BA737 arrived at gate 2"*.) The SS|LOC|CAT|ASPECT of (71) is the aspectual class of the modified expression *after* attaching the *"at ... "* adverbial. This will be discussed further below.

(72) (the HEAD of (71)) shows that (71) can be used when *"at"* introduces a modifier of a state or point expression. It also requires the expression being modified to be a finite sentence (a finite verb form that has combined with its complements and subject) or a past-participle (*psp*) verb phrase. We generally require temporal adverbials (e.g. *"at 5:00pm"*, *"on Monday"*) to modify whole finite sentences (e.g. *"J.Adams inspected BA737"*) rather than finite verb phrases (e.g. *"inspected BA737"*), because in HPSG the latter approach would lead to problems with questions like (73): in (73), *"was"* combines in one step with both its subject *"J.Adams"* and its complement *"BA737"*, following the head-subject-complement schema of Pollard and Sag (1994); hence, there would be no verb phrase constituent (verb that has combined with its complements but not its subject) to be modified by *"at 5:00pm"*.

(73) Was J.Adams inspecting BA737 at 5:00pm?

Temporal adverbials are also allowed to modify past-participle verb phrases (see (72)). This is needed to be able to generate both readings of past perfect sentences like (44). We discuss this in following paragraphs.

When *"at"* combines with *"5:00pm"* in (70), *"at 5:00pm"* receives (74).

$$(74)\quad \begin{bmatrix} \textsc{phon} & \langle \textit{at, 5:00pm} \rangle \\ \textsc{ss} \mid \textsc{loc} \mid \textsc{cat} & \begin{bmatrix} \textsc{head} & (72) \\ \textsc{subj} & \langle\rangle \\ \textsc{comps} & \langle\rangle \\ \textsc{aspect} & \textit{point} \end{bmatrix} \\ \textsc{ss} \mid \textsc{cont} & At[\boxed{1}, \boxed{2}] \\ \textsc{qstore} & \left\{ Part[\textit{5:00pm}^g, \boxed{1}] \right\} \end{bmatrix}$$

The QSTORE (quantifier storage) of (74) shows that the TOP variable represented by $\boxed{1}$ is existentially quantified over all 5:00pm-periods. (QSTORE is intuitively a set containing quantifiers along with their restrictions. Currently, all TOP variables are implicitly existentially quantified. This is why no explicit existential quantifier is shown in (74).)

The *"at 5:00pm"* then attaches to *"J.Adams was inspecting BA737"*, and (70) receives (75). The principles of HPSG cause (75) to inherit the HEAD of (61), and the CONT and QSTORE of (74). Our ASPECT principle (section 4.1) also causes (75) to inherit the point ASPECT of (74). In effect, the point-specifying adverbial has caused the originally state sentence to become point. (This aspectual shift is needed to block some impossible readings in sentences with multiple temporal adverbials; see Androutsopoulos (1996).)

(75)
$$\begin{bmatrix} \text{PHON} & \langle \textit{J.Adams.,was,inspecting,BA737,at,5:00pm} \rangle \\ \text{SS} \mid \text{LOC} \mid \text{CAT} & \begin{bmatrix} \text{HEAD} & {}_{\textit{verb}}\begin{bmatrix}\text{VFORM} & \textit{fin} \\ \text{AUX} & + \end{bmatrix} \\ \text{ASPECT} & \textit{point} \\ \text{SUBJ} & \langle\rangle \\ \text{COMPS} & \langle\rangle \end{bmatrix} \\ \text{SS} \mid \text{CONT} & \begin{array}{l} At[\boxed{1}, \\ Past[e^v, inspecting(ep^v, ja, ba737)]] \end{array} \\ \text{QSTORE} & \left\{ Part[\textit{5:00pm}^g, \boxed{1}] \right\} \end{bmatrix}$$

To obtain the TOP translation of (70), the expressions in the QSTORE of (75) have to be added in front of the CONT formula. (The variable-representing $\boxed{1}$ also has to be replaced by an actual TOP variable.) This leads to (76), which requires the inspection to have been ongoing at some 5:00pm-minute. (The reader is reminded that assertions are treated as yes/no questions. (73) receives the same formula.)

(76) $Part[\textit{5:00pm}^g, f^v] \land At[f^v, Past[e^v, inspecting(ep^v, ja, ba737)]]$

These examples also demonstrate an important area in which our framework needs to be extended. In (73) (or (70)) the user would typically have a particular 5:00pm-minute in mind (e.g. the 5:00pm-minute of the current or a previously mentioned day; see section 5.5.1 of Kamp and Reyle (1993)). A temporal anaphora resolution mechanism would be needed to determine that particular minute from the context, but our framework currently provides no such mechanism. The answer to (73) would therefore be affirmative if J.Adams was inspecting BA737 at *any* past 5:00pm-minute. A similar mechanism is also needed to cope with the anaphoric nature of tenses (Webber 1988). (77), for example, is typically a request to report flights that were circling at a particular contextually salient past time. In contrast, our framework currently generates an answer that reports all the flights that were circling at any past time.

(77) Which flights were circling?

Let us now consider the case where an *"at ... "* adverbial attaches to a culminating activity. The intended meaning is usually that the situation either starts or reaches its completion at the time of the adverbial. The first reading is the preferred one in (78). The second reading is the preferred one in (79).

(78) J.Adams inspected BA737 at 5:00pm.

(79) BA737 landed at 5:00pm.

There is an *"at"* sign for each reading. (There are also additional signs for cases where *"at ... "* adverbials modify activities. Cases where *"at ... "* adverbials modify states or points are covered by (71).) (81) is the sign for the reading where the situation of the culminating activity *starts* at the adverbial's time. The HEAD of (80) is shown in (81).

(80) $$\left[\begin{array}{ll} \text{PHON} & \langle \textit{at} \rangle \\ \text{SS} \mid \text{LOC} \mid \text{CAT} & \left[\begin{array}{ll} \text{HEAD} & (81) \\ \text{SUBJ} & \langle\rangle \\ \text{COMPS} & \langle \text{NP}_{min\_per\boxed{1}} \rangle \\ \text{ASPECT} & \textit{point} \end{array}\right] \\ \text{SS} \mid \text{CONT} & At[\boxed{1}, Begin[\boxed{2}]] \end{array}\right]$$

(81) $${}_{prep}\left[\begin{array}{ll} \text{MOD} \mid \text{LOC} \mid \text{CAT} \mid \text{ASPECT} & \textit{culmact} \\ \text{MOD} & \text{S}\left[\text{VFORM } \textit{fin}\right]{:}\boxed{2} \lor \text{VP}\left[\text{VFORM } \textit{psp}\right]{:}\boxed{2} \end{array}\right]$$

Along with (57), (80) causes (78) to be mapped to (82). (82) requires the period that covers the entire inspection (from start to completion) to be located in the past, and its start-point to fall within a 5:00pm-minute.

(82) $Part[5{:}00pm^g, f^v] \land At[f^v, Begin[\ Past[e^v, Culm[inspecting(ep^v, ja, ba737)]]]]$

The sign for the reading where the situation reaches its completion at the adverbial's time is the same as (80), except that it introduces an *End* rather than a *Begin* operator. That sign leads to a second formula for (78), that contains an *End* instead of a *Begin*. That formula corresponds to an unlikely reading in the case of (78), but it would capture the preferred reading in (79). A complete NLDB would typically paraphrase both formulae, and ask the user to select the intended one (see De Roeck and Lowden (1986) for related discussion). Our prototype NLTDB simply attempts to generate a separate answer for each formula.

We now explain briefly the reason for allowing temporal adverbials to modify not only finite sentences, but also past-participle verb phrases. In (44) (see section 3.9 above), this allows *"on 1/1/95"* to attach either to *"had J.Adams inspected BA737"* or to *"inspected BA737"*. That is, (44) is taken to have two possible parses, sketched in (83) and (84). These give rise to (45) and (46) respectively, that express the possible readings of (44).

(83) [Had J.A. [inspected BA737]] on 1/1/95?

(84) Had J.A. [[inspected BA737] on 1/1/95]?

Like *"at ... "* adverbials, *"on ... "* adverbials introduce At operators. The $Past$ and $Perf$ operators of (45) and (46) are introduced by the *"had"* auxiliary, while the $Culm$ operator is introduced by the lexical entry of the past participle *"inspected"*. In (84), the adverbial attaches to *"inspected BA737"* before *"had"*. Hence, in (46) the At ends up within the $Past$ and $Perf$. In (83), in contrast, the *"had"* attaches to the verb phrase first. This causes the At in (45) to have wider scope over the $Past$ and $Perf$.

4.4 Fronted temporal modifiers

When temporal modifiers attach to sentences, we allow them to either follow or precede the sentences they modify. This admits both (85) and (86). Both sentences receive (87). (An alternative approach would be to allow temporal modifiers to participate in unbounded dependencies; see pp. 176 – 181 of Pollard and Sag (1994).)

(85) Tank 2 was empty on Monday.

(86) On Monday tank 2 was empty.

(87) $Part[monday^g, m^v] \land At[m^v, Past[e^v, empty(tk2)]]$

When temporal modifiers attach to past-participle verb phrases, we require the modifiers to follow the verb phrases, as in (88). This rules out unacceptable sentences like (89).

(88) BA737 had [[entered sector 2] at 5:00pm].

(89)*BA737 had at 5:00pm entered sector 2.

Our approach causes (90) to receive only (91) (where the adverbial specifies the reference time), because in (90) *"at 5:00pm"* can modify only *"BA737 had entered sector 2"*; it cannot modify just *"entered sector 2"* because of the intervening *"BA737 had"*.

(90) At 5:00pm BA737 had entered sector 2.

(91) $Part[\textit{5:00pm}^g, f^v] \wedge At[f^v,\ Past[e1^v, Perf[e2^v, enter(ba737, s2)]]]$

In contrast, (92) receives both (91) and (93), because in that case the adverbial can modify either *"BA737 had entered sector 2"* or *"entered sector 2"*. In (93) the adverbial specifies directly the time of the entrance.

(92) BA737 had entered sector 2 at 5:00pm.

(93) $Part[\textit{5:00pm}^g, f^v] \wedge Past[e1^v,\ Perf[e2^v, At[f^v, enter(ba737, s2)]]]$

The fact that (90) does not receive (93) does not seem to be a disadvantage, because in (90) the reading where the adverbial specifies directly the time of the entrance seems unlikely.

There is, however, a complication with our approach: when a sentence is modified by both a preceding and a trailing temporal modifier (as in (94)), two parses are generated: one where the trailing modifier attaches first to the sentence, and one where the preceding modifier attaches first. In (94), this generates two semantically equivalent formulae, (95) and (96). Our framework currently provides no mechanism to eliminate one of the two.

(94) On 1/7/95 BA737 was at gate 2 for two hours.

(95) $At[\textit{1/7/95}, For[hour^c, 2,\ Past[e^v, location\_of(ba737, g2)]]]$

(96) $For[hour^c, 2, At[\textit{1/7/95},\ Past[e^v, location\_of(ba737, g2)]]]$

4.5 Temporal subordinate clauses

Like temporal adverbials, temporal subordinate clauses are treated as modifiers of finite sentences or past-participle verb phrases. In (97), for example, *"while runway 2 was open"* is taken to modify *"did any flight circle"*. As with temporal adverbials, when the modifier attaches to a sentence, it is allowed to either precede or follow the sentence. This admits both (97) and (98).

(97) Did any flight circle while runway 2 was open?

(98) While runway 2 was open, did any flight circle?

Space does not permit a detailed presentation of our treatment of temporal subordinate clauses. We note only that the words that introduce the subordinate clauses (e.g. *"while"*, *"before"*) receive several signs in the lexicon, that are sensitive to the aspectual classes of both the main and the subordinate clause. There are, for example, two *"after"* signs for the case where the main clause is a point (e.g. *"which flights departed"* in the airport domain) and the *"after ... "* clause is a state (e.g. *"system 32 was in operation"*). The first sign introduces an At operator, whose first argument is the TOP representation of the subordinate clause, and the second argument is the TOP representation of the main clause. The second sign is similar, but it also introduces a *Begin* operator in front of the formula of the subordinate clause. These two signs cause (99) to receive (100) and (101).

(99) Which flights departed after system 32 was in operation?

(100) $?f^v\ flight(f^v) \land\ After[Past[e1^v, in\_oper(sys32)], Past[e2^v, depart(f^v)]]$

(101) $?f^v\ flight(f^v) \land\ After[Begin[Past[e1^v, in\_oper(sys32)]], Past[e2^v, depart(f^v)]]$

(99) is ambiguous between a reading where the departures must have happened after the *beginning* of a maximal period where system 32 was in operation, and a reading where the departures must have happened after the *end* of such a maximal period. The two readings are captured by (100) and (101) respectively.

4.6 Nominals and temporal reference

Following a distinction introduced in Pollard and Sag (1987), we use different signs for predicative and non-predicative uses of nouns (e.g. *"president"* is predicative in (102), but non-predicative in (104)). Predicative noun signs contribute predicates ($president(w^v)$ in (103)) that always end up within the operators of the verb's tense (the *Past* in (103)). (There is a special sign for the non-auxiliary *"was"* in (102). This introduces a *Past* operator, and inserts the predicate of *"president"* into the second argument-slot of the *Past*.) This arrangement captures the fact that predicative nouns always refer to the time of the verb tense (e^v in (103), the event time that has to fall within 1991).

(102) Who was the president in 1991?

(103) $?w^v\ At[y1991, Past[e^v, president(w^v)]]$

(104) The president visited Rome in 1991.

In contrast, non-predicative nouns can generally refer either to the time of the verb tense, or to the speech time. As already discussed in section 3.10, for example, the *"president"* of (104) may refer to the person who was the president at the time of the visit, or to the present president. Non-predicative nouns can actually refer to any contextually salient time (see Dalrymple (1988), Enc (1986), and Hinrichs (1988)), but limiting these salient times to *st* and the time of the verb tense is a reasonable simplification at this stage of development of NLTDBs, since a more thorough treatment would require a complete system of temporal anaphora.

To account for the possible meanings of non-predicative nouns, non-predicative noun signs place the nouns' predicates within *Ntense* operators (e.g. $Ntense[t^v, president(p^v)]$). These predicates (along with the *Ntense*s) are inserted into the quantifier storage, and at the end of the parsing they are added in front of the rest of the formula, outside the operators of the verb tenses. This causes (104) to receive (105).

(105) $Ntense[t^v, president(p^v)] \land \ At[y1991, Past[e^v, visiting(p^v, rome)]]$

In the formulae that are generated at the end of the parsing, the *Ntense*s require the nouns' predicates to hold at unspecified times (t^v in (105)). During a subsequent post-processing phase (to be discussed in section 4.8), each *Ntense* generally leads to multiple formulae ((54) and (55) in the case of (105)), where the first argument of the *Ntense* has been replaced by now^* or a variable that occurs as a first argument of a *Past* or *Perf* operator. These formulae capture the readings where the noun refers to the speech time or the time of the verb tense respectively. (In past perfect sentences, the "time of the verb tense" can be either the reference time or the time of the past situation.) In present sentences, like (106) that is initially mapped to (107), this arrangement correctly generates only one formula, where the first argument of the *Ntense* is now^* (current president; in this case there is no *Past* or *Perf* operator).

(106) The president is visiting Rome.

(107) $Ntense[t^v, president(p^v)] \land \ Pres[visiting(p^v, rome)]$

This strategy has the disadvantage that it leads to a multiplicity of formulae in past sentences with several non-predicative nouns. Our prototype NLTDB uses a rather ad hoc mechanism to reduce drastically the number of generated formulae, that is based on the observation that with many nouns, satisfactory answers can be obtained by assuming that they refer always to *st*, or always to the time of the verb tense. We do not discuss this mechanism here (see Androutsopoulos (1996)).

4.7 Interrogatives

The interrogatives *"who"* and *"what"* are treated syntactically as noun phrases. For example, (108) receives the same syntactic analysis as (110). (We ignore punctuation.) The only difference from ordinary noun phrases is that *"who"* and *"what"* introduce interrogative quantifiers. (108) and (110) receive (109) and (111) respectively.

(108) Who was inspecting BA737?

(109) $?w^v \; Past[inspecting(ep^v, w^v, ba737)]$

(110) J.Adams was inspecting BA737.

(111) $Past[inspecting(ep^v, ja, ba737)]$

Similarly, the interrogative *"which"* is treated in the same way as other noun-phrase determiners (e.g. "a" in (114)), except that it introduces an interrogative quantifier. (112) and (114) receive (113) and (115) respectively.

(112) Which tank is empty?

(113) $?k^v \; Ntense[now^*, tank(k^v)] \; \wedge Pres[empty(k^v)]$

(114) A tank is empty.

(115) $Ntense[now^*, tank(k^v)] \wedge Pres[empty(k^v)]$

Our HPSG version also supports questions with multiple interrogatives, like (116) which is mapped to (117). (116) receives the same syntactic analysis as (110).

(116) Who was inspecting what.

(117) $?w1^v \; ?w2^v$
$Past[e^v, inspecting(ep^v, w1^v, w2^v)]$

Questions like (118), where the interrogative refers to the object of the verb, are treated using HPSG's unbounded-dependencies mechanisms (see Androutsopoulos (1996) and Pollard and Sag (1994) for details).

(118) Which flight did J.Adams inspect?

Finally, the *"when"* of (119) is treated syntactically as a temporal-adverbial modifier of finite sentences. That is, (119) receives the same syntactic analysis as (120).

(119) When was tank 2 empty?

(120) On 1/7/95 was tank 2 empty?

The *"when"* of (119) introduces a $?_{mxl}$ quantifier (see section 3.2). In the formula that is generated at the end of the parsing, the variable of this quantifier does not occur elsewhere in the formula (e.g. (119) initially receives (121)). During the post-processing phase (discussed in section 4.8 below), this variable is replaced by the first argument of a *Past* or *Perf* operator introduced by the main clause. In (121), this replaces t^v by e^v, leading to a formula that asks for the past maximal intervals where tank 2 was empty.

(121) $?_{mxl}t^v\ Past[e^v, empty(tk2)]$

4.8 Post-processing

Before being translated into TSQL2, the TOP formulae that are generated by the parsing undergo an additional post-processing phase. This is a collection of minor transformations that cannot be carried out easily during the parsing. Two of these transformations have already been discussed in sections 4.6 and 4.7: replacing the variables of *Ntense* operators and $?_{mxl}$ quantifiers with variables that occur as arguments of *Past* or *Perf* operators.

The third post-processing transformation is needed when *"for ... "* duration adverbials combine with verb forms that normally imply that a climax was reached. In these cases, the *"for ... "* adverbials (when felicitous) cancel the normal implication that the climax was reached. (122), for example, carries no implication that BA737 arrived at gate 2 (cf. (124)). The problem is that the parsing of (122) generates (123), which requires the taxiing to have been completed.

(122) BA737 taxied to gate 2 for five minutes.

(123) $For[minute^c, 5, Past[e^v, Culm[taxiing\_to(ep^v, ba737, g2)]]$

(124) BA737 taxied to gate 2.

Roughly speaking, to solve this problem during the post-processing any *Culm* operator that is within the scope of a *For* operator is removed. In (123), this results in a formula that no longer requires the taxiing to have been completed.

"In ... " duration adverbials also introduce *For* operators (e.g. parsing (125) generates (123)). *For* operators are actually marked with a flag that shows if the operator was introduced by a *"for ... "* or *"in ... "* adverbial,

and the post-processing removes only *Culm* operators from within *For* operators introduced by *"for ... "* adverbials. This distinction is necessary because unlike *"for ... "* adverbials, *"in ... "* duration adverbials do not cancel the implication that the climax was reached ((123) is the correct representation of (125)).

(125) BA737 taxied to gate 2 in five minutes.

Admittedly the post-processing compromises the compositionality of our English to TOP mapping. This, however, does not seem to have any practical disadvantages.

4.9 The linguistic coverage

We conclude the discussion of our English to TOP mapping with a list of temporal linguistic phenomena that our framework can or cannot handle. There is a wealth of temporal mechanisms in English (and most natural languages), and it would have been impossible to consider all of them in the time that we had available. Instead, we targetted a carefully selected set of temporal mechanisms, which is both non-trivial and representative of many types of temporal phenomena. Of course, perfect grammatical/semantic descriptions are not achievable, even within non-computational projects, so we are well aware that our proposed analyses are no more than useful simplifications. Nevertheless, we believe that our framework can serve as a basis for further work that will extend its coverage of temporal phenomena (see also section 8).

Verb tenses: We support six tenses: simple present, simple past, present continuous, past continuous, present perfect (treated as simple past), and past perfect. We have not considered the past perfect continuous, future tenses, or futurate readings of present and past tenses (e.g. the futurate reading of the simple present – see section 2.2). We do not expect adding more tenses to be particularly difficult.

Temporal verbs: Among special temporal verbs, we support *"to start"*, *"to begin"*, *"to stop"*, and *"to finish"*. We have not considered other special temporal verbs, like *"to happen"* or *"to follow"*.

Temporal nouns: Our framework supports nouns like *"year"*, *"month"*, *"day"*, etc. (and proper names like *"Monday"*, *"January"*, *"1/5/92"*). Nouns referring to more abstract temporal entities (e.g. *"period"*, *"event"*), and nouns that introduce situations (e.g. *"the construction of bridge 2"*) are not supported.

Temporal adjectives: No temporal adjectives (e.g. *"first"*, *"earliest"*) are supported, with the exception of *"current"*, that we use to restrict the times to which some nouns can refer (e.g. *"current president"*; see section 4.6).

Temporal adverbials: Our framework supports temporal location adverbials introduced by *"at"*, *"on"*, *"in"*, *"before"*, and *"after"* (e.g. *"at 5:00pm"*, *"on Monday"*, *"in January"*, *"before 5:00pm"*, *"after 1/1/95"*), as well as *"yesterday"*, *"today"*, and duration adverbials introduced by *"for"* and *"in"* (*"for two days"*, *"in two hours"*). Other adverbials that specify boundaries (e.g. *"since 5:00pm"*, *"until 1/1/95"*) should be easy to add. Frequency adverbials (e.g. *"twice"*, *"daily"*) seem more difficult to support.

Temporal clauses: Only temporal subordinate clauses introduced by *"while"*, *"before"*, and *"after"* are currently handled. Adding support for other boundary-specifying clauses (e.g. *"since J.Adams became manager"*, *"until tank 2 is empty"*) should not be difficult. *"When ..."* clauses are known to be the most difficult to support (see Moens and Steedman (1988) and Ritchie (1979) for related discussion).

Temporal anaphora: Among temporal anaphoric phenomena, only the anaphoric nature of nominals like *"the president"* is supported (section 4.6). As discussed in section 4.3, our framework currently provides no mechanism to determine the particular times to which adverbials like *"at 5:00pm"* or verb tenses refer.

5 The TSQL2 language

We now turn to TSQL2, the database language into which we translate the TOP formulae that the linguistic processing generates. TSQL2 (Snodgrass 1995) is a temporal extension of SQL (Melton and Simon 1993), the *de facto* standard database language. Traditional database languages such as SQL can and have been used to handle temporal information, leading some researchers to question the need for special temporal support in database languages (see Davies, Lazell, Hughes, and Cooper (1995) for related discussion). The lack of special temporal support in these languages, however, makes it difficult to manipulate time-dependent information. TSQL2, for example, adds to SQL a period datatype, and keywords to compute the intersection of two periods, to check if two periods overlap, etc. Without these, the rules that translate from TOP to database language (to be discussed in section 6) would be much more difficult to formulate, and much harder to comprehend.

We also note that until recently there was little consensus among temporal database researchers on how temporal support should be added to SQL (or other database languages), with every researcher essentially adopting his/her own temporal database language. Perhaps as a result of this, only a very few database management systems (DBMSs) that support these languages have been implemented (mostly early prototypes; see Boehlen (1995)). In contrast, TSQL2 was designed by a committee comprising most leading temporal database researchers, and hence it has a much better chance of being supported by (or at least influencing) forthcoming temporal database systems. As already mentioned in section 1, a prototype database system (called TIMEDB) that supports a version of TSQL2 has already appeared.

Like SQL, TSQL2 is designed to be used with the relational database model, where information is stored in *relations*, intuitively tables consisting of rows (called *tuples*) and columns (*attributes*). As an example of a TSQL2 relation, *tank_contents* shows the contents of various tanks over time.

tank_contents		
tank	*content*	
*tank*1	*oil*	{[1/1/95, 31/5/95], [1/9/95, 1/5/96]}
*tank*1	*empty*	{[1/6/95, 31/8/95]}
*tank*2	*water*	{[1/1/95, 31/5/95], [1/8/95, 31/8/95]}
*tank*2	*empty*	{[1/6/95, 31/7/95], [1/9/95, 1/5/96]}
*tank*3	*foam*	{[1/1/95, 1/5/96]}

tank_contents has two *explicit* attributes (*tank* and *content*), and an unnamed *time-stamping* attribute that shows when the information of each tuple was true (e.g. tank1 contained oil from 1/1/95 to 31/5/95, and from 1/9/95 to 1/5/96). We use a double line to separate the time-stamping attribute from the explicit ones. *tank_contents* is in a normalised form where there is no pair of tuples with the same explicit attribute values, and the time-stamps (the values of the time-stamping attribute) are sets of maximal periods. For simplicity, we assume in this example that time-stamps are specified at the granularity of days. In the airport domain, the time-stamps are actually specified at the granularity of minutes (e.g. the last time-stamp of *tank_contents* above would be something like {[8:32*am* 1/1/95, 4:16*pm* 1/5/96]}).

Information can be retrieved from the database using TSQL2 `SELECT` statements (these are temporally enhanced forms of the corresponding SQL statements). For example, (126) generates the relation in (127), that shows the tanks that were empty at any time during July 1995.

(126)
```
SELECT SNAPSHOT t1.tank
FROM tank_contents AS t1
WHERE t1.content = 'empty'
  AND VALID(t1) OVERLAPS PERIOD '[1/7/95 - 31/7/95]'
```

(127)

result of (126)
tank
*tank*1
*tank*2

The `t1` in the `FROM` clause of (126) is a *correlation name*. It can be thought of as a tuple-variable that ranges over the tuples of *tank_contents*. The `WHERE` clause allows `t1` to range over only tuples whose *content* is *empty*, and whose time-stamp overlaps the period from 1/7/95 to 31/7/95 (`VALID(t1)` refers to the time-stamp of the `t1`-tuple). Finally, the `SELECT` clause specifies that the result should be a *snapshot relation* (a relation with no time-stamping attribute), whose only attribute contains the *tank* values of the `t1`-tuples.

As a further example, (128) generates (129), where each time-stamp is a single maximal period where the corresponding tank was empty.

(128)
```
SELECT t1.tank
VALID VALID(t1)
FROM tank_contents(PERIOD) AS t1
WHERE t1.content = 'empty'
```

(129)

result of (128)	
tank	
*tank*1	[1/6/95, 31/8/95]
*tank*2	[1/6/95, 31/7/95]
*tank*2	[1/9/95, 1/5/96]

The `(PERIOD)` in the `FROM` clause of (128) generates an intermediate relation that is the same as *tank_contents*, except that there is a separate tuple for each period in the time-stamps of *tank_contents* (a tuple for *tank*1, *oil*, and [1/1/95, 31/5/95], a tuple for *tank*1, *oil*, and [1/9/95, 1/5/96], etc.). The `t1` of (128) ranges over tuples of the intermediate relation whose *content* is *empty*. The `SELECT` and `VALID` clauses specify that the resulting relation should have one explicit and one time-stamping attribute, and that the values of these attributes should be the *tank* values and time-stamps of the `t1`-tuples.

To simplify the mapping from TOP to TSQL2 (to be discussed in section 6), and to bypass some obscure points in the definition of TSQL2, we had to introduce some modifications in TSQL2.[5] These are relatively few and well-documented (in Androutsopoulos (1996)), to the extent that we are confident that they do not undermine the research value of our TOP to TSQL2 mapping. For example, we assume that attributes are ordered, and we refer to attributes by number rather than by name (e.g. we write (130) instead of (128)).

(130)
```
SELECT t1.1
VALID VALID(t1)
FROM tank_contents(PERIOD) AS t1
WHERE t1.2 = 'empty'
```

The most significant of our modifications is the addition of a `(SUBPERIOD)` keyword. This is intended to be used with relations like (129), where each time-stamp is a single maximal period. When applied to (129), `(SUBPERIOD)` generates (132), that apart from the tuples of (129) contains tuples for all the subperiods of the original time-stamps. The use of `(SUBPERIOD)` is illustrated in (131), where " ... (130) ... " stands for the statement of (130) (the current versions of SQL and TSQL2 allow embedded `SELECT` statements in the `FROM` clause). (131) generates a relation that contains all the tuples of (132) whose time-stamp is a subperiod of [4/6/95, 27/8/95].

(131)
```
SELECT t2.1
VALID VALID(t2)
FROM ( ... (130) ... )(SUBPERIOD) AS t2
WHERE PERIOD '[4/6/95 - 27/8/95]' CONTAINS VALID(t2)
```

[5] Our work also revealed a number of possible improvements to TSQL2. These were reported in Androutsopoulos, Ritchie, and Thanisch (1995a).

(132)

result of (130)	
tank	
*tank*1	[1/6/95, 31/8/95]
*tank*1	[3/6/95, 30/8/95]
*tank*1	[5/6/95, 26/8/95]
*tank*1	[1/7/95, 1/7/95]
...	...
*tank*2	[1/6/95, 31/7/95]
*tank*2	[4/6/95, 19/7/95]
...	...
*tank*2	[1/9/95, 1/5/96]
*tank*2	[3/9/95, 1/2/96]
...	...

It is important that an implementation of the `(SUBPERIOD)` construct does not involve explicitly computing and storing this entire relation, as this would involve vast amounts of space. The regular structure of the relation, and the very fact that it is computable from a formula, allows a database system to use a virtual relation, which behaves as if it contained all the many tuples of the entire relation, without the need for explicit representation of each one.

6 From TOP to TSQL2

We now discuss how TOP formulae are translated into TSQL2. Although a method to translate from a form of temporal logic to TSQL2 has already been proposed (Boehlen, Chomicki, Snodgrass, and Toman 1996), that method is not applicable to NLTDBs, because it is not coupled to a systematic mapping from natural language to logic, and because the nature of the logic of Boehlen *et al.* (1996) makes devising such a mapping very difficult (this is discussed further in Androutsopoulos (1996)).

6.1 Primitive TOP expressions

One central issue for any kind of NLDB is that of portability. Customising an NLDB or NLTDB to handle a new or different domain is bound to involve some human effort in defining how natural language expressions or lexical entries relate to symbolic structures in the database and/or expressions in the database query language. A well-designed system will limit the range of structures that have to be altered, and make it very clear what items must be edited and in what way. We have attempted to do this by specifying a well-defined set of functions (the h' functions of figure 1) which form this particular interface.

The person who configures the NLTDB for a new database has to specify how primitive TOP expressions like constants and predicate functors relate to TSQL2 expressions. In the case of TOP constants, the configurer has to specify a function h'_{cons} that maps each constant to a TSQL2 literal that represents the same entity in the world. For example, the TOP constant $tk2$ (used, for example, in (121)) represents the same tank as the attribute value $tank2$ in $tank\_contents$, and this attribute value is written in TSQL2 as `'tank2'`; i.e. $h'_{cons}(tk2)$ = `'tank2'`. Similarly, we use the TOP constant $y1991$ (e.g. in (105)) to refer to the time-period that is written in TSQL2 as `PERIOD '[1/1/91 - 31/12/91]'`; i.e. $h'_{cons}(y1991)$ = `PERIOD '[1/1/91 - 31/12/91]'`.

In the case of TOP predicate functors (e.g. *empty* in (121)), the configurer has to specify a function h'_{pfuns} that maps each functor to a TSQL2 `SELECT` statement.[6] That statement must generate a relation of the same form as (129) that shows for each combination of predicate arguments, the maximal periods where the situation of the predicate is true (i.e. $h'_{pfuns}(empty)$ would be the statement (128) that generates (129)).[7] With predicate functors introduced by culminating activity verbs (e.g. *inspecting* in (117)), the configurer specifies an additional `SELECT` statement. This generates a relation that shows for each combination of predicate arguments, whether or not the situation of the predicate has reached its climax. We do not discuss this here (see Androutsopoulos (1996)).

To some extent, the specification of h' functions can be supported by standard templates for the configurer to use, or by adopting rewrite rules that specify the values of the h' functions for whole classes of TOP expressions. In a particular domain, for example, it may be appropriate to use a rewrite rule stating that TOP constants containing underscores are mapped to multi-word TSQL2 strings by replacing underscores with spaces and capitalising the first letter of each word in the TSQL2 string (this sets h'_{cons} (united_kingdom) = `'United Kingdom'`). Rewrite rules of this kind can be easily incorporated into our prototype NLTDB, where the h' functions are implemented as Prolog rules.

[6] In Androutsopoulos (1996) h'_{pfuns} has an additional argument that corresponds to the arity of the predicate. For simplicity, we ignore this argument here.

[7] This assumes that each predicate can be mapped directly to a relation computed from the contents of the database. There are cases (e.g. the "doctor-on-board" problem (Rayner 1993)) where this is not possible, and an intermediate reasoning layer is needed. See Androutsopoulos (1996) for discussion on how this reasoning layer would fit into our framework.

6.2 The translation rules

The TOP formulae are translated into TSQL2 by a set of translation rules of the form:

$$trans(\phi, \lambda) = \Sigma$$

where ϕ is the formula to be translated, λ is a TSQL2 expression that intuitively represents what was the localisation time when ϕ was encountered (this will become clearer below), and Σ is the TSQL2 translation of ϕ (Σ is always a `SELECT` statement). There are: (i) basic (non-recursive) translation rules for the cases where ϕ is a atomic formula or a formula of the form $Culm[\phi']$, and (ii) recursive translation rules for more complex formulae, that translate ϕ by calling other translation rules to translate subformulae of ϕ.

For example, ignoring some details, the translation rule for the case where ϕ is an atomic formula $\pi(\tau_1, \ldots, \tau_n)$ is the following:

$trans(\pi(\tau_1, \ldots, \tau_n), \lambda) =$
`SELECT` $\alpha.1$, $\alpha.2$, ... , $\alpha.n$
`VALID VALID(`α`)`
`FROM (`$h'_{pfuns}(\pi)$`)(SUBPERIOD) AS` α
`WHERE` ...
  `AND` ...
  ⋮
  `AND` ...
  `AND` λ `CONTAINS VALID(`α`)`

Whenever the translation rule is used, α is a new correlation name (section 5). The " ... " in the `WHERE` clause are all the strings in $S_1 \cup S_2$:

$$\begin{aligned} S_1 &= \{\text{“}\alpha.i = h'_{cons}(\tau_i)\text{”} \mid i \in \{1, 2, 3, \ldots, n\} \\ &\qquad \text{and } \tau_i \text{ is a TOP constant}\} \\ S_2 &= \{\text{“}\alpha.i = \alpha.j\text{”} \mid i, j \in \{1, 2, 3, \ldots, n\}, \\ &\qquad \tau_i = \tau_j, \text{ and } \tau_i, \tau_j \text{ are TOP variables}\} \end{aligned}$$

Informally, S_2 contains restrictions that are needed when a variable appears at more than one predicate-argument positions. The S_2 restrictions ensure that the attributes of the resulting relation that correspond to these argument positions have the same values in each tuple.

Let us assume, for example, that we use the rule above to translate the formula $empty(tk^v)$, and that λ is set to the λ_1 of (133), where χ is the TSQL2 name of the finest available granularity (the level of *chronons* in

TSQL2 terminology). The `TIMESTAMP 'now' - INTERVAL '1'` χ below refers to the time-point immediately before st.

(133) λ_1 = `INTERSECT(`
```
      PERIOD(TIMESTAMP 'beginning',
             TIMESTAMP 'forever),
      PERIOD(TIMESTAMP 'beginning',
             TIMESTAMP 'now' - INTERVAL '1' χ))
```

λ_1 is the TSQL2 representation of $[t_{first},\ t_{last}] \cap [t_{first},\ st)$, where t_{first} and t_{last} are the earliest and latest time-points respectively (like TOP, TSQL2 assumes that time is discrete and bounded). $[t_{first},\ t_{last}] \cap [t_{first},\ st)$ is, of course, equal to $[t_{first},\ st)$. We use λ_1 here to make this example easier to integrate with later examples. The translation rule maps $empty(tk^v)$ to (134), where $h'_{pfuns}(empty)$ is (128). Assuming that 1/5/96 is in the past, (134) generates (132), that shows for all the possible values of tk^v, the past event-time periods where $empty(tk^v)$ is true that fall within lt (the period of λ_1). (The `WHERE` clause of (134) has no effect in this case, because $lt = [t_{first},\ st)$, and all the time-stamps of the `t2`-tuples are subperiods of lt.)

(134)
```
SELECT t2.1
VALID VALID(t2)
FROM (h'_pfuns(empty))(SUBPERIOD) AS t2
WHERE λ1 CONTAINS VALID(t2)
```

In this particular example, S_1 and S_2 are empty. If the predicate contained constants as arguments (e.g. $empty(tk2)$), S_1 would introduce restrictions in the `WHERE` clause that would cause the resulting relation to contain information only about the world entities of those constants. In $empty(tk2)$, for example, (134) would contain the additional restriction `t2.1 = 'tank2'` in its `WHERE` clause, which would remove from the result tuples that do not refer to tank 2.

Let us now consider the recursive translation rule for the case where ϕ has the form $Past[\beta, \phi']$ (β is a TOP variable, and ϕ' a formula):

$trans(Past[\beta, \phi'], \lambda) =$
```
SELECT VALID(α), α.1, α.2, ... , α.n
VALID VALID(α)
FROM (trans(φ', λ')) AS α
```

Every time the translation rule is used, α is a new correlation name. n is the number of explicit attributes of the relation generated by $trans(\phi', \lambda')$, and λ' is the TSQL2 expression:

```
INTERSECT(λ,PERIOD(TIMESTAMP 'beginning',TIMESTAMP 'now' - INTERVAL '1'
χ))
```

The translation rule for $Past[\beta, \phi']$ narrows the period of λ (intuitively lt) to its intersection with $[t_{first}, st)$ (this is in accordance with the semantics of the $Past$ operator; see section 3.1).

Let us assume, for example, that the rule above is used to translate $Past[e^v, empty(tk^v)]$, and that λ has been set to the λ_2 of (135) (that specifies a period that covers the entire time-axis). $Past[e^v, empty(tk^v)]$ would be mapped to (136), where $trans(empty(tk^v), \lambda_1)$ is (134).

(135) $\lambda_2 =$ `PERIOD(TIMESTAMP 'beginning', TIMESTAMP 'forever')`

(136)
```
SELECT VALID(t3), t3.1
VALID VALID(t3)
FROM (trans(empty(tk^v), λ1)) AS t3
```

The translation rules for formulae with no interrogative quantifiers (e.g. $Past[\beta, \phi']$) generate `SELECT` statements that return relations containing one explicit attribute for each predicate-argument and each variable that occurs as argument of a temporal operator. (136) returns (137), where the first and second explicit attributes correspond to e^v and tk^v respectively, and the time-stamps represent event-time periods where $Past[e^v, empty(tk^v)]$ holds, for the various combinations of e^v and tk^v values.

(137)

...	$tank1$	[1/6/95, 31/8/95]
...	$tank1$	[3/6/95, 30/8/95]
...	$tank1$	[5/6/95, 26/8/95]
...	$tank1$	[1/7/95, 1/7/95]
...	...	...
...	$tank2$	[1/6/95, 31/7/95]
...	$tank2$	[4/6/95, 19/7/95]
...	...	...
...	$tank2$	[1/9/95, 1/5/96]
...	$tank2$	[3/9/95, 1/2/96]
...	...	...

The `SELECT` clause of (136) specifies that (137) must have two explicit attributes, and that in each tuple the first explicit attribute (the one for e^v) must have the same value as the time-stamp. (To save space, we do not show the values of the first explicit attribute in (137).) This reflects the fact that in $Past[e^v, empty(tk^v)]$ the semantics of $Past$ requires the value of e^v to be the event time of $empty(tk^v)$ (section 3.2).

When translating from TOP to TSQL2, λ is initially set to the λ_2 of (135). (This reflects the fact that lt initially covers the whole time-axis – section 3.1.) For example, the TSQL2 translation of (138) (*"Was anything empty?"* (at any past time)) is $trans(Past[e^v, empty(tk^v)], \lambda_2)$, i.e. (136). In the case of yes/no questions, the answer is affirmative if the generated TSQL2 code returns a relation with at least one tuple, and negative otherwise. In (138), the generated TSQL2 code (i.e. (136)) returns (137), and hence the answer would be affirmative.

(138) $Past[e^v, empty(tk^v)]$

As a final example of a translation rule, we show below the rule for the case where ϕ has the form $?\beta_1\ ?\beta_2\ \ldots ?\beta_k\ \phi'$ ($\beta_1, \beta_2, \ldots, \beta_k$ are TOP variables, and ϕ' a formula). This type of formula is intended to reflect the meaning of *wh*-questions in English, with each $?\beta_i$ corresponding to a wh-word (*"what"*, *"which"*, etc.; see sections 3.2 and 4.7).

$trans(?\beta_1\ ?\beta_2\ \ldots\ ?\beta_k\ \phi', \lambda) =$
SELECT SNAPSHOT $\alpha.\omega_1$, $\alpha.\omega_2$, ... , $\alpha.\omega_k$
FROM $trans(\phi', \lambda)$ AS α

Whenever this rule is used, α is a new correlation name. ω_1, ω_2, ... , ω_k are numbers referring to the explicit attributes of the relation returned by $trans(\phi', \lambda)$. The ω_1-th, ω_2-th, ... , ω_k-th explicit attributes of that relation must correspond to β_1, β_2, ... , β_k respectively. (This correspondance is defined formally in Androutsopoulos (1996).)

For example, the TSQL2 translation of $\phi = ?tk^v\ Past[e^v, empty(tk^v)]$ is $trans(\phi, \lambda_2)$, which according to the translation rule above is (139). (The $trans(Past[e^v, empty(tk^v)], \lambda_2)$ in (139) is (136).)

(139) SELECT SNAPSHOT t4.2
FROM ($trans(Past[e^v, empty(tk^v)], \lambda_2)$) AS t4

(139) generates a one-attribute snapshot relation that lists all the tanks whose names appear in the second explicit attribute of (137) (i.e. tanks that were empty in the past).

6.3 Outline of the correctness proof

We have proven formally (in Androutsopoulos (1996)) the correctness of the TOP to TSQL2 translation rules. The full details of this proof are highly formal and voluminous, and hence beyond the scope of this paper. Here, we only attempt to provide an outline of our approach.

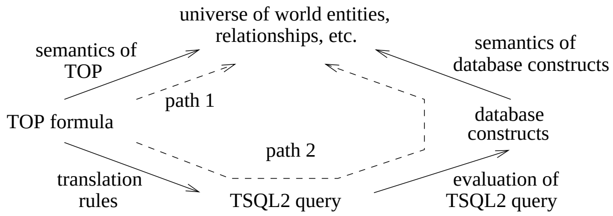

Figure 2: From TOP to world entities

We assume that a TOP expression and a TSQL2 query both refer to some universe of world entities, relationships, etc. (much as in first-order logic), including temporal entities (such as periods). The denotation, in terms of this universe, of a TOP formula is provided by the definition of TOP's semantics (see Androutsopoulos (1996)). The denotation of a TSQL2 expression is given indirectly, in that we assume that there is an "evaluation" function which will map a TSQL2 query to some database constructs (attribute values, relations, etc.), and the semantics of the database indicates how such constructs are related to the universe of world entities, relationships, periods, etc. The situation is roughly as in figure 2.

We have proven that the translation rules are correct, in the sense that the denotation of a TOP formula (roughly speaking its answer) as defined by the semantics of TOP is the same as the denotation (answer) of the corresponding TSQL2 query when determined by way of the "evaluation" function and the database semantics. In terms of figure 2 above, the same semantic content is reached whether path 1 or path 2 is chosen.

7 Comparison to previous work

Although there has been a significant body of work on tense and aspect theories, temporal logics, and temporal database, only a few researchers have attempted to apply this work on NLDBs, most notably Clifford (Clifford 1988, 1990; Clifford and Warren 1983), De et al. (De, Pan, and Whinston 1985, 1987), and Moens (Moens 1987, 1988; Moens and Steedman 1988).

Clifford's work constitutes the most significant previous approach to NLTDBs. Clifford defines formally a temporal version of the relational database model, and a fragment of English that can be used to query databases structured according to his model. Following the Montague tradition (Dowty, Wall, and Peters 1981), Clifford employs an intensional higherorder language (called IL_s) to represent the semantics of the English questions, and a temporally enhanced version of Montague's PTQ grammar to map from English to IL_s. Clifford's coverage of English, however, is extremely narrow, and the semantics of the English temporal mechanisms are oversimplified. For example, perfect and continuous tenses are not supported, and no aspectual taxonomy (section 2) is employed. Clifford also defines a version of relational algebra (Ullman 1988) for his database model (this can be thought of as a theoretical database query language), and sketches a method to translate from IL_s to that algebra. The description of this mapping, however, is very informal, and with no proof of its correctness. There is also no indication that Clifford's theory was ever used to implement an actual NLTDB.

De et al. describe a question-answering system that can handle a fragment of English questions involving time. The "temporal database" in this case is a rather *ad hoc* collection of facts and inference rules, expressed in a kind of logic-programming language. There is no clear intermediate meaning representation language, and it is very difficult to distinguish the part of the system that is responsible for the linguistic processing from the part of the system that is responsible for retrieving information from the database. De et al. consider this an advantage, but it clearly sacrifices modularity and portability (cf. remarks in section 6.1 above). For example, it is very hard to see which parts of the software would have to be modified if the natural language processor were to be used with a commercial DBMS. De et al.'s system does not seem to be based on any clear linguistic analysis. There is also very little information in De *et al.* (1985) and De *et al.* (1987) on exactly which temporal linguistic mechanisms are supported, and which semantics are assigned to these mechanisms.

Moens' work on temporal linguistic phenomena has been highly influential in the area of tense and aspect theories. In the last part of Moens (1987) (see also Moens (1988)), Moens develops a simplistic NLTDB. This has a very limited linguistic coverage, and is mainly intended to illustrate Moens' tense and aspect theory, rather than to constitute a detailed exploration of NLTDBs. As in the case of De et al., Moens' "database" is a rather idiosyncratic collection of facts and inference rules (expressed in Prolog). Moens' system uses a subset of Prolog as its meaning representation language. English questions are translated into expressions of this subset using a DCG grammar (Pereira and Warren 1980), and there are Prolog rules that directly evaluate the meaning representation language expressions against the database. Moens provides essentially no information about the DCG grammar. The definition of the meaning representation language is also very unclear (e.g. it is difficult to see exactly which Prolog expressions are part of the representation language), and the semantics of the language is defined very informally (by listing Prolog code that evaluates some of the possible expressions of the representation language).

We argue in Androutsopoulos (1996) that all previous approaches to NLTDBs suffer from one or more of the following: (i) they ignore important English temporal mechanisms, or (ii) they assign to them over-simplified semantics (both (i) and (ii) apply to Clifford and De et al.), (iii) they lack clearly defined meaning representation languages (De et al. and Moens), (iv) they do not provide complete descriptions of the mappings from natural language to meaning representation language (De et al., Moens), or (v) from meaning representation language to database language (Clifford), (vi) they adopt idiosyncratic (Clifford) and possibly also not well-defined database models or languages (De et al., Moens), (vii) they do not demonstrate that their ideas are implementable (Clifford).

Our framework avoids completely pitfalls (iii) – (vi): our meaning representation language (TOP) is completely and formally defined (see appendix A, and also Androutsopoulos (1996)); the mapping from English to TOP is based on a well-known and widely used grammar framework (HPSG), and all our modifications to it are well-documented (in Androutsopoulos (1996)); we have adopted a database language (TSQL2) and its underlying database model that the temporal database community is considering as a basis for a standard; and we have developed a formally defined (and provably correct) method to translate from TOP to TSQL2.

Our proposal is open to criticism on (i), in the sense that several important temporal linguistic mechanisms remain to be added to our linguistic coverage. The temporal mechanisms that our framework supports, however, are assigned sufficiently elaborate semantics, and therefore our framework does not suffer from (ii). Finally, our framework is partially open to criticism on point (vii), in the sense that our prototype NLTDB has not been linked to an actual DBMS. We are confident that our framework is sufficiently well-documented and concretely founded, to the extent that future work will be able to improve it with regard to points (i) and (vii).

8 Conclusions and further work

We have presented a theoretical framework for building NLTDBs, and a prototype NLTDB that proves the implementability of that framework. We have also argued that our framework constitutes an improvement over previous work on NLTDBs. We discuss below some possible extensions to our work.

Extending the linguistic coverage: We have already pointed out the need to extend the linguistic coverage of our framework. Among the temporal linguistic mechanisms that we have not considered (section 4.9), temporal anaphoric phenomena (discussed briefly in section 4.3) seem to be particularly interesting: several researchers have examined temporal anaphora (e.g. Eberle and Kasper (1989), Hinrichs (1986), Kamp and Reyle (1993), Partee (1984), Webber (1988)), and it would be interesting to explore the applicability of their proposals to NLTDBs. A Wizard of Oz experiment could also be performed to determine which temporal phenomena most urgently need to be added to the linguistic coverage of our framework, and to collect sample questions that could be used as a test suite for NLTDBs (King 1996). (In a Wizard of Oz experiment, users interact through terminals with a person that pretends to be a natural language interface; see Diaper (1986).)

Cooperative responses: In Androutsopoulos (1996) we reach the conclusion that *cooperative responses* (Kaplan 1982) are particularly important in NLTDBs, and that a mechanism to generate such responses should be added to our framework. For example, (140) is currently mapped to a TOP formula that requires BA737 to have *reached* gate 2 for the answer to be affirmative. This causes a negative response to be generated if BA737 was taxiing to gate 2 but never reached it. While a simple negative response is strictly speaking correct, it is hardly satisfactory in this case. A more cooperative response like (141) is needed.

(140) Did BA737 taxi to gate 2?

(141) `BA737 was taxiing to gate 2, but never reached it.`

As noted by Kaplan (1982) and others, cooperative responses can also be a useful enhancement to a NLDB's capabilities even where the reply is positive:

(142) Did BA737 taxi to gate 2?

(143) `BA737 taxied to gate 2 at 9:05am on 10/11/94`

Linking to a DBMS: We have already discussed the need to link the prototype NLTDB to a TSQL2 DBMS (TIMEDB being currently the only option we know of).

Extending the prototype NLTDB: As already noted in section 1, several components would have to be added to the prototype NLTDB, if it were to be used in real-life applications (e.g. a lexical preprocessor, a module for paraphrasing the user's questions, a module for quantifier-scoping, a reasoning component, configuration tools, etc.; see Androutsopoulos (1996) for discussion on how these would fit into our existing architecture). As most of these components are not directly sensitive to temporal issues, it should be possible to implement them by borrowing existing techniques from conventional (non-temporal) NLDBs.

Optimising the TSQL2 code: Although provably correct, the queries that the TOP to TSQL2 translator generates are often verbose. It is usually easy for a TSQL2 programmer to see ways to shorten significantly the generated TSQL2 code. Long database language expressions can confuse DBMSs (especially prototype ones), leading them to unacceptable performance. A TSQL2 optimiser is needed that will apply simplifications to the TSQL2 code of the TOP to TSQL2 translator, before the TSQL2 queries are passed to the DBMS (see Androutsopoulos (1996) for related discussion).

References

Allen, J. (1984). Towards a General Theory of Action and Time. *Artificial Intelligence*, **23**, 123–154.

Androutsopoulos, I. (1996). *A Principled Framework for Constructing Natural Language Interfaces to Temporal Databases*. Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh.

Androutsopoulos, I., Ritchie, G., and Thanisch, P. (1995a). Experience Using TSQL2 in a Natural Language Interface. In J. Clifford and A. Tuzhilin, editors, *Recent Advances in Temporal Databases – Proceedings of the International Workshop on Temporal Databases, Zurich, Switzerland*, Workshops in Computing, pages 113–132. Springer-Verlag, Berlin.

Androutsopoulos, I., Ritchie, G., and Thanisch, P. (1995b). Natural Language Interfaces to Databases – An Introduction. *Natural Language Engineering*, **1**(1), 29–81.

Blackburn, P., Gardent, C., and de Rijke, M. (1994). Back and Forth through Time and Events. In D. Gabbay, editor, *Temporal Logic – Proceedings of the First International Conference, Bonn, Germany*, number 827 in Lecture Notes in Artificial Intelligence, pages 225–237. Springer-Verlag, Berlin.

Boehlen, M. (1995). Temporal Database System Implementations. Department of Mathematics and Computer Science, Aalborg University.

Boehlen, M., Chomicki, J., Snodgrass, R., and Toman, D. (1996). Querying TSQL2 Databases with Temporal Logic. In *Proceedings of the International Conference on Extended Database Technology*.

Carpenter, B. (1992). *The Logic of Typed Feature Structures*. Number 32 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Carpenter, B. and Penn, G. (1994). The Attribute Logic Engine – User's Guide. Philosophy Department, Carnegie Mellon University.

Clifford, J. (1988). Natural Language Querying of Historical Databases. *Computational Linguistics*, **14**(4), 10–34.

Clifford, J. (1990). *Formal Semantics and Pragmatics for Natural Language Querying*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press.

Clifford, J. and Warren, D. (1983). Formal Semantics for Time in Databases. *ACM Transactions on Database Systems*, **8**(2), 215–254.

Comrie, B. (1976). *Aspect*. Cambridge University Press.

Comrie, B. (1985). *Tense*. Cambridge University Press.

Copestake, A. and Sparck Jones, K. (1990). Natural Language Interfaces to Databases. *The Knowledge Engineering Review*, **5**(4), 225–249.

Crouch, R. and Pulman, S. (1993). Time and Modality in a Natural Language Interface to a Planning System. *Artificial Intelligence*, **63**, 265–304.

Dalrymple, M. (1988). The Interpretation of Tense and Aspect in English. In *In Proceedings of the 26th Annual Meeting of ACL*, pages 68–74.

Davidson, J. and Kaplan, S. (1983). Natural Language Access to Data Bases: Interpreting Update Requests. *Computational Linguistics*, **9**(2), 57–68.

Davies, C., Lazell, B., Hughes, M., and Cooper, L. (1995). Time is Just Another Attribute – or at Least, Just Another Dimension. In J. Clifford and A. Tuzhilin, editors, *Recent Advances in Temporal Databases – Proceedings of the International Workshop on Temporal Databases, Zurich,* *Switzerland*, Workshops in Computing, pages 175–193. Springer-Verlag, Berlin.

De, S., Pan, S., and Whinston, A. (1985). Natural Language Query Processing in a Temporal Database. *Data & Knowledge Engineering*, **1**, 3–15.

De, S., Pan, S., and Whinston, A. (1987). Temporal Semantics and Natural Language Processing in a Decision Support System. *Information Systems*, **12**(1), 29–47.

De Roeck, A. and Lowden, B. (1986). Generating English Paraphrases from Formal Relational Calculus Expressions. In *Proceedings of the 11th International Conference on Computational Linguistics, Bonn, Germany*, pages 581–583.

Diaper, D. (1986). Identifying the Knowledge Requirements of an Expert System's Natural Language Processing Interface. In M. Harrison and A. Monk, editors, *People and Computers: Designing for Usability – Proceedings of the Second Conference of the British Computer Society, Human Computer Interaction Specialist Group, University of York*, pages 263–280. Cambridge University Press.

Dowty, D. (1977). Toward a Semantic Analysis of Verb Aspect and the English 'Imperfective' Progressive. *Linguistics and Philosophy*, **1**, 45–77.

Dowty, D. (1986). The Effects of Aspectual Class on the Temporal Structure of Discourse: Semantics or Pragmatics? *Linguistics and Philosophy*, **9**, 37–61.

Dowty, D., Wall, R., and Peters, S. (1981). *Introduction to Montague Semantics*. D.Reidel Publishing Company, Dordrecht, Holland.

Eberle, K. and Kasper, W. (1989). Tenses as Anaphora. In *Proceedings of the 4th Conference of the European Chapter of ACL, Manchester, England*, pages 43–50.

Enc, M. (1986). Towards a Referential Analysis of Temporal Expressions. *Linguistics and Philosophy*, **9**, 405–426.

Gabbay, D., Hodkinson, I., and Reynolds, M. (1994). *Temporal Logic: Mathematical Foundations and Computational Aspects*. Oxford University Press.

Grosz, B., Sparck Jones, K., and Webber, B. (1986). *Readings in Natural Language Processing*. Morgan Kaufmann.

Hinrichs, E. (1986). Temporal Anaphora in Discourses of English. *Linguistics and Philosophy*, **9**, 63–82.

Hinrichs, E. (1988). Tense, Quantifiers, and Contexts. *Computational Linguistics*, **14**(2), 3–14.

Jensen, C., editor (1993). *A Consensus Glossary of Temporal Database Concepts*. Technical Report R-93-2035, Institute for Electronic Systems, Department of Mathematics and Computer Science, Aalborg University, Denmark. Earlier version in *SIGMOD Record*, 21(3):35–43, September 1992.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic*. Kluer Academic Publishers.

Kaplan, S. (1982). Cooperative Responses from a Portable Natural Language Data Base Query System. *Artificial Intelligence*, **19**, 165–187.

Kent, S. (1993). *Modelling Events from Natural Language*. Ph.D. thesis, Imperial College of Science Technology and Medicine.

King, M. (1996). Evaluating Natural Language Processing Systems. *Communications of the ACM*, **39**(1), 73–79.

Lascarides, A. (1988). *A Formal Semantic Analysis of the Progressive*. Ph.D. thesis, University of Edinburgh.

Melton, J. and Simon, A. (1993). *Understanding the New SQL: A Complete Guide*. Morgan Kaufmann Publishers, San Mateo, California.

Moens, M. (1987). *Tense, Aspect and Temporal Reference*. Ph.D. thesis, University of Edinburgh.

Moens, M. (1988). Temporal Databases and Natural Language. In C. Rolland, F. Bodart, and M. Leonard, editors, *Proceedings of Working Conference on Temporal Aspects in Information Systems, Sophia-Antipolis, France*, pages 171–183. Elsevier Science Publishers.

Moens, M. and Steedman, M. (1988). Temporal Ontology and Temporal Reference. *Computational Linguistics*, **14**(2), 15–28.

Mourelatos, A. (1978). Events, Processes, and States. *Linguistics and Philosophy*, **2**, 415–434.

Parsons, T. (1990). *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press.

Partee, B. (1984). Nominal and Temporal Anaphora. *Linguistics and Philosophy*, **7**, 243–286.

Pereira, F. and Warren, D. (1980). Definite Clause Grammars for Language Analysis – A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence*, **13**, 231–278. Reprinted in Grosz *et al.* (1986), pages 101–124.

Perrault, C. and Grosz, B. (1988). Natural Language Interfaces. In H. Shrobe, editor, *Exploring Artificial Intelligence*, pages 133–172. Morgan Kaufmann Publishers Inc., San Mateo, California.

Pollard, C. and Sag, I. (1987). *Information-Based Syntax and Semantics – Volume 1, Fundamentals*. Center for the Study of Language and Information, Stanford.

Pollard, C. and Sag, I. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press and Center for the Study of Language and Information, Stanford.

Rayner, M. (1993). *Abductive Equivalential Translation and its Application to Natural Language Database Interfacing*. Ph.D. thesis, Royal Institute of Technology, Stockholm.

Reichenbach, H. (1947). *Elements of Symbolic Logic*. Collier-Macmillan, London.

Richards, B., Bethke, I., van der Does, J., and Oberlander, J. (1989). *Temporal Representation and Inference*. Cognitive Science Series, Academic Press, Harcourt Brace Jovanovich Publishers.

Ritchie, G. (1979). Temporal Clauses in English. *Theoretical Linguistics*, **6**(1), 87–115.

Snodgrass, R., editor (1995). *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers.

Tansel, A., Clifford, J., Gadia, S., Jajodia, S., Segev, A., and Snodgrass, R. (1993). *Temporal Databases – Theory, Design, and Implementation*. Benjamin/Cummings, California.

Tsotras, V. and Kumar, A. (1996). Temporal Database Bibliography Update. *ACM SIGMOD Record*, **25**(1).

Ullman, J. (1988). *Principles of Database and Knowledge-Base Systems– Volume 1*. Computer Science Press, Rockville, Maryland.

van Benthem, J. (1983). *The Logic of Time*. D.Reidel Publishing Company, Dordrecht, Holland.

Vendler, Z. (1967). Verbs and Times. In *Linguistics in Philosophy*, chapter 4, pages 97–121. Cornell University Press, Ithaca, NY.

Vlach, F. (1993). Temporal Adverbials, Tenses and the Perfect. *Linguistics and Philosophy*, **16**, 231–283.

Webber, B. (1988). Tense as Discourse Anaphor. *Computational Linguistics*, **14**(2), 61–73.

A Appendix : The Top Language

A.1 The Syntax of TOP

Notation

{ } : Used to group elements in BNF. Not part of TOP language.
F^* : Zero or more repetitions of F.
F^+ : One or more repetitions of F.
| : Separates different possible RHSs of BNF rule.
$\langle n \rangle$: Footnote – indicates some further condition on rule;
terminal symbols are in lower case, usually with an initial capital,
nonterminals are in upper case

Grammar

The distinguished symbol is $FORMS$.

$$
\begin{array}{lll}
FORMS \rightarrow YNFORMS \quad | \quad WHFORMS & & \\
WHFORMS \rightarrow WHFORMS1 \quad | \quad WHFORMS2 & & \\
WHFORMS1 \rightarrow \{?\,VARS\}^+\,YNFORMS & & \langle 1 \rangle \\
WHFORMS2 \rightarrow ?_{mxl}\,VARS\{?\,VARS\}^*\,YNFORMS & & \langle 2 \rangle \\
YNFORMS \rightarrow AFORMS & & \\
 & |\ YNFORMS \wedge YNFORMS & \\
 & |\ Pres[YNFORMS] & \\
 & |\ Past[VARS, YNFORMS] & \langle 3 \rangle \\
 & |\ Perf[VARS, YNFORMS] & \langle 3 \rangle \\
 & |\ At[TERMS, YNFORMS] & \langle 4 \rangle \\
 & |\ At[YNFORMS, YNFORMS] & \\
 & |\ Before[TERMS, YNFORMS] & \langle 4 \rangle \\
 & |\ Before[YNFORMS, YNFORMS] & \\
 & |\ After[TERMS, YNFORMS] & \langle 4 \rangle \\
 & |\ After[YNFORMS, YNFORMS] & \\
 & |\ Ntense[VARS, YNFORMS] & \langle 3 \rangle \\
 & |\ Ntense[now*, YNFORMS] & \\
 & |\ For[CPARTS, VQTY, YNFORMS] & \\
 & |\ Fills[YNFORMS] & \\
 & |\ Begin[YNFORMS] & \\
 & |\ End[YNFORMS] & \\
 & |\ Culm[LITERAL] & \\
AFORMS \rightarrow LITERAL \quad | \quad Part[PARTS, VARS] \quad | \quad Part[PARTS, VARS, VORD] & &
\end{array}
$$

$$LITERAL \rightarrow PFUNS(\{TERMS,\}^* TERMS)$$
$$TERMS \rightarrow CONS \quad | \quad VARS$$
$$PARTS \rightarrow CPARTS \quad | \quad GPARTS$$
$$VORD \rightarrow 0 \quad | \quad -1 \quad | \quad -2 \quad | \quad \ldots$$
$$VQTY \rightarrow 1 \quad | \quad 2 \quad | \quad 3 \quad | \quad \ldots$$

PFUNS, *CPARTS*, *GPARTS*, *CONS*, *VARS* are disjoint open classes of terminal symbols.

Footnotes

$\langle 1 \rangle$: Each *VARS* terminal which appears in the $\{?VARS\}^+$ expression must occur inside the *YNFORMS* expression.

$\langle 2 \rangle$: Each *VARS* terminal which appears in the $\{?VARS\}^+$ expression must occur inside the *YNFORMS* expression, and the *VARS* terminal of the $?_{mxl}VARS$ expression occurs at least once inside the $YNFORMS$ expression as the first argument of a $Past, Perf, At, Before, After$, or $Ntense$ operator, or as the second argument of a $Part$ operator.

$\langle 3 \rangle$: The *VARS* expression does not occur in the *FORMS* expression.

$\langle 4 \rangle$: If the *TERMS* expression is a *VARS*, then the *VARS* expression does not occur in the *YNFORMS* expression.

A.2 Semantics of TOP

Ontology

Point structure: A *point structure* for TOP is an ordered pair $\langle PTS, \prec \rangle$, such that PTS is a non-empty set, $\prec$ is a binary relation over $PTS \times PTS$, and $\langle PTS, \prec \rangle$ has the following five properties:

transitivity: If $t_1, t_2, t_3 \in PTS$, $t_1 \prec t_2$, and $t_2 \prec t_3$, then $t_1 \prec t_3$.

irreflexivity: If $t \in PTS$, then $t \prec t$ does not hold.

linearity: If $t_1, t_2 \in PTS$ and $t_1 \neq t_2$, then exactly one of the following holds: $t_1 \prec t_2$ or $t_2 \prec t_1$.

left and right boundedness: There is a $t_{first} \in PTS$, such that for all $t \in PTS$, $t_{first} \preceq t$. Similarly, there is a $t_{last} \in PTS$, such that for all $t \in PTS$, $t \preceq t_{last}$.

discreteness: For every $t_1, t_2 \in PTS$, with $t_1 \neq t_2$, there is at most a finite number of $t_3 \in PTS$, such that $t_1 \prec t_3 \prec t_2$.

The elements of PTS are called *time-points*. The following functions are defined on time-points:

prev(t) and next(t): If $t_1 \in PTS - \{t_{last}\}$, then $next(t_1)$ denotes a $t_2 \in PTS$, such that $t_1 \prec t_2$ and for no $t_3 \in PTS$ is it true that $t_1 \prec t_3 \prec t_2$. Similarly, if $t_1 \in PTS - \{t_{first}\}$, then $prev(t_1)$ denotes a $t_2 \in PTS$, such that $t_2 \prec t_1$ and for no $t_3 \in PTS$ is it true that $t_2 \prec t_3 \prec t_1$. Here, whenever $next(t)$ is used, it is assumed that $t \neq t_{last}$. Similarly, whenever $prev(t)$ is used, it is assumed that $t \neq t_{first}$.

Periods and instantaneous periods: A *period* p over $\langle PTS, \prec \rangle$ is a non-empty subset of PTS with the following property:

convexity: If $t_1, t_2 \in p$, $t_3 \in PTS$, and $t_1 \prec t_3 \prec t_2$, then $t_3 \in p$.

$PERIODS_{\langle PTS, \prec \rangle}$ is the set of all periods over $\langle PTS, \prec \rangle$. If $p \in PERIODS_{\langle PTS, \prec \rangle}$ and p contains only one time-point, then p is an *instantaneous period over* $\langle PTS, \prec \rangle$. $INSTANTS_{\langle PTS, \prec \rangle}$ is the set of all instantaneous periods over $\langle PTS, \prec \rangle$. For simplicity, we may write $PERIODS$ and $INSTANTS$ instead of $PERIODS_{\langle PTS, \prec \rangle}$ and $INSTANTS_{\langle PTS, \prec \rangle}$. $PERIODS^*_{\langle PTS, \prec \rangle}$ (or simply $PERIODS^*$) is the set $PERIODS \cup \{\emptyset\}$.

Subperiods: p_1 is a *subperiod* of p_2, iff $p_1, p_2 \in PERIODS$ and $p_1 \subseteq p_2$. In this case we write $p_1 \sqsubseteq p_2$. ($p_1 \subseteq p_2$ is weaker than $p_1 \sqsubseteq p_2$, because it does not guarantee that $p_1, p_2 \in PERIODS$.) Similarly, p_1 is a *proper subperiod* of p_2, iff $p_1, p_2 \in PERIODS$ and $p_1 \subset p_2$. In this case we write $p_1 \sqsubset p_2$.

Maximal periods: If S is a set of periods, then $mxlpers(S)$ is the set of *maximal periods* of S. $mxlpers(S) \equiv \{p \in S \mid \text{for no } p' \in S \text{ is it true that } p \sqsubset p'\}$.

minpt(S) and maxpt(S): If $S \subseteq PTS$, $minpt(S)$ denotes the time-point $t \in S$, such that for every $t' \in S$, $t \preceq t'$. Similarly, if $S \subseteq PTS$, $maxpt(S)$ denotes the time-point $t \in S$, such that for every $t' \in S$, $t' \preceq t$.

Notation: Following standard conventions, $[t_1, t_2]$ denotes the set $\{t \in PTS \mid t_1 \preceq t \preceq t_2\}$. Similarly, $(t_1, t_2]$ denotes the set $\{t \in PTS \mid t_1 \prec t \preceq t_2\}$. $[t_1, t_2)$ and (t_1, t_2) are defined similarly.

Model structure

A TOP model M is an ordered 7-tuple:

$$M = \langle \langle PTS, \prec \rangle, OBJS, f_{cons}, f_{pfuns}, f_{culms}, f_{gparts}, f_{cparts} \rangle$$

such that $\langle PTS, \prec \rangle$ is a point structure for TOP (section A.2), $PERIODS_{\langle PTS, \prec \rangle} \subseteq OBJS$, and f_{cons}, f_{pfuns}, f_{culms}, f_{gparts}, and f_{cparts} are as specified below.

OBJS: $OBJS$ is a set containing all the objects in the modelled world that can be denoted by TOP terms.

$\mathbf{f_{cons}}$: f_{cons} is a function $CONS \mapsto OBJS$, i.e. f_{cons} maps each TOP constant to a world object.

$\mathbf{f_{pfuns}}$: f_{pfuns} is a function that maps each pair $\langle \pi, n \rangle$, where $\pi \in PFUNS$ and $n \in \{1, 2, 3, \dots\}$, to another function $(OBJS)^n \mapsto pow(PERIODS)$ (where pow (S) denotes the powerset of S).

For every $\pi \in PFUNS$ and $n \in \{1, 2, 3, \dots\}$, $f_{pfuns}(\pi, n)$ must have the following property: for every $\langle o_1, o_2, \dots, o_n \rangle \in (OBJS)^n$, it must be the case that:

if $p_1, p_2 \in f_{pfuns}(\pi, n)(o_1, o_2, \dots, o_n)$ and $p_1 \cup p_2 \in PERIODS$, then $p_1 = p_2$

Intuitively, f_{pfuns} shows the maximal periods where the situation represented by $\pi(\tau_1, \dots, \tau_n)$ holds, for every combination of arguments $\tau_1, \dots, \tau_n$.

$\mathbf{f_{culms}}$: f_{culms} is a function that maps each pair $\langle \pi, n \rangle$, where $\pi \in PFUNS$ and $n \in \{1, 2, 3, \dots\}$, to another function $(OBJS)^n \mapsto \{T, F\}$. Intuitively, f_{culms} shows whether or not a situation reaches a climax at the latest time-point where the situation is ongoing.

$\mathbf{f_{gparts}}$: f_{gparts} is a function that maps each element of $GPARTS$ to a *gappy partitioning*. A gappy partitioning is a subset S of $PERIODS$, such that for every $p_1, p_2 \in S$, $p_1 \cap p_2 = \emptyset$, and $\bigcup_{p \in S} p \neq PTS$.

$\mathbf{f_{cparts}}$: f_{cparts} is a function that maps each element of $CPARTS$ to a *complete partitioning*. A complete partitioning is a subset S of $PERIODS$, such that for every $p_1, p_2 \in S$, $p_1 \cap p_2 = \emptyset$, and $\bigcup_{p \in S} p = PTS$.

Variable assignment

A variable assignment w.r.t. a TOP model M is a function $g : VARS \mapsto OBJS$ (g assigns to each variable an element of $OBJS$). G_M, or simply G, is the set of all possible variable assignments w.r.t. M.

If $g \in G$, $\beta \in VARS$, and $o \in OBJS$, then g_o^β is the variable assignment defined as follows: $g_o^\beta(\beta) = o$, and for every $\beta' \in VARS$ with $\beta' \neq \beta$, $g_o^\beta(\beta') = g(\beta)$.

Denotation of a TOP expression

In the following, we shall use the non-terminal symbols of the TOP syntax given in section A.1 above as the names of sets containing expressions which can be analysed syntactically as the corresponding non-terminal. That is, where we stipulate, for example, that "$\phi \in YNFORMS$", this means that "ϕ is a TOP expression which syntactically is a *YNFORMS*". Also, we will assume that in the semantic definitions for complex expressions, all the various sub-expressions are of the correct syntactic types, as given by the grammar in section A.1 above.

An *index of evaluation* is an ordered 3-tuple $\langle st, et, lt \rangle$, such that $st \in PTS$, $et \in PERIODS$, and $lt \in PERIODS^*$.

The *denotation* of a TOP expression ξ w.r.t. a model M, an index of evaluation $\langle st, et, lt \rangle$, and a variable assignment g, is written $\|\xi\|^{M,st,et,lt,g}$ or simply $\|\xi\|^{st,et,lt,g}$. When the denotation of ξ does not depend on st, et, and lt, we may write $\|\xi\|^{M,g}$ or simply $\|\xi\|^g$.

- If $\kappa \in CONS$, then $\|\kappa\|^g = f_{cons}(\kappa)$.
- If $\beta \in VARS$, then $\|\beta\|^g = g(\beta)$.
- If $\phi \in YNFORMS$, then $\|\phi\|^{st,et,lt,g} \in \{T, F\}$.
- If $\pi(\tau_1, \tau_2, \ldots, \tau_n) \in LITERAL$, then $\|\pi(\tau_1, \tau_2, \ldots, \tau_n)\|^{st,et,lt,g} = T$ iff $et \sqsubseteq lt$ and for some $p_{mxl} \in f_{pfuns}(\pi, n)(\|\tau_1\|^g, \|\tau_2\|^g, \ldots, \|\tau_n\|^g)$, $et \sqsubseteq p_{mxl}$.
- If $\phi_1, \phi_2 \in YNFORMS$, then $\|\phi_1 \land \phi_2\|^{st,et,lt,g} = T$ iff $\|\phi_1\|^{st,et,lt,g} = T$ and $\|\phi_2\|^{st,et,lt,g} = T$.
- $\|Part[\sigma, \beta, \nu_{ord}]\|^g$ is T, iff all the following hold (below $f = f_{cparts}$ if $\sigma \in CPARTS$, and $f = f_{gparts}$ if $\sigma \in GPARTS$):
  - $g(\beta) \in f(\sigma)$,
  - if $\nu_{ord} = 0$, then $st \in g(\beta)$,
  - if $\nu_{ord} \leq -1$, then the following set contains exactly $-\nu_{ord} - 1$ elements:

$$\{p \in f(\sigma) \mid maxpt(g(\beta)) \prec minpt(p) \text{ and } maxpt(p) \prec st\}$$

- $\|Part[\sigma,\beta]\|^g = T$ iff $g(\beta) \in f(\sigma)$ (where $f = f_{cparts}$ if $\sigma \in CPARTS$, and $f = f_{gparts}$ if $\sigma \in GPARTS$).

- $\|Pres[\phi]\|^{st,et,lt,g} = T$, iff $st \in et$ and $\|\phi\|^{st,et,lt,g} = T$.

- $\|Past[\beta,\phi]\|^{st,et,lt,g} = T$, iff $g(\beta) = et$ and $\|\phi\|^{st,et,lt \cap [t_{first},st),g} = T$.

- $\|Culm[\pi(\tau_1,\ldots,\tau_n)]\|^{st,et,lt,g} = T$, iff $et \sqsubseteq lt$, $f_{culms}(\pi,n)(\|\tau_1\|^g,\ldots,\|\tau_n\|^g) = T$, $S \neq \emptyset$, and $et = [minpt(S), maxpt(S)]$, where:

$$S = \bigcup_{p \in f_{pfuns}(\pi,n)(\|\tau_1\|^g,\ldots,\|\tau_n\|^g)} p$$

- $\|At[\tau,\phi]\|^{st,et,lt,g} = T$, iff $\|\tau\|^g \in PERIODS$ and $\|\phi\|^{st,et,lt \cap \|\tau\|^g,g} = T$.

- $\|At[\phi_1,\phi_2]\|^{st,et,lt,g} = T$, iff for some et' $et' \in mxlpers(\{e \in PERIODS \mid \|\phi_1\|^{st,e,PTS,g} = T\})$ and $\|\phi_2\|^{st,et,lt \cap et',g} = T$.

- $\|Before[\tau,\phi]\|^{st,et,lt,g} = T$, iff $\|\tau\|^g \in PERIODS$ and $\|\phi\|^{st,et,lt \cap [t_{first},minpt(\|\tau\|^g)),g} = T$.

- $\|Before[\phi_1,\phi_2]\|^{st,et,lt,g} = T$, iff for some et', $et' \in mxlpers(\{e \in PERIODS \mid \|\phi_1\|^{st,e,PTS,g} = T\})$, and $\|\phi_2\|^{st,et,lt \cap [t_{first},minpt(et')),g} = T$.

- $\|After[\tau,\phi]\|^{st,et,lt,g} = T$, iff $\|\tau\|^g \in PERIODS$ and $\|\phi\|^{st,et,lt \cap (maxpt(\|\tau\|^g),t_{last}],g} = T$.

- $\|After[\phi_1,\phi_2]\|^{st,et,lt,g} = T$, iff for some et', $et' \in mxlpers(\{e \mid \|\phi_1\|^{st,e,PTS,g} = T\})$ and $\|\phi_2\|^{st,et,lt \cap (maxpt(et'),t_{last}],g} = T$.

- $\|Fills[\phi]\|^{st,et,lt,g} = T$, iff $et = lt$ and $\|\phi\|^{st,et,lt,g} = T$.

- $\|Begin[\phi]\|^{st,et,lt,g} = T$, iff $et \sqsubseteq lt$ and for some et' $et' \in mxlpers(\{e \in PERIODS \mid \|\phi\|^{st,e,PTS,g} = T\})$ and $et = \{minpt(et')\}$.

- $\|End[\phi]\|^{st,et,lt,g} = T$, iff $et \sqsubseteq lt$ and for some et' $et' \in mxlpers(\{e \in PERIODS \mid \|\phi\|^{st,e,PTS,g} = T\})$ and $et = \{maxpt(et')\}$.

- $\|Ntense[\beta,\phi]\|^{st,et,lt,g} = T$, iff for some $et' \in PERIODS$, $g(\beta) = et'$ and $\|\phi\|^{st,et',PTS,g} = T$.

- $\|Ntense[now^*, \phi]\|^{st,et,lt,g} = T$, iff $\|\phi\|^{st,\{st\},PTS,g} = T$.

- $\|For[\sigma_c, \nu_{qty}, \phi]\|^{st,et,lt,g} = T$, iff $\|\phi\|^{st,et,lt,g} = T$, and for some $p_1, p_2, \ldots, p_{\nu_{qty}} \in f_{cparts}(\sigma_c)$, it is true that $minpt(p_1) = minpt(et)$, $next(maxpt(p_1)) = minpt(p_2)$, $next(maxpt(p_2)) = minpt(p_3), \ldots, next(maxpt(p_{\nu_{qty}-1})) = minpt(p_{\nu_{qty}})$, and $maxpt(p_{\nu_{qty}}) = maxpt(et)$.

- $\|Perf[\beta, \phi]\|^{st,et,lt,g} = T$, iff $et \sqsubseteq lt$, and for some $et' \in PERIODS$, it is true that $g(\beta) = et'$, $maxpt(et') \prec minpt(et)$, and $\|\phi\|^{st,et',PTS,g} = T$.

- $\|?\beta_1 \; ?\beta_2 \; \ldots \; ?\beta_n \; \phi\|^{st,et,lt,g} = \{\langle g(\beta_1), g(\beta_2), \ldots, g(\beta_n)\rangle \mid \|\phi\|^{st,et,lt,g} = T\}$

- $\|?_{mxl}\beta_1 \; ?\beta_2 \; ?\beta_3 \; \ldots \; ?\beta_n \; \phi\|^{st,et,lt,g} = \{\langle g(\beta_1), g(\beta_2), g(\beta_3), \ldots, g(\beta_n)\rangle \mid \|\phi\|^{st,et,lt,g} = T$, and for no $et' \in PERIODS$ and $g' \in G$ is it true that $\|\phi\|^{st,et',lt,g'} = T$, $g(\beta_1) \sqsubset g'(\beta_1)$, $g(\beta_2) = g'(\beta_2)$, $g(\beta_3) = g'(\beta_3)$, $\ldots$, $g(\beta_n) = g'(\beta_n)\}$

The denotation of ϕ w.r.t. M, st, written $\|\phi\|^{M,st}$ or simply $\|\phi\|^{st}$, is defined only for TOP formulae:

- If $\phi \in YNFORMS$, then $\|\phi\|^{st} =$
  - T, if for some $g \in G$ and $et \in PERIODS$, $\|\phi\|^{st,et,PTS,g} = T$,
  - F, otherwise

- If $\phi \in WHFORMS$, then $\|\phi\|^{st} = \bigcup_{g \in G,\; et \in PERIODS} \|\phi\|^{st,et,PTS,g}$.

Within the NLTDB, each yes/no question (e.g. *"Is tank 2 empty?"*) is mapped to a $\phi \in YNFORMS$ (multiple formulae are generated for ambiguous questions). The answer is affirmative iff $\|\phi\|^{st} = T$. With questions containing wh-words (e.g. *"Who repaired what?"*), the question is mapped to a $\phi \in WHFORMS$ (again multiple formulae are generated if the question is ambiguous), and the answer reports the tuples in $\|\phi\|^{st}$.