Gefördert vom

bmb+f

Bundesministerium für
Bildung, Wissenschaft,
Forschung und Technologie

Verbmobil

Verbundvorhaben

Tracking and Understanding Temporal Descriptions in Dialogue

Manfred Stede
Stefan Haas
Uwe Küssner

Technische Universität Berlin

Vm

Report 232
Oktober 1998

Oktober 1998

Manfred Stede
Stefan Haas
Uwe Küssner

Kontaktadresse:
Technische Universität Berlin
Fachbereich Informatik (13)
Projektgruppe KIT
Sekr. FR 6-10
Franklinstraße 28/29
D-10587 Berlin

Tel.: (030) 314 - 24944
Fax: (030) 314 - 24929
e-mail: kitvm11@cs.tu-berlin.de

Gehört zum Antragsabschnitt: AP4.2 Domänenmodellierung
AP4.5 Desambiguierung/Klärungsdialoge

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) unter dem Förderkennzeichen 01 IV 701 P 9 gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei dem Autor.
ISSN 1434-8845

Understanding and tracking temporal descriptions in dialogue*

Manfred Stede, Stefan Haas, Uwe Küssner
Technische Universität Berlin

Abstract

The Verbmobil system provides speech-to-speech translations for appointment scheduling dialogues, where two partners seek a date that suits both. The 'context-evaluation' module of Verbmobil monitors this negotiation by tracking and comparing the temporal descriptions in a two-step manner: first, the temporal expressions of an utterance are mapped to a shallow representation language "ZeitGram"; these terms are then given a semantic interpretation as "interval descriptions" (IDs). In this paper, we describe both representation levels and the mapping between them. We sketch the inferences implemented on the ID level and illustrate their application in Verbmobil's context evaluation.

1 Temporal expressions in negotiating appointments

1.1 The Verbmobil scenario

The Verbmobil system [Wahlster 1993] provides speech-to-speech translations among German, English, and Japanese for dialogues in the domain of appointment negotiation. In such dialogues, the partners try to find a date that suits both their schedules; typical conversations of this kind include a sequence of proposals and rejections, or a successive "zooming in" on some particular date. A sample dialogue (taken from the Verbmobil corpus) containing both these elements is given in Figure 1.

1.2 Tasks of the ConEval module

Given the domain of discourse, the majority of utterances processed by Verbmobil contain temporal expressions. The evaluation of these expressions takes

*This paper appeared in: B. Schröder, W. Lenders, W. Hess, T. Portele (eds.): Computers, Linguistics, and Phonetics between Language and Speech — Proceedings of the 4th Conference on Natural Language Processing KONVENS '98. Frankfurt: Peter Lang, 1998.

A1: *ja, guten Morgen, Icks hier. ich rufe an, damit wir einen Termin finden für unsere fünftägige Reise zur DGFS Tagung in Mannheim.*
B1: *ja, Morgen, Frau Icks, wann wär's Ihnen denn recht.*
A2: *ja, ich könnte zum Beispiel am siebzehnten Januar, wenn das der erste Tag wäre, bis zum ja, bis zum zweiundzwanzigsten Januar.*
B2: *oh, das tut mir leid, ich muß am achtzehnten und neunzehnten zum ZDF in Mainz.*
A3: *ja, dann klappt das ja schon mal nicht. wir müssen auch ins Auge fassen , daß wir ein Vorbereitungstreffen vorher haben müssen. vielleicht sollten wir das erstmal festlegen, weil einen gemeinsamen Tag zu finden, wird ja einfacher sein.*
B3: *ja, mit dem eintägigen Vorbereitungstreffen. wär' Ihnen das recht, wenn wir das auf 'n Wochenende legen?*
A4: *ja, natürlich, kein Problem. also, schlagen Sie was vor.*
B4: *wie wär's mit Februar äh der vierten Februar, das ist ein Samstag.*
A5: *ja, gut. ist zwar nur recht spät, aber also, ich kann da. das legen wir da mal fest. okay. und sollen wir dann direkt die anschließende Woche für die fünftägige Reise nehmen?*
B5: *ja, das wär' mir ganz recht. allerdings kann ich am siebten und achten auch nicht, das heißt, wir müßten die fünftägige Reise über ein Wochenende legen. wäre Ihnen das recht?*
A6: *kein Problem. wenn wir dann vom neunten bis zum vierzehnten fahren würden.*
B6: *ja, das ist mir auch sehr recht. alles klar.*

Figure 1: Sample dialogue from the Verbmobil corpus

place in the contextual evaluation (ConEval) module. To provide the background for our approach, we first briefly describe how ConEval fits into the overall architecture of Verbmobil.

In the "deep analysis" line of Verbmobil, the central data structure is a syntactic/semantic representation called the Verbmobil Interface Term (VIT). It is produced in the analysis component and passed on to the transfer module, which converts it to a target language VIT. In parallel, the source VIT is also given to ConEval, which is in charge of deeper semantic and pragmatic analyses. ConEval performs disambiguations, resolves anaphors, determines the dialogue act of the utterance, and extracts the "kernel message" from the utterance (which is later used for a dialogue protocol). For these tasks, we map the VIT to a conceptual representation implemented in the description logic LOOM [MacGregor, Bates 1987]; the sequence of utterance representations is our context model.

From the perspective of translation, a "deep understanding" of temporal expressions is often not necessary: in order to translate *etwas später* into *a bit later* one need not know what exactly is being negotiated to be later than what. On the other hand, determining the dialogue act often requires the information whether a proposed date is in fact a new suggestion, a return to a previous suggestion, or a specialization of a broader suggestion made just earlier. Thus, the same utterance can in different contexts be a SUGGEST or an ACCEPT. Also, extracting kernel messages and protocoling the dialogue requires knowledge on how temporal expressions connect; protocoling requires abstracting from fine-grained details and presupposes knowledge of the precise relationships between temporal expressions.

A final reason for paying close attention to the temporal expressions is robustness of the overall system. When translating spontaneous speech, recognition errors need to be reckoned with. And since negotiating an appointment is the primary goal of the dialogue, such recognition errors should, if at all possible, not lead to incorrect translations of temporal expressions. To have a chance of identifying implausible analyses of temporal expressions, we need to check both the consistency of individual expressions (to notice, e.g., *∗dreissigster Februar*) as well as the overall consistency of the date negotiation process.

For all these reasons, the ConEval module monitors temporal expressions and builds complete and coherent date descriptions from them. E.g., given the utterance *An dem Montag kann ich erst etwas später*, ConEval figures out which exact Monday is being referred to and what the reference point for *später* is. Using knowledge of the preceding discourse as well as language-independent calendar information, we determine that the speaker has stated his availability at (for example) any time after 3pm on Monday the 30th of March, 1998.

1.3 Related work

A similar problem is addressed in the COSMA system [Busemann et al. 1997], which provides a NL interface for communication with appointment scheduling agents. COSMA also tracks temporal expressions and thereby resolves "temporal ellipses" by supplementing from previous utterances those parts of date descriptions that are not explicit in the current utterance; it also detects certain kinds of inconsistencies. The major differences to our work are COSMA's dealing with German input only and with written language; as indicated above, the confrontation with spontaneous speech creates many additional problems in Verbmobil. In particular, the range of temporal expressions to be handled seems much wider in our corpus, even though a detailed comparison is not possible on the basis of the paper [Busemann et al. 1997]. Further, due to the different scenario, COSMA employs shallow parsing techniques for its purpose of analyzing email messages, which is not sufficient for translation in Verbmobil.

Wiebe et al. [1997] also work in the same domain and present an algorithm for temporal reference resolution, which maintains descriptions of temporal intervals by looking for the most likely antecedent temporal expression in the prior discourse. They do not address the issues of successively constructing the interval descriptions from a syntactic representation, but already assume a semantic representation as input. In contrast, our emphasis here is on robustly building up the representations from a syntactic input.

2 Two-step evaluation

For the task of tracking temporal expressions, our application scenario poses three specific problems. 1) Natural language offers many ways to refer to a particular date/time, and we have to reduce them all to a single canonical representation. 2) As we are dealing with spontaneous speech, we cannot rely on nice and clean input, but on the contrary have to reckon with incomplete or erroneous information. 3) As the system performs bidirectional translation, we must deal with both German and English input.

Under these circumstances, it is not feasible to map the VIT directly to a canonical representation which allows for the necessary computations — the gap between such a level and the VIT input is too wide. Therefore, we introduce an intermediate level of representation for temporal expressions, the 'ZeitGram' language. ZeitGram expressions are relatively close to natural language expressions; for example, *der dritte Montag nach Ostern* is represented as `[after(3,dow:mon,holiday:easter)]`. In this way, ZeitGram abstracts from surface-linguistic idiosyncrasies but mirrors the underlying constructions. The target for developing ZeitGram was thus a language whose expressions are relatively easy to build from a VIT, and that can be further processed by deeper referential analysis. This second step maps ZeitGram expressions to 'interval descriptions' (IDs): canonical representations of temporal information that allow for the necessary computations. In the following, we first explain the ZeitGram language, and then the ID representations together with the possible inferences.

2.1 ZeitGram

For reasons of space, our description of ZeitGram given here is quite brief and omits a few details. In the following, we first explain the general form of ZeitGram expressions using a context-free grammar[1] and discuss the role of modifiers (*<u>früh</u> am Morgen, in der <u>Mitte</u> des Monats, ...*). Then we turn to simple expressions like time of day and weekday, and proceed to more complex expressions: intervals (*zwischen acht und neun Uhr*), expressions with a point

[1]For the description, we use an extended BNF notation: `ITEM+` denotes a nonempty list of `ITEMs`, `ITEM*` a list that might be empty. `<ITEM1 | ITEM2 | ITEM3>` says that one of the `ITEMs` is to be chosen.

of reference and a temporal distance (*drei Wochen vor Ostern*), and expression with a counter (*der dritte Freitag im Februar*).

The shape of ZeitGram expressions The basic shape of a ZeitGram expression is a list of `TEMPOBJ`s; a phrase like *Am Dienstag um elf Uhr* is thus represented by this unordered list: `[dow:tue, tod:11:00]`. `TEMPOBJ`s have the shape `TYPE:VALUE`, i.e., each expression has a type prefix. The other subtype of `DATE_EXPR` adds a "fuzzyfier" to the list.

```
DATE_EXPR ::= TEMPOBJ+ | fuzzy(TEMPOBJ+)
TEMPOBJ ::= POINT | INTERVAL | MOD | COMPOSED_POINTLIKE |
            MOD(MODIFIABLE_POINTLIKE) | MODIFIABLE_POINTLIKE
MODIFIABLE_POINTLIKE ::= UNIT | POINTLIKE | COUNT_POINTLIKE
MOD ::= early | late | begin | middle | end
```

Modifiers and "fuzzyfication" Most `TEMPOBJ`s can be qualified with modifiers such as `early` or `late`, which have scope over a `POINTLIKE` object. Examples are *Mitte der Woche, früh am Nachmittag*. Among the non-modifiable `TEMPOBJ`s are clock times (see below): *?früh um vierzehn Uhr*. However, the modifier can also be used as a single object, without scope over a `TEMPOBJ`: *treffen wir uns früh, um sieben vielleicht* is represented as `[early, tod:07:00]`. Therefore, `MOD` in itself is also a `TEMPOBJ`. "Fuzzy" in our terminology are expressions like *gegen drei Uhr; so um den achten Mai*. For these, we use the functor `fuzzy` with scope over a `TEMPOBJ` list.

Simple `TEMPOBJ`s Among the simple `TEMPOBJ`s, clock times are to be separated from other expressions. The former denote points of time (`POINT`), while the latter denote time spans; these can however be conceptualized as points, and hence are called `POINTLIKE`.

Clock times (time of day, `TOD`) are normalized to the form `hh:mm`, so we abstract over linguistic variants like *zwanzig nach elf* or *fünf vor halb acht*. The reason is that such variants are typically language-specific or even dialect-specific (e.g. *dreiviertel vier*). For ambiguous expressions like *halb vier* (03:30 or 15:30) we add an expression `am` or `pm` (as part of `POD`, see below) if the intended reading is clear; otherwise the expression is left underspecified. Hence, `TOD` runs from `00:00` to `11:59`.

```
POINT ::=  tod:TOD
TOD   ::=  00:00 | ... | 11:59
```

`POINTLIKE`s comprise part of day (`POD`), day of week, part of year (seasons), week of year (e.g. *die dritte Kalenderwoche*), week of month (i.e. *die zweite Woche im Januar*), day of month, month, year, and holidays. Furthermore, a totally unspecific expression as in *Paßt Ihnen der dritte* can be represented as xoy (for 'x of y'). In most cases, this would refer to the third day of a month, but other options are possible: *Wie wäre der zweite Montag im Februar? — Nein. Paßt Ihnen der dritte?* In English, due to the lack of morphological gender, *the third* can also refer to the third week. The above-mentioned am and pm are taken to belong to POD, because they can occur as independent expressions. E.g., English offers phrases like *let's have a pm meeting tomorrow*.

The final POINTLIKE is ana, short for 'anaphoric', which can be used in expressions like *der Tag danach* = [after(1,day,ana)] or *vier Wochen früher* = [before(4,week,ana)].

```
POINTLIKE ::= ana | pod:POD | dow:DOW | poy:POY | woy:WOY | wom:WOM |
              dom:DOM | moy:MOY | year:YEAR | holiday:HOLIDAY | xoy:NUMBER
POD ::= morning | beforenoon | noon | afternoon | evening | night |
        daytime | am | pm
DOW ::= mon | tue | wed | thu | fri | sat | sun | today | tomorrow |
        totomorrow | yesterday | yeyesterday | workday | weekend
POY ::=  spring | summer | fall | winter
WOY ::=  1 | 2 | ... | 52
WOM ::=  1 | 2 | 3 | 4 | 5
DOM ::=  1 | 2 | ... | 31
MOY ::=  jan | feb | ... | dec
YEAR ::=  1900 |...| 1999
HOLIDAY ::=  easter | christmas | ...
```

Intervals As INTERVAL we treat all time spans with one or two boundaries. For one thing, these are DURATIONs that refer to the length of an interval (*Wir sollten uns für drei Stunden treffen*) and are represented by a NUMBER and a temporal UNIT.

For intervals with one boundary (*ab 14 Uhr, vor dem Dienstag*), we use the functors before and after, followed by a DATE_EXPR. Also, we allow the variants 'inclusive' and 'exclusive', which are sometimes explicitly referred to: *Nächste Woche paßt es nur bis einschließlich Dienstag.*

Closed intervals consist of two DATE_EXPRs and an atom describing the kind of interval: 1) from_to: The expression denotes the complete span between the boundaries of the interval – *Wir treffen uns von halb vier bis um sieben.* 2) point_between: The expression denotes a time span in which a point is to be localized – *Wir sollten zwischen drei und vier Uhr anfangen.* 3) between: The expression is ambiguous between the two cases – *Wie wäre es zwischen fünf und sechs?* As the examples show, the verb can often help in disambiguating the category of INTERVAL.

```
INTERVAL ::= DURATION | LIMIT | BOUNDARIES
DURATION ::= dur(NUMBER,UNIT)
UNIT ::= year | month | week | day | hour | minute
LIMIT ::= BEFORE_OR_AFTER(TEMPOBJ+)
BEFORE_OR_AFTER ::= before | after | ex_before | ex_after |
```

```
                    in_before| in_after
BOUNDARIES ::= boundaries(TEMPOBJ+, TEMPOBJ+, <from_to | between |
                                               point_between>)
```

Expressions with reference point A time point can be characterized by a reference point and the distance from it: In *vier Tage nach Ostern*, *Ostern* is the reference point. We distinguish four cases. 1) Implicit reference: if the reference point is not mentioned, it is to be understood as 'now'. Example: *Wir treffen uns in drei Wochen* means *heute in drei Wochen*. For these cases, we use the functor `in` and a `DURATION`. 2) Weekday as reference: phrases like *Freitag in acht Tagen* or *Freitag in drei Wochen* are quite common in German. For these we also use `in`, which hence can also be a two-place functor, with the extra argument of a weekday. 3) Time span before/after reference point: a `POINTLIKE` serves as referent; a `COUNTER` and a `UNIT` or a weekday (`DOW`) characterize the time span before or after. Examples: *die letzte Woche vor Ostern* = `[before(last, week, holiday:ostern)]`; *drei Tage danach* = `[after(3, day, ana)]`; *der zweite Samstag nach Neujahr* = `[after(2, dow:sat, holiday:neujahr)]` 4) Weeks with reference day: as a special case, calendar weeks are often referred to by stating a day therein: *die Woche ab dem vierzehnten, die Woche vor Ostern, die Woche um den dreizehnten herum.* For theses cases we use a `POINT_MOD`: `before`, `after`, or `around`.

```
COMPOSED_POINTLIKE ::= <before | after>(COUNTER, <UNIT | DOW>, POINTLIKE) |
              in(DURATION) | in(DOW, DURATION) | POINT_MOD(week, POINTLIKE)
POINT_MOD ::= before | after | around
```

Expressions with counter The final class comprises expressions that identify the time point or interval by counting a `UNIT` or a weekday. In case of a `UNIT`, there is always a larger `UNIT` serving as reference span. We distinguish three cases: 1) The reference span is not mentioned: *in der dritten Woche*. 2) Weekdays or months can be modified by *nächsten, letzten* o.ä. (`DEIC_COUNTER`): *am vorletzten Freitag, im nächsten Januar.* 3) When giving the reference span, weeks or weekdays can be determined by counting: *die dritte Woche im Januar, der letzte Samstag im Januar.* Furthermore, the reference span can be determined by counting either of the `UNITs` `year` or `month`; here, however, the counter is restricted to `DEIC_COUNTER`: *die dritte Woche im nächsten Jahr, der zweite Samstag im nächsten Monat.*

```
COUNT_POINTLIKE ::= counted(COUNTER,UNIT) | DEIC_COUNTER(<DOW | MOY>) |
                    counted(COUNTER,<week | DOW>, COUNT_UNIT_OR_MOY)
COUNT_UNIT_OR_MOY ::= MOY | DEIC_COUNTER(<year | month>)
COUNTER ::= NUMBER | DEIC_COUNTER
DEIC_COUNTER ::=  this | last | lastlast | lastlastlast | next | nextnext
NUMBER ::= 1 | 2 | ...
```

2.2 Constructing interval descriptions

The Verbmobil application scenario implies that a good deal of information necessary for understanding an utterance is given implicitly in the context. This holds in particular for temporal expressions; in Figure 1, for instance, the year is missing in A2 (and implicitly "filled in" by the hearer as the current or following year). The response B2 does not give an explicit month, but it can be inferred to be the same as that in the previous utterance. For automatically tracking temporal expressions, inferences of this kind need to be computed. For this step, we map the ZeitGram terms to another level of representation, which is more amenable to semantic interpretation.

IDs In order to deal elegantly with underspecifications, we chose typed feature structures as representation formalism, where any temporal expression is represented as an interval (following Allen [1984]). We devised a modified unification algorithm that realizes "zooming" as a monotonic addition of information. The algorithm takes two interval descriptions and computes a consistent partial unifier and possibly a pair of non-unifiable rest items. We impose an ordering Spec on unifying items following their granularity; this way, the inconsistent rest of unifications is most specific.

An interval description ID is a FS of type **Id** consisting of two **date** expressions:

$$\begin{bmatrix} \textbf{Id} \\ \textit{BEG:} \begin{bmatrix} \textbf{date} \end{bmatrix} \\ \textit{END:} \begin{bmatrix} \textbf{date} \end{bmatrix} \end{bmatrix}$$

date has a range of subtypes reflecting the presence and absence of the four attributes YEAR, MONTH, DAY, and TIME. Any combination of these corresponds to a distinct type, which allows for measuring the informativeness of a **date** expression. Figure 2 shows the type hierarchy. As an example, the most specific type, **tdmy_date** is defined as follows:

$$\begin{bmatrix} \textbf{tdmy\_date} \\ \textit{YEAR:} \begin{bmatrix} \textbf{year} \end{bmatrix} \\ \textit{MONTH:} \begin{bmatrix} \textbf{month} \end{bmatrix} \\ \textit{DAY:} \begin{bmatrix} \textbf{day} \end{bmatrix} \\ \textit{TIME:} \begin{bmatrix} \textbf{time} \end{bmatrix} \end{bmatrix}$$

The three types **month, day, time** are in turn composed structures. **day** has the two attributes **dow** (day of week) and **dom** (day of month), because it can be defined either way (*am Montag* versus *am vierten*). **month** is decomposed into **moy** and **wom** (week of month), and **time** into **hour** and **minute**. The terminal atomic types, finally, can be ordered by their specificity. To this end, we define the Spec relation, which reflects the inclusion relation between temporal units:

$$\textbf{year} >_{\mathsf{Spec}} \textbf{moy} >_{\mathsf{Spec}} \textbf{wom} >_{\mathsf{Spec}} \textbf{dow} >_{\mathsf{Spec}} \textbf{dom} >_{\mathsf{Spec}} \textbf{hour} >_{\mathsf{Spec}} \textbf{minute}$$

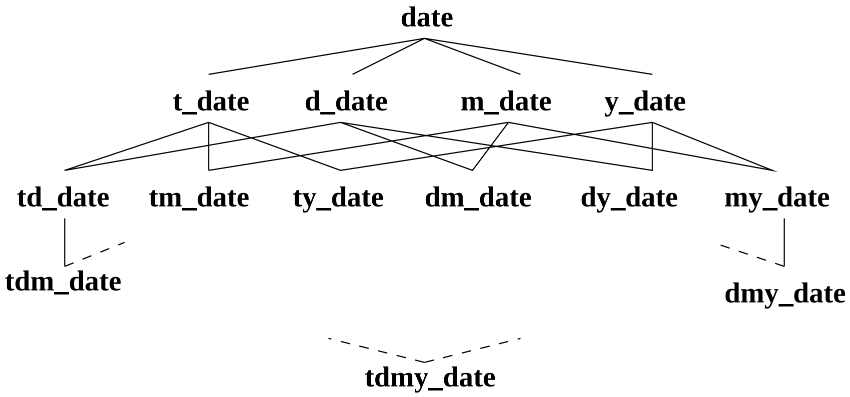

Figure 2: The type hierarchy for **date**

Building IDs from `TEMPOBJs` The basic idea of translating ZeitGram terms, i.e. lists of `TEMPOBJs`, to IDs is to successively build an underspecified FS for each item in the `TEMPOBJ list` and to unify it with the FS built up so far. Following Spec, unification proceeds from the coarse to the fine-grained terms. In case unification fails at some point, the ZeitGram term is inconsistent. Otherwise, the resulting ID represents all the information derived from the term. For translating the individual `TEMPOBJs`, we use four knowledge sources: 1) A table provides a heuristic mapping from `pod` and `poy` terms (e.g., `morning, evening, summer`, ...) to concrete intervals. 2) A calendar provides the interpretation model, including dates for holidays, the precise intervals for week-of-year, etc. 3) The situation of utterance provides a time stamp for interpreting indexical terms (e.g., `today`). 4) The previous context provides anchors for resolving anaphoric expressions such as *in der folgenden Woche*.

Figure 3 shows an abridged sample derivation. Given the utterance *das Treffen beginnt um 3:30, Montag nachmittag*, we first show the ZeitGram term and then the FSs for the individual `TEMPOBJs`. In the first one, notice the disjunction representing the ambiguity between 3:30 and 15:30. The bottom of the figure gives the final result for the complete term.

Comparing IDs After constructing IDs for individual utterances, we now have to provide comparisons for pairs of IDs, so that the relation between subsequent temporal expressions in dialogue turns can be found. This step is performed in in conjunction with the dialogue processing module of Verbmobil [Alexandersson et al. 1995].

Our unification, as pointed out above, produces a unifier plus (possibly) an incompatible rest structure. Since unification follows the Spec ordering, any

S: *das Treffen beginnt um 3:30, montag nachmittag*
ZG: `[tod:3:30, dow:mon, pod:afternoon]`

$\mathcal{T}($`[tod:3:30]`$) :=$

$$id_1 : BEG : \left[\begin{array}{ll} \multicolumn{2}{l}{\boxed{1}\,\mathbf{tdmy\_date}} \\ \mathit{YEAR:} & [\,\mathbf{year}\,] \\ \mathit{MONTH:} & [\,\mathbf{month}\,] \\ \mathit{DAY:} & [\,\mathbf{day}\,] \\ \mathit{TIME:} & \left[\begin{array}{ll} \multicolumn{2}{l}{\mathbf{mh\_time}} \\ \mathit{HOUR:} & \{\,[\,\mathbf{15}\,]\,,\ [\,\mathbf{3}\,]\,\} \\ \mathit{MINUTE:} & [\,\mathbf{30}\,] \end{array}\right] \end{array}\right],\ id_1 : END : \left[\boxed{1}\,\mathbf{tdmy\_date}\right]$$

$\mathcal{T}($`[dow:mon]`$) :=$

$$id_2 : BEG : \left[\begin{array}{ll} \multicolumn{2}{l}{\boxed{1}\,\mathbf{dmy\_date}} \\ \multicolumn{2}{l}{\ldots} \\ \mathit{DAY:} & \left[\begin{array}{ll} \multicolumn{2}{l}{\mathbf{dow\_day}} \\ \mathit{DOW:} & [\,\mathbf{mon}\,] \end{array}\right] \end{array}\right],\ id_2 : END : \left[\boxed{1}\,\mathbf{dmy\_date}\right]$$

$\mathcal{T}($`[pod:afternoon]`$) :=$

$$id_3 : BEG : \left[\begin{array}{ll} \multicolumn{2}{l}{\mathbf{tdmy\_date}} \\ \mathit{YEAR:} & \left[\boxed{1}\,\mathbf{year}\right] \\ \mathit{MONTH:} & \left[\boxed{2}\,\mathbf{month}\right] \\ \mathit{DAY:} & \left[\boxed{3}\,\mathbf{day}\right] \\ \mathit{TIME:} & \left[\begin{array}{ll} \multicolumn{2}{l}{\mathbf{h\_time}} \\ \mathit{HOUR:} & [\,\mathbf{13}\,] \end{array}\right] \end{array}\right],id_3 : END : \left[\begin{array}{ll} \multicolumn{2}{l}{\mathbf{tdmy\_date}} \\ \mathit{YEAR:} & \left[\boxed{1}\,\mathbf{year}\right] \\ \mathit{MONTH:} & \left[\boxed{2}\,\mathbf{month}\right] \\ \mathit{DAY:} & \left[\boxed{3}\,\mathbf{day}\right] \\ \mathit{TIME:} & \left[\begin{array}{ll} \multicolumn{2}{l}{\mathbf{h\_time}} \\ \mathit{HOUR:} & [\,\mathbf{18}\,] \end{array}\right] \end{array}\right]$$

$id_1 \sqcup id_2 \sqcup id_3 :=$

$$id : BEG : \left[\begin{array}{ll} \multicolumn{2}{l}{\boxed{1}\,\mathbf{tdmy\_date}} \\ \multicolumn{2}{l}{\ldots} \\ \mathit{DAY:} & \left[\begin{array}{ll} \multicolumn{2}{l}{\mathbf{dow\_day}} \\ \mathit{DOW:} & [\,\mathbf{mon}\,] \end{array}\right] \\ \mathit{TIME:} & \left[\begin{array}{ll} \multicolumn{2}{l}{\mathbf{hm\_time}} \\ \mathit{HOUR:} & [\,\mathbf{15}\,] \\ \mathit{MINUTE:} & [\,\mathbf{30}\,] \end{array}\right] \end{array}\right],id : END : \left[\boxed{1}\,\mathbf{tdmy\_date}\right]$$

Figure 3: Representation example

failure will occur at the least specific attribute. Therefore, the pair of unifier and incompatible rest FS can serve as the reference frame for comparing two **date** FSs. Since the attributes where unification fails have the same type (one of **year, month, day, time**), a relation before can be defined for computing which **date** precedes the other on the calendar. Then, by applying before to the combinations of BEGINs and ENDs of two interval descriptions, the relationships used by Allen [1984] can be determined; for example, we determine whether one interval precedes the other or one includes the other. To this end, the specificity of an ID is defined by the width of the interval as well as the respective types of the end points. A FS ID1 is more specific than a FS ID2 if 1) the interval denoted by ID1 is within the boundaries of ID2, or if 2) both unify without rest and the begin/end types of ID2 subsume those of ID1, and they are of the same or a coarser granularity w.r.t. Spec (i.e., $id2.BEG \geq_{\mathsf{Spec}} id1.BEG$ ...)

The types of unifier and incompatible rest also provide important clues for relating the current ID to previous ones and thus for finding anaphoric antecedents within the "history list" of previous expressions. This part of our algorithm is still under development, though.

3 The representations at work: temporal inferences

Finally, we describe how our procedure would deal with the dialogue in Figure 1. In A1, we assign the dialogue act INIT and recognize the presence of a temporal expression ('tempex' for short), represented as `[dur(5, day)]`. B1 is a REQUEST-SUGGEST-DATE, and A2 is a SUGGEST-DATE with tempex `[boundaries([dom:17, moy:jan], [dom:22], between)]`. At this point, we verify that the tempex is compatible with the previous `[dur(5, day)]`. Next, speaker B performs a SUGGEST-EXCLUDE-DATE with the tempex `[boundaries( [dom:18], [dom:19], from_to)]`. Comparing this to the previous tempex yields the relation INCLUSION and we conclude that the proposal made in A2 is rejected. In A3, speaker A opens discussion on a separate appointment and states its duration as `[dur(1, day)]`. B3 responds with a SUGGEST-SUPPORT-DATE but makes a fairly general proposal, indicated by the indefinite determiner (*'n Wochenende*): `[dow:weekend]`. It follows a REQUEST-SUGGEST-DATE in A4, and then in B4 we get a SUGGEST-DATE with `[moy:feb, dom:4, dow:sat]`. Using the calendar knowledge, we verify the consistency of this description and also find that it is indeed compatible with the previous `[dow:weekend]` – the negotiation is still on track. In A5 we first get an ACCEPT-DATE and then the speaker returns to the original appointment under negotiation. *Die direkt anschließende Woche* is `[after([ana]), dur(1, week)]`. To resolve ana, we take the most recent tempex, which was Sat, Feb 4; and thus we can calculate *die anschließende Woche* as Feb 6 to Feb 13. Now, however, in B5 we encounter a SUGGEST-EXCLUDE-DATE by speaker B, with `[boundaries([dom:7], [dom:8], from_to)]` which we find to be included by the previous proposal; accordingly, the week suggested by A is rejected. Still in B5, B suggests `[dow:weekend]`, and A responds with `[boundaries([dom:9], [dom:14], from_to)]`. Filling in the month information from the previous turn (February), we verify that a weekend is indeed included. Finally, B6 is an ACCEPT-DATE that concludes the conversation, and ConEval regards the date negotiation as having succeeded.

4 Summary

For a system translating appointment-scheduling dialogues, it is important that temporal expressions be understood and tracked, so that the relevant connections between the expressions can be computed. Given the wide range of linguistic possibilities for stating a date and time in German and English, we have argued that a two-step mapping of such expressions is appropriate. To this end, we first defined 'ZeitGram', a representation language that is relatively close to linguistic surface forms but provides some first abstractions. Then, ZeitGram terms are mapped to Interval Descriptions, typed feature structures that allow for well-defined computations. We have described the syntax of IDs and the mechanism of a unification-like procedure that successively constructs IDs from ZeitGram terms. Also, we described how the unification procedure can be employed for comparing distinct IDs and determining which relationship holds between them. For dialogue processing, we have devised an initial algorithm that determines the points at which new date proposals are made or old ones are re-opened.

References

[Alexandersson et al. 1995] J. Alexandersson, E. Maier, N. Reithinger. "A Robust and Efficient Three-Layered Dialog Component for a Speech-to-Speech Translation System." In: Proceedings of the 7th Conference of the European Chapter of the ACL (EACL-95), Dublin, 1995.

[Allen 1984] J. Allen. "Toward a general theory of action and time." In: *Artificial Intelligence* 23: 123-154

[Busemann et al. 1997] S. Busemann, T. Declerck, A.K. Diagne, L. Dini, J. Klein, S. Schmeier. "Natural Language Dialogue Service for Appointment Scheduling Agents." In: *Proceedings of Fifth Conference of Applied Natural Language Processing*, Washington DC, April 1997.

[MacGregor and Bates 1987] R. MacGregor, R. Bates. "The loom knowledge representation language." Technical Report ISI/RS-87-188, USC/ISI, 1987.

[Wahlster 1993] W. Wahlster. *Verbmobil: Translation of face-to-face dialogues.* In: Proceedings of the Third European Conference on Speech Communication and Technology, Berlin, 1993.

[Wiebe et al. 1997] J. Wiebe, T. O'Hara, K. McKeever, T. Öhrström-Sandgren. "An empirical approach to temporal reference resolution." In: *Proceedings of the Second Conference On Empirical Methods in Natural Language Processing (EMNLP-2)*, August 1997, Providence, RI.