

LTIMEX: Representing the Local Semantics of Temporal Expressions

Paweł Mazur^{1,2}Robert Dale²

¹Institute of Applied Informatics,
Wrocław University of Technology
Wyb. Wyspiańskiego 27,
50-370 Wrocław, Poland
Email: pawel@mazur.wroclaw.pl

²Centre for Language Technology
Macquarie University
NSW 2109, Sydney, Australia
Email: Robert.Dale@mq.edu.au
Email: Pawel.Mazur@mq.edu.au

Abstract—Semantic information retrieval requires that we have a means of capturing the semantics of documents; and a potentially useful feature of the semantics of many documents is the temporal information they contain. In particular, the temporal expressions contained in documents provide important information about the time course of the events those documents describe. Unfortunately, temporal expressions are often context-dependent, requiring the application of information about the context in order to work out their true values. We describe a representational formalism for temporal information that captures what we call the *local semantics* of such expressions; this permits a modularity whereby the context-independent contribution to meaning can be computed independently of the global context of interpretation, which may not be immediately or easily available. Our representation, LTIMEX, is intended as an extension to widely-used TIMEX2 and TimeML representations.

I. INTRODUCTION

AN IMPORTANT part of the meaning of many documents is the temporal information they contain. In particular, many documents present narratives over sequences of events, and specifications of dates and times in the form of *temporal expressions* provide timestamps for these events; these are of significant utility for any application that aims to mine information about events across a large document set. From the information retrieval perspective, time is an important notion that can be used for indexing, organizing, retrieving and, finally, presenting the content of documents; these issues have been the topic of recent studies (see, for example, [1]).

Unfortunately, not all such temporal information is expressed in easy-to-capture and easy-to-interpret expressions. While fully-specified expressions, such as dates like *25th November 2010*, are not uncommon in text, far more common are context-dependent temporal expressions, like *yesterday*, *10th June*, and *the previous summer*. Properly assigning values to such expressions is the aim of *temporal expression tagging*; the TIMEX2 standard (introduced as annotation guidelines in [2]) and TimeML (see [3], [4]) have been developed as forms of representation for such values, and a considerable body of research focuses on developing tools that can accurately annotate such expressions with their absolute values (see, for example, [5]–[10]).

The assumption underlying the processing carried out by existing tools is that there are two distinct stages involved: first the *extent* of the temporal expression must be determined—in other words, it must be properly recognised—and then its value may be computed, taking into account both the lexical content of the expression and the wider context in which it is situated.

This second step conflates knowledge from two sources: information that can be derived from the content of the temporal expression itself, and arbitrary real-world reasoning that takes account of contextual information. As a simple example, the meaning of *yesterday* depends crucially on the time at which is uttered; however, common to all instances of its use is that it means ‘the day before today’. Based on this observation, we can compute the partial meaning of such expressions, thus permitting a modularisation of the process into a part that computes the context-independent *local semantics* of the expression, and a part which uses the wider context to determine the *global semantics*. The first of these can proceed in the absence of the second. However, we then require an interface between the two levels of processing; for this we propose what we call LTIMEX, an extension of the TIMEX2 and TimeML representations that allows us to capture partial meanings.

In this paper, which further develops our earlier work presented in [11], we describe the LTIMEX representation in some detail. Section II explains the relationship between LTIMEX and the existing TIMEX2 and TimeML standards; Section III describes in detail how we represent a wide variety of categories of temporal expressions using LTIMEX; and Section IV draws some conclusions about the utility of the scheme.

II. BACKGROUND: THE TIMEX2 REPRESENTATION

TIMEX2, developed by the information extraction community, is a widely-used annotation standard for temporal expressions in text; it serves as a target representation for temporal expression taggers. For the purpose of its use in this standard, a temporal expression is defined as a linguistic expression which

TABLE I
ENCODING OF POINTS AND PERIODS IN TIMEX2.

Attribute Value	Meaning
1	The second millennium AD
19	The 20th century AD
199	The 1990s
1992	Year 1992
1992-06-27	27 June 1992
1992-06-27T18:04	27 June 1992 18:04
1992-06-27T18:04:56	1992-06-27 18:04:56
1992-06-27TMO	morning of 27 June 1992
1992-W04	The fourth week of 1992
1992-SU	Summer of 1992
1992-H1	1st half of 1992
1992-Q3	3rd quarter of 1992
BC0346	Year 346 BCE
MA6	6 million years ago
PAST_REF	vague reference to past
T19:00	7pm (used in a non-specific context)
XXXX-06-XX	a day in June (non-specific context)
P2Y3M	2 years and 3 months
P2DT6H	2 days and 6 hours
PT3.5H	3.5 hours
P2DE	2 decades

references a *point in time* (such as a calendar date or time of day) or a *period* (also called a *duration*).¹

TIMEX2 defines five attributes to represent the meaning of a temporal expression: `VAL`, `ANCHOR_VAL`, `ANCHOR_DIR`, `MOD`, `SET`. These attributes are used, respectively, to encode the temporal location of a point on a timeline or a duration of a period; to encode the temporal location of one of the period's end-points; to capture the direction of temporal reference from the anchoring point; to express modifications to more basic temporal values; and to flag whether the temporal expression refers to a set of temporal entities.

Values of the `VAL` and `ANCHOR_VAL` attributes use a string representation based on formats defined in the ISO-8601 standard: calendar date (`YYYY-MM-DD`), week date (`YYYY-Www-D`), time of day (`hh:mm:ss`), date and time (`YYYY-MM-DDThh:mm:ss`), and duration (`PnYnMnDtnHnMnS` or `PnW`). The individual character positions in the date and time strings correspond to particular granularities of temporal information, as demonstrated by the examples in Table I. TIMEX2 extends the encodings provided in the ISO standard by introducing tokens representing additional temporal granularities: for example, in place of a month number, TIMEX2 also permits codes for year seasons (e.g. `SU` for summer), half-years (e.g. `H1`), and quarter-years (e.g. `Q3`). It also adds support for BCE years, references to the distant past (i.e. billions, millions, and thousands of years ago) and general references to the past, present and future. For non-specific

¹The distinction between a point and a period in TIMEX2 is different from the distinction often made in artificial intelligence and work on the philosophy of time. In contrast to usage in those areas, here a point is not a durationless instant, but a point on a timeline of some granularity. For example, a month is annotated as a point, not a period, when referenced as an element of a calendar (e.g. *He graduated in November*).

use of expressions (as in, for example, *a sunny day in June*) TIMEX2 uses an uppercase X to fill the slots at the unspecified granularities (for example, `XXXX-06-XX`). In regard to the encoding of duration, TIMEX2 adds new temporal units for decades, centuries and millennia.

In documents, a temporal expression tagger encodes these values as attributes of inline XML annotations, as in the following example:

(1) I left town on `<TIMEX2 VAL="2010-07-15">15th July 2010</TIMEX2>`.

What is notable about this temporal expression is that it is *fully-specified* or *explicit*: the temporal value can be computed using the lexical content of the string alone, without any reference to context.²

Not all temporal expressions are of this kind; rather, many are context-dependent, in that they only partially specify a temporal value, and require incorporation of information available in the context in order to derive their full interpretation. Unfortunately, the nature of the TIMEX2 representation means that there is no way of annotating the value of temporal expressions until this full interpretation has been carried out.

To address this problem, in this paper we describe an extension of TIMEX2 that uses a few simple notational devices to permit the representation of partially-interpreted temporal expressions in a way that is consistent with the original TIMEX2 specification. This has the benefit that it is easy to learn for those already familiar with TIMEX2 annotation; it can also take advantage of existing tools for evaluating the performance of temporal expression taggers. Most importantly, though, it provides us with an interface language that enables a modularisation of the process of temporal expression interpretation, so that we can use distinct components for determining the local semantic interpretation of a temporal expression and for the process of incorporating contextual information.

In TimeML, temporal expressions are annotated inline using TIMEX3 tags. TIMEX3 is a subsequent annotation standard to TIMEX2 which uses a different set of attributes. A detailed description is not appropriate here, but we note that its `value` attribute is used in exactly the same way as TIMEX2 uses the `VAL` attribute, which is the most important and relevant attribute for the development of LTIMEX. TIMEX3 represents the end-points of a period by means of additional TIMEX3 annotations, so its `value` attribute also replaces the `ANCHOR_VAL` attribute in TIMEX2. Thus, although LTIMEX was originally developed as an extension to TIMEX2, it is also compatible with TimeML and TIMEX3.

III. REPRESENTING PARTIAL INTERPRETATIONS

TIMEX2 was designed to annotate temporal expressions with their global semantics, i.e. the temporal value obtained by interpreting the expression in the context of the content of the document in which it is used. Our experience with the

²This is not entirely true, since at least a particular calendar and a particular timezone are assumed, so values will always be relative to these. For most practical purposes, however, this makes the expressions context-free.

TABLE II
A SUMMARY OF THE LTIMEX ATTRIBUTES.

Attribute	Comments
L-VAL	Encodes the local semantics of expressions concerning the temporal location of points: for underspecified values, the missing slots are filled with x; underspecified time is separated from the date components with t; for offsets, the encodings start with +, -, > or <; ordinally-specified elements are encoded with the pattern \$nu. Encoding of durations is the same as in TIMEX2.
L-ANCHOR_VAL	Local semantics of temporal location of end-points; see the description of L-VAL.
L-ANCHOR_DIR, L-MOD, L-SET	Same values as for the corresponding attributes in TIMEX2.
L-TYPE	Encodes the taxonomical type of an expression. The possible values are: EXPLICIT, UNDERSPECIFIED, OFFSET, OFFSET-DEICTIC, OFFSET-ANAPHORIC, EVENT-BASED, EVENT-BASED-POINT, EVENT-BASED-PERIOD, PERIOD, SET, SET-POINTS, SET-PERIODS.
L-EVENT_ID	Stores an identifier of an event for event-based expressions.
L-ANCHOR_TYPE	For anchored period expressions where the anchor is an offset, indicates the type of the offset; the possible values of the attribute are DEICTIC and ANAPHORIC.

development of a temporal expression tagger (presented in [9]) revealed that it is beneficial for both design and evaluation to explicitly recognize the semantics of the expression with no context involved; we refer to this as the *local semantics*, representing the partial and underspecified context-free meaning of the expression.

LTIMEX extends the set of attributes from TIMEX2 to provide a vocabulary for capturing partially-specified meaning. Some of these local attributes simply mirror the existing attributes; others, however, add new types of information that are intended to be of use to a subsequent processing stage that determines the global semantics of the temporal expression. The attributes provided by LTIMEX are shown in Table II.

A key feature here is the L-TYPE attribute, which stores the type of an expression; this captures the essential distinctions between different kinds of expressions, so that this information can be used to guide subsequent processing. In Table III we present the types of temporal expressions that we distinguish in this work, along with examples of each. These do not represent a flat taxonomy: the major types are expressions referring to single point and duration temporal entities, each of which may have subtypes; but there are also expressions referring to sets of such temporal entities, as well as ordinally-specified, modified and non-specific expressions.

Many of these types are introduced in TIMEX2 at the level of what we refer to as global semantics. The literature on temporal expression tagging identifies subtypes of point expressions that require different interpretation algorithms to derive their global semantics: *explicit*, *underspecified*, and *deictic* and *anaphoric offsets*.³ We follow these taxonomic distinctions for the purpose of representing the local semantics. For the same reason, we also identify *ordinally-specified* expressions, allowing a further level of distinction for explicit, underspecified and offset expressions.

³The terminology used in literature in this regard varies; in this work we use what we believe are the most intuitive terms.

TABLE III
THE TYPES OF TEMPORAL EXPRESSIONS.

Expression Type	Example Expression
Explicit Point	<i>Friday, 3 April 1998</i>
Underspecified Point	<i>23rd June</i>
Deictic Offset	<i>tomorrow</i>
Anaphoric Offset	<i>the next month</i>
Event-based Point	<i>the day when the last fortress fell</i>
Duration	<i>six months and two days</i>
Event-based Duration	<i>the first two minutes of the meeting</i>
Ordinally-specified	<i>the last Tuesday in 1997</i>
Modified Points	<i>the middle of August</i>
Modified Durations	<i>nearly two decades</i>
Non-specific Point	<i>a sunny day in July</i>
Set	<i>every Tuesday</i>

TABLE IV
VALUES ASSIGNED TO EXPLICIT DATES AND TIMES IN TIMEX2.

No	Expression	Representation (VAL)
1	3rd January 1987	1987-01-03
2	Friday, 3 April 1998	1998-04-03
3	24/03/1980	1980-03-24
4	03/24/1980	1980-03-24
5	November 1996	1996-11
6	1960s	196
7	12th January 2001 11:59 pm	2001-01-12T23:59

Below, we present the LTIMEX scheme by discussing the representation of local semantics for a wide variety of types of temporal expressions that are found in real texts.

A. Explicit Expressions

These expressions are the only context-independent point expressions. For these, the local semantics is always the same as the global semantics, so our L-VAL simply mirrors the VAL in TIMEX2. We present some examples in Table IV.

TABLE V
EXAMPLES OF UNDERSPECIFIED EXPRESSIONS IN LTIMEX.

No	Expression	Representation (L-VAL)
1	January 3	xxxx-01-03
2	the nineteenth	xxxx-xx-19
3	November	xxxx-11
4	summer	xxxx-SU
5	'63	xx63
6	the '60s	xx6
7	9 pm	xxxx-xx-xxT21
8	11:59 pm	xxxx-xx-xxT23:59
9	eleven in the morning	xxxx-xx-xxT11:00
10	ten minutes to 3	xxxx-xx-xxT02:50
11	15 minutes after the hour	xxxx-xx-xxTxx:15
12	Friday	xxxx-Wxx-5
13	8:00 p.m. Friday	xxxx-Wxx-5T20:00
14	eight o'clock Friday	xxxx-Wxx-5t08:00

B. Underspecified Expressions

Underspecified expressions differ from explicit expressions in that they omit elements of information, which then have to be recovered from the context by some process of interpretation. LTIMEX provides for the representation of underspecified expressions by marking those elements of the temporal value that are missing with a special symbol; here we use a lowercase *x*, reminiscent of its common use as a variable.⁴ Table V presents examples of a range of underspecified expressions along with their L-VAL attributes using this encoding.

For underspecified expressions referring to times that do not indicate the part of day (either 'am' or 'pm'), such as those in Rows 10, 11 and 14 of the table, we use a lowercase *t* separator (instead of the standard *T* separator) between the date and time parts of the representation. Together, these notational conventions indicate explicitly those parts of a temporal value that remain uninstantiated.

A few other elements of this representation are worthy of mention. The local semantics of bare weekday names, such as *Monday* or *Friday*, can not be represented in the standard month-based format *yyyy-mm-dd*, and therefore must be represented in the week-based format *yyyy-Wnn-d*, where *nn* is the ISO week number and *d* is the number of the weekday within that week (1 denotes Monday and 7 is used for Sunday).

C. Offset Expressions

Offset expressions, as we call them, encode a function which, when applied to a *reference time*, returns the global semantic value denoting the temporal location of the entity referred to by the expression. This temporal function either adds or subtracts a number of units at some granularity: for example, *last*

⁴Underspecified expressions should not be confused with non-specific expressions; these represent two quite independent semantic phenomena. The former omits some information because it is assumed the reader will be able to retrieve it from the context (e.g. *14th June*), while the latter is typically used generically (e.g. 'The dry season starts in *May*'). In the string-based semantic representation, the underspecified expressions we introduce in LTIMEX use lowercase *xs* (e.g. *xxxx-06-14*), while non-specific expressions, already part of TIMEX2, are annotated with uppercase *Xs* (e.g. *XXXX-WXX-7TMO*).

TABLE VI
LOCAL SEMANTICS OF OFFSET EXPRESSIONS OF DATES.

No	Deictic Expression	Anaphoric Expression	L-VAL
1	today	the same day	+0000-00-00
2	tomorrow	the following day	+0000-00-01
3	yesterday	the previous day	-0000-00-01
4	five days ago	five days earlier	-0000-00-05
5	last month	the previous month	-0000-01
6	last summer	the previous summer	-0001-SU
7	two weeks ago	two weeks earlier	-0000-W02
8	(in) two weeks	two weeks later	+0000-W02
9	this weekend	that weekend	+0000-W00-WE
10	this year	that year	+0000
11	three years ago	three years earlier	-0003
12	next century	the following century	+01

year is equivalent to subtracting one year from the year of the reference date, and *three days later* means adding three days.

Based on the reference time used, we further distinguish two kinds of offsets: these may be either *deictic* or *anaphoric*. For deictic expressions the reference time is the time-stamp of the utterance⁵ (*S* in the Reichenbachian framework [12, pp. 291–298]) and for anaphoric expressions the reference is to be found somewhere in the context. For example, *yesterday* is deictic, but *the previous day* is an anaphoric expression.

In LTIMEX, both these expressions have the same offset encoded as the value of the L-VAL attribute; the L-TYPE attribute indicates whether the expression is *OFFSET-DEICTIC* or *OFFSET-ANAPHORIC*. The interpretation algorithm can use the value of this attribute to decide whether to apply the offset to the time-stamp of the document or to use a temporal focus tracking mechanism to select the correct reference time. If, for any reason, the annotator or a temporal expression tagger cannot decide on the subtype of the offset, L-TYPE can be specified simply as *OFFSET*, leaving the decision about the subtype to the interpretation module.

Table VI presents pairings of deictic and anaphoric date expressions which share the same value for the L-VAL attribute. A leading + or - indicates whether the operation to be performed is addition or subtraction; for consistency with TIMEX2, we use the ISO-based format to encode the magnitude of the offset. The number of filled slots determines the granularity of the unit of the operation. For example, +0000-00-05 encodes the addition of five days and -0002 encodes the subtraction of two years. Of course, for expressions with zero offset (e.g. *today*) one could use either + or -; by convention we use +.

An offset date expression may be accompanied by unambiguous (e.g. *6 a.m.*) or ambiguous (e.g. *6 o'clock*) information about the time within the referred-to day; see Rows 1–9 of Table VII. In these cases only the date component of the expression (e.g. *today* or *tomorrow*) has the form of an offset; here the *T* and *t* separators combine an offset on their left with an absolute value on their right.

⁵In practical terms, the utterance time may be the time of speaking, the date of publication, the date and time of sending an email, and so on.

TABLE VII
THE LOCAL SEMANTICS OF OFFSET EXPRESSIONS WITH REFERENCES TO TIMES OF DAY.

No	Expression	Representation (L-VAL)	Type
1	6 a.m. today	+0000-00-00T06:00	deictic date offset + explicit time
2	6 p.m. that day	+0000-00-00T18:00	anaphoric date offset + explicit time
3	6 p.m. two days ago	-0000-00-02T18:00	deictic date offset + explicit time
4	6 o'clock two days ago	-0000-00-02t06:00	deictic date offset + underspecified time
5	tomorrow morning	+0000-00-01TMO	deictic date offset + explicit time
6	morning the day before	-0000-00-01TMO	anaphoric date offset + explicit time
7	last night	-0000-00-01TNI	deictic date offset + explicit time
8	11pm last night	-0000-00-01T23:00	deictic date offset + explicit time
9	2am last night	+0000-00-00T02:00	deictic date offset + explicit time
10	two hours earlier	+0000-00-00T-02	anaphoric time offset
11	an hour and twenty minutes later	+0000-00-00T+01:20	anaphoric time offset
12	in six hours time	+0000-00-00T+06	deictic time offset
13	five minutes ago	+0000-00-00T-00:05	deictic time offset
14	in sixty seconds	+0000-00-00T+00:00:60	deictic time offset
15	sixty seconds later	+0000-00-00T+00:00:60	anaphoric time offset
16	tomorrow two hours later	+0000-00-01T+02	deictic date offset + time offset
17	the next day two hours later	+0000-00-01T+02	anaphoric date offset + time offset
18	8 May 2001, one hour later	2001-05-08T+01	explicit date + time offset
19	17 May, one hour earlier	xxxx-05-17T-01	underspecified date + time offset

Just as there can be date offsets that have no time information, we can also have time offsets with no date information; for example, *five minutes ago*. In such cases we add the operator (+ or -) just after the T separator. Consider the representation of *eighteen hours and fifteen minutes later*: this has the value +0000-00-00T+18:15, making it distinct from the representation of *6:15pm today*, which is +0000-00-00T18:15. More examples are provided in Rows 10–15 of Table VII.

A time offset may also appear together with a date offset, as shown in Rows 16 and 17 in Table VII, or even with an explicit or underspecified date, as shown in Rows 18 and 19. In the first case the representation combines a non-zero date offset with a time offset; in the second case we have a non-offset representation of a date followed by the encoding of the time offset.

Finally, we also need to be able to represent offset expressions built on cycle-based calendar elements (weekday and month names), such as *last Monday* or *next March*. In Table VIII we present examples with the proper encodings. In our representation we only indicate the direction (< for *last*, > for *next*) and the weekday or month name mentioned in the expression (D_n and M_{nn} , respectively). It will be the task of the interpretation stage to determine which calendar week and year is intended. Expressions using the determiner *this* (e.g. *this Wednesday* or *this June*) are treated as underspecified expressions unless the determiner is used together with other tokens indicating the direction of interpretation (e.g. *this coming Wednesday*).

D. Event-based Point Expressions

An event-based expression identifies a temporal entity by means of a reference to an event. In such expressions, the L-TYPE attribute has the value

TABLE VIII
THE LOCAL SEMANTICS OF OFFSET EXPRESSIONS INVOLVING ELEMENTS OF CYCLE-BASED CALENDARS.

No	Deictic Expression	Anaphoric Expression	L-VAL
1	last Monday	the previous Monday	<D1
2	next Wednesday	the next Wednesday	>D3
3	this coming Wednesday	that coming Wednesday	>D3
4	this Wednesday	that Wednesday	xxxx-Wxx-3
5	last June	the previous June	<M06
6	next June	the next June	>M06
7	this June	that June	xxxx-06

L-TYPE=EVENT-BASED-POINT, and we provide the identifier of the event in the L-EVENT_ID attribute. If an application does not perform event recognition, or in a given circumstance is unable to identify the event in question, then the value of this attribute is left empty.

In some cases the temporal value is expressed as an offset to the time of an event, as in Example (2):

- (2) *Ten seconds after the second explosion* the plane hit the ground.
L-VAL=+0000-00-00T+00:10
L-TYPE=EVENT-BASED-POINT L-EVENT_ID=e
- (3) Jane got a salary raise *the day after Michael lost his job*.
L-VAL=+0000-00-01 L-TYPE=EVENT-BASED-POINT
L-EVENT_ID=e

Here the L-VAL attribute encodes the offset, just as it does in offset point expressions. The specified type of the expression indicates that the reference time to be used in the interpretation is the time of the event indicated by the e event variable.

In cases when the time denoted by the expression can be computed from the time-stamp of the event simply by refining its granularity, we use a zero-offset just to indicate the granularity (temporal unit) of the result. Consider the

following example:

- (4) I met my wife *the year when I bought my house*.
 L-VAL=+0000
 L-TYPE=EVENT-BASED-POINT L-EVENT_ID=e

The temporal value of the expression is to be calculated here by adding zero years to the year of the event time-stamp, and discarding any more detailed information that the time-stamp might provide (e.g. the month and day).

In other cases, the time denoted by the expression may be exactly the time of the event, as in the following example:

- (5) *At the time of the peace agreement* the United States agreed to replace equipment on a one-by-one basis.
 L-VAL=EVENT_TIME
 L-TYPE=EVENT-BASED-POINT L-EVENT_ID=e

Note that the expression does not indicate the granularity. In such cases, the L-VAL attribute contains the EVENT_TIME token, which means that the temporal value is the time of the underlying event.

For point temporal expressions which refer to a part of an event, as in Example (6), we use the encoding of ordinaly-specified expressions, which we discuss in detail in Section III-G:

- (6) The casualties included 19,240 dead on *the third day of the Battle of the Somme*.
 L-VAL=3D L-TYPE=EVENT-BASED-POINT
 L-EVENT_ID=e

The 3D value tells us that, of the whole time span of the event, the expression refers only to the third day.

E. Period Expressions

For expressions that denote periods, L-VAL takes the same values as the corresponding VAL attribute in TIMEX2; this also covers those cases where the duration mixes different units, as in the following example:

- (7) This project will run for *one year and two months*.
 L-VAL=P1Y2M

The anchoring attributes are to be filled in only if the anchor is mentioned within the extent of the expression. The anchor may be provided in various forms, including an explicit (see Example (8)), underspecified (see Example (9)) or offset (see Examples (10) and (11)) point. In each case, the L-ANCHOR_VAL attributes encode that anchoring point in one of the formats we have already introduced:

- (8) The accounts are paid in full for *the six months ended 31 March 2009*.
 L-VAL=P6M
 L-ANCHOR_VAL=2009-03-31 L-ANCHOR_DIR=ENDING
- (9) The accounts are paid in full for *the six months ended March 31*.
 L-VAL=P6M
 L-ANCHOR_VAL=xxxx-03-31 L-ANCHOR_DIR=ENDING
- (10) The renovations will last *five days starting tomorrow*.
 L-VAL=P5D L-ANCHOR_TYPE=DEICTIC
 L-ANCHOR_VAL=+0000-00-01
 L-ANCHOR_DIR=STARTING

- (11) The movie festival will end on *18 July*, but then we have the theatre workshops that will run for *a whole week starting just the very next day*.

L-VAL=P1W L-ANCHOR_TYPE=ANAPHORIC
 L-ANCHOR_VAL=+0000-00-01
 L-ANCHOR_DIR=STARTING

In the last example above we also use the L-ANCHOR_TYPE attribute to encode the type of the offset of the anchor; the possible values here are DEICTIC and ANAPHORIC. The expression may be also anchored implicitly, as in the following example:

- (12) *The next three days* were extremely hot and humid.
 L-VAL=P3D
 L-ANCHOR_VAL=+0000-00-00
 L-ANCHOR_DIR=STARTING

In such cases we provide the offset in the L-ANCHOR_VAL attribute, but leave it to the interpretation algorithm to decide (for example, based on the tense of the sentence) whether the anchor is deictic or anaphoric.

If the expression itself does not state when the period starts or ends, then no anchor-related attributes are specified:

- (13) The Nile Movie Festival lasted *five days*. L-VAL=P5D

If the rest of the document provides such information, the anchor is to be determined in the interpretation stage, when the global semantics is derived; this also means that the final annotation does not have the L-ANCHOR_VAL and L-ANCHOR_DIR attributes, it only has ANCHOR_VAL and ANCHOR_DIR.

F. Event-based Period Expressions

For event-based periods, we encode the duration in the L-VAL attribute just as in the case of other durations discussed in Section III-E, but the type of the expression in the L-TYPE attribute is specified as EVENT-BASED-PERIOD. Similarly to the annotation of event-based point expressions, we provide the identifier of the underlying event in the L-EVENT_ID attribute. The time of the event, however, does not serve here as the reference time to be used in the following interpretation stage to calculate the value of the VAL attribute; rather, it determines the location of the period, and is used to compute one of the period's anchors. Consider the following examples:

- (14) The rate of US combat deaths in Baghdad nearly doubled in *the first seven weeks of the "surge" in security activity*.
 L-VAL=P7W
 L-ANCHOR_VAL=EVENT_START
 L-ANCHOR_DIR=STARTING
 L-EVENT_ID=e L-TYPE=EVENT-BASED-PERIOD
- (15) *The last three days of the battle* were extremely brutal.
 L-VAL=P3D
 L-ANCHOR_VAL=EVENT_END L-ANCHOR_DIR=ENDING
 L-EVENT_ID=e L-TYPE=EVENT-BASED-PERIOD
- (16) I was so panicked I could not take a single step for *30 minutes after the earth quake*.
 L-VAL=PT30M
 L-ANCHOR_VAL=EVENT_END

TABLE IX
THE LOCAL SEMANTICS OF ORDINALLY-SPECIFIED REFERENCES.

No	Expression	Representation (L-VAL)
1	the first Tuesday	1D2
2	the third day	3D
3	the last Tuesday	\$1D2
4	the last day	\$1D
5	the last but one day	\$2D
6	the penultimate day	\$2D
7	the last month	\$1M
8	the last February	\$1M02

L-ANCHOR_DIR=STARTING

L-EVENT_ID=e L-TYPE=EVENT-BASED-PERIOD

- (17) There was no terrorist warning in *the three years before the bombing in the underground.*

L-VAL=P3Y

L-ANCHOR_VAL=EVENT_START

L-ANCHOR_DIR=ENDING

L-EVENT_ID=e L-TYPE=EVENT-BASED-PERIOD

To handle the fact that events themselves span over some periods of time, we introduce two tokens, `EVENT_START` and `EVENT_END`, to be used in the `L-ANCHOR_VAL` attribute. These tokens indicate which end of the period of the event is to be used as the anchor.

G. Ordinally-specified expressions

Some temporal expressions use what we call *ordinally-specified elements*; for example, the expressions in Examples (18a)–(18c) make reference to a specific day by means of selecting the third day of some coarser temporal unit or an event.

- (18) a. *the third day of the next month*
 b. *the third day of the previous decade*
 c. *the third day of the trip*

To encode such ordinally-specified elements we use the format $\$nu$, where n is a number, u indicates the temporal unit to be used, and $\$$ is an optional marker used when the ordinal is to be counted from the end of some chunk of time (e.g. *last*, *penultimate*). Examples of ordinally-specified elements of expressions and their representations are shown in Table IX.

Expressions using ordinally-specified elements are annotated with multiple TIMEX2 annotations, as shown in Examples (19)–(21). The ordinally-specified format is recorded only in the outermost annotation; the inner expression, which may be, for example, underspecified or an offset, receives its own proper representation of its local semantics.

- (19) <TIMEX2 L-VAL="3D">*the third day of*
 <TIMEX2 L-VAL="+0000-01">*the next month*
 </TIMEX2></TIMEX2>
- (20) <TIMEX2 L-VAL="\$1D1">*the last Monday of* <TIMEX2
 L-VAL="xxxx-05">*May*</TIMEX2></TIMEX2>
- (21) <TIMEX2 L-VAL="1D">*the first day of*
 <TIMEX2 L-VAL="2M">*the second month of*

<TIMEX2 L-VAL="+0001">*next year*</TIMEX2>
 </TIMEX2></TIMEX2>

When deriving the global semantics from the local semantics, the individual values of the nested expressions must be combined together; the process is carried out recursively from the outermost to the innermost, resolving the temporal references while backtracking from the innermost to the outermost.

The type recorded in the `L-TYPE` of an expression whose `L-VAL` is ordinally specified is the same as the type of its innermost expression; Example (19) is an anaphoric offset, Example (20) is underspecified, and Example (21) is deictic.

H. Non-Specific Expressions

In many cases the decision that a temporal expression is non-specific can be only made when analysing the whole sentence, or even the entire document. For example, consider the generic references to months in the following sentence:

- (22) In the southern hemisphere *days* are much longer in *January* than in *July*.

These are not obviously non-specific when we consider only the extent of the expressions themselves. The local semantic representations are therefore underspecified, i.e. `xxxx-01` and `xxxx-07`. In the interpretation stage, the lowercase `xs` must not be instantiated with a specific year, but must be converted into markers of non-specificity (uppercase `Xs`, for example, if TIMEX2 is the scheme used for global semantics representation).

Indefinite noun phrases, on the other hand, can already be recognized as non-specific at the level of local semantics:

- (23) a. I was born on *a Sunday*. L-VAL=XXXX-WXX-7
 b. I met my wife on *a sunny day in July*.
 L-VAL=XXXX-07-XX

In such cases, we can already mark the relevant slots as non-specific, obtaining the same value as expected in `VAL`.

Periods of indefinite duration, such as *a few days*, can also be recognized as non-specific without reference to the context. The encoding of such durations uses `X` instead of a specific number, e.g. `PXD`.

Similarly, some set expressions can be identified as non-specific already at the stage of local semantic analysis; for example, *every few days* or *some Mondays in 2004*. Unfortunately, TIMEX2 is unable to represent the semantics of these expressions correctly,⁶ and in consequence our representation fails here too.

I. Set Expressions

The semantic representation of set expressions is complex, because these expressions do not refer to a single entity, but to a set of entities. Neither TIMEX2 nor TimeML express the semantics of these expressions sufficiently well to make these schemes applicable to all set expressions. As an alternative, Pan's [13] first-order logic representation for set expressions, which is formally sound and has much broader coverage, can

⁶For example, *some Mondays in 2004* is represented just in the same way as *all Mondays in 2004*: `VAL=2004-WXX-1`, `SET=YES`.

be encoded in OWL; but the complexity of OWL goes far beyond the goals of TIMEX2 and TimeML.

As indicated earlier, our aim is to provide a representation for local semantics that is compatible with the use of TIMEX2 for representing the global semantics of temporal expressions. Inevitably, this compromises the expressiveness of our representation.

We indicate the set type by assigning the value YES to the L-SET attribute (following the use of the SET attribute in TIMEX2), and we specify any underspecification or offset that might appear, as in the following examples:

- (24) a. *every winter in the 80s* L-VAL=xx8-WI L-SET=YES
 b. *monthly* L-VAL=xxxx-XX L-SET=YES

In some cases we may be able to obtain a reliable representation by using values or attribute combinations not authorized in TIMEX2. For instance, in Example (25a) the expression is represented by means of any period of two years with its ending anchored on years having zero as their final digit (e.g. 1960, 1990, 2000). In Example (25b) we do something similar, but we anchor the periods on the last day of a month (and in doing so we specify the anchor with the format used for ordinal-specified references).

- (25) a. *the last two years of every decade*
 L-VAL=P2Y L-SET=YES
 L-ANCHOR_VAL=XXX0 L-ANCHOR_DIR=ENDING
 b. *the last two days of every month*
 L-VAL=P2D L-SET=YES
 L-ANCHOR_VAL=XXXX-XX-\$1D
 L-ANCHOR_DIR=ENDING

This, however, already goes beyond the TIMEX2 rules, which prohibit using the anchor attributes for set expressions [2, p. 42].

IV. CONCLUSION

We have developed a string-based representation of the context-independent semantics of temporal expressions, which we call LTIMEX. It can be easily integrated with the existing annotation schemes (specifically, TIMEX2 and TimeML) which currently allow only for the representation of fully-interpreted semantics. We are thus proposing an extension to these schemes that provides a means of support for an additional level of semantic representation; this in turn leads to a modular design of temporal expression tagging, with a well-defined interface between the recognition and interpretation modules, and allows for more detailed evaluation of taggers.

Table II summarises the attributes used in LTIMEX and their values. We use in total eight attributes: three are used in the same way as their TIMEX2 counterparts (L-MOD, L-SET and L-ANCHOR_DIR); L-VAL represents the partial⁷ context-independent meaning of the expression; similarly, L-ANCHOR_VAL encodes information about the temporal location of an anchor of a period; and three attributes are

⁷It is partial in the sense that it does not capture information about temporal modifiers and anchors, which are encompassed in separate attributes: L-MOD, L-ANCHOR_DIR, and L-ANCHOR_VAL.

completely new: L-TYPE, which encodes the taxonomical type of the expression; L-EVENT_ID, which for event-based expressions stores the identifier of the event; and L-ANCHOR_TYPE, which, for durations with the anchor expressed by means of an offset, encodes whether it is deictic or anaphoric.

The first obvious task that arises as possible future work is to use LTIMEX for a significant data annotation task; possible candidate corpora already annotated with temporal expressions are WikiWars [14] (TIMEX2) and TimeBank⁸ (TimeML).

Another area left for future work is the improvement in the representation of set expressions. This could perhaps be aligned with further development of TimeML, which is to become an ISO standard (see the discussion in [15]); although this goes further than TIMEX2, it still does not have a proper means to represent the global semantics of set expressions.

REFERENCES

- [1] O. R. Alonso, "Temporal Information Retrieval," Ph.D. dissertation, University of California, 2008.
- [2] L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson, "TIDES 2005 Standard for the Annotation of Temporal Expressions," MITRE, Tech. Rep., September 2005.
- [3] J. Pustejovsky, J. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz, "TimeML: Robust Specification of Event and Temporal Expressions in Text," in *IWCS-5, Fifth International Workshop on Computational Semantics*, Tilburg, The Netherlands, January 2003.
- [4] R. Saurí, J. Littman, B. Knippen, R. Gaizauskas, A. Setzer, and J. Pustejovsky, "TimeML Annotation Guidelines Version 1.2.1," January 2006. [Online]. Available: <http://www.timeml.org/site/publications/specs.html>
- [5] NIST, "The ACE 2004 Evaluation Plan," 2004, www.itl.nist.gov/iad/mig/tests/ace/2004/doc/ace04-evalplan-v7.pdf.
- [6] I. Mani and G. Wilson, "Robust Temporal Processing of News," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL '00)*, Morristown, NJ, USA: Association for Computational Linguistics, October 2000, pp. 69–76.
- [7] F. Schilder, "Extracting Meaning from Temporal Nouns and Temporal Prepositions," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 3, no. 1, pp. 33–50, March 2004.
- [8] D. Ahn, J. van Rantwijk, and M. de Rijke, "A Cascaded Machine Learning Approach to Interpreting Temporal Expressions," in *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2007)*, Rochester, NY, USA, April 2007.
- [9] P. Mazur and R. Dale, "The DANTE Temporal Expression Tagger," in *Proceedings of the 3rd Language And Technology Conference (LTC)*, Z. Vetulani, Ed., Poznan, Poland, October 2007.
- [10] J. Strötgen and M. Gertz, "HeidelTime: High Quality Rule-Based Extraction and Normalization of Temporal Expressions," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: ACL, July 2010, pp. 321–324.
- [11] P. Mazur and R. Dale, "An Intermediate Representation for the Interpretation of Temporal Expressions," in *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*. Sydney, Australia: Association for Computational Linguistics, July 2006, pp. 33–36.
- [12] H. Reichenbach, *Elements of Symbolic Logic*. Macmillan, 1947.
- [13] F. Pan, "Representing Complex Temporal Phenomena for the Semantic Web and Natural Language," Ph.D. dissertation, University of Southern California, 2007.
- [14] P. Mazur and R. Dale, "Wikiwars: A new corpus for research on temporal expressions," in *Proceedings of the EMNLP 2010, Conference on Empirical Methods in Natural Language Processing*, 2010.
- [15] J. Pustejovsky, K. Lee, H. Bunt, and L. Romary, "ISO-TimeML: An International Standard for Semantic Annotation," in *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA), May 2010.

⁸See the catalogue entry LDC2006T08 at <http://www ldc.upenn.edu>.